
Gosh's LDID's

The [DeleteMe] section removes an entire key and ALL of it values

Additions must use the format: MainRegistryKey,"SubKeyToDelete"

Valid MainRegistryKey abbreviations are:
HKCR = Hkey_Classes_Root
HKCU = Hkey_Current_User
HKLM = Hkey_Local_Machine
HKU = Hkey_Users

Registry Data Types
types with NCLOB don't overwrite existing values

REG_SZ = 0x00000000 (or 0)
REG_BINARY = 0x00000001 (or 1)
Reg_SZ_NOCLOBBER = 0x00000002 (or 2)
REG_BINARY_NOCLOBBER = 0x00000003 (or 3)
REG_MULTI_SZ_APPEND = 0x0001000A
REG_MULTI_SZ = 0x00010000
REG_DWORD = 0x00010001
REG_MULTI_SZ_NOCLOBBER = 0x00010002
REG_DWORD_NOCLOBBER = 0x00010003
REG_MULTI_SZ_DELVAL = 0x00010006
REG_ADDREG_APPEND = 0x00010008
REG_EXPAND_SZ = 0x00020000
REG_EXPAND_SZ_NOCLOBBER = 0x00020002

To make a default value, use ,,, For example see iesearch settings

A DIRID can be one of the following values: -01,

Value	Destination Directory
01	SourceDrive:\pathname (the directory from which the INF file was installed)
10	Windows directory This is equivalent to %windir%.
11	System directory This is equivalent to %windir%\system32 for NT-based systems, and to %windir%\system for Windows 9x/Me.
12	Drivers directory This is equivalent to %windir%\system32\drivers for NT-based platforms, and to %windir%\system\IoSubsys on Windows 9x/Me platforms.
17	INF file directory
18	Help directory
20	Fonts directory
21	Viewers directory
23	Color directory (ICM) (not used for installing printer drivers)
24	Root directory of the system disk.

This is the root directory of the disk on which Windows files are installed.
For example, if dirid 10 is "C:\winnt", then dirid 24 is "C:\".

25 Shared directory

30 Root directory of the boot disk, also known as "ARC system partition,"
for NT-based systems. (This might or might not be the same directory as
the one represented by dirid 24.)

50 System directory for NT-based operating systems
This is equivalent to %windir%\system (NT-based systems only).

51 Spool directory (not used for installing printer drivers see Printer Dirids)
52 Spool drivers directory (not used for installing printer drivers)
53 User profile directory
54 Directory where ntldr.exe and osloader.exe are located (NT-based systems only)
55 Print processors directory (not used for installing printer drivers)
-1 Absolute path

Value	Shell Special Folder
16384	%userprofile%\Desktop
16386	%userprofile%\Start Menu\Programs
16389	%userprofile%\My Documents
16390	%userprofile%\Favorites
16391	%userprofile%\Start Menu\Programs\Startup
16392	%userprofile%\Recent
16393	%userprofile%\SendTo
16395	%userprofile%\Start Menu
16397	%userprofile%\My Documents\My Music
16398	%userprofile%\My Documents\My Videos
16400	%userprofile%\Desktop
16403	%userprofile%\NetHood
16404	C:\WINDOWS\Fonts
16405	%userprofile%\Templates
16406	%allusersprofile%\Start Menu
16407	%allusersprofile%\Start Menu\Programs
16408	%allusersprofile%\Start Menu\Programs\Startup
16409	%allusersprofile%\Desktop
16410	%userprofile%\Application Data
16411	%userprofile%\PrintHood
16412	%userprofile%\Local Settings\Application Data
16415	%allusersprofile%\Favorites
16416	%userprofile%\Local Settings\Temporary Internet Files
16417	%userprofile%\Cookies
16418	%userprofile%\Local Settings\History
16419	%allusersprofile%\Application Data
16420	System Root (C:\WINDOWS)
16421	C:\WINDOWS\System32
16422	Program Files
16423	%userprofile%\My Documents\My Pictures
16424	%userprofile%
16425	C:\WINDOWS\System32
16427	C:\Program Files\Common Files
16429	%allusersprofile%\Templates

16430	%allusersprofile%\Documents
16431	%allusersprofile%\Start Menu\Programs\Administrative Tools
16432	%userprofile%\Start Menu\Programs\Administrative Tools
16437	%allusersprofile%\Documents\My Music
16438	%allusersprofile%\Documents\My Pictures
16439	%allusersprofile%\Documents\My Videos
16440	C:\WINDOWS\resources
16441	C:\WINDOWS\resources\0409
16443	%userprofile%\Local Settings\Application Data\Microsoft\CD Burning

ProfileItems

0x00000001	Create menu item in Current User flag
0x00000002	Delete menu item flag
0x00000004	Create Start menu group under All Users\Start\Programs
0x00000005	Create Start menu group under Current User\Start\Programs
0x00000006	Delete Start menu group under All Users\Start\Programs
0x00000007	Delete Start menu group under Current User\Start\Programs

Advanced ProfileItems (by rickendo64)

Flag Location

0x00000008,2 = "%UserProfile%\Start Menu\Programs"
0x00000008,5 = "%UserProfile%\My Documents"
0x00000008,6 = "%UserProfile%\Favorites" <- For adding custom IE Bookmarks
0x00000008,7 = "%UserProfile%\Start Menu\Programs\Startup"
0x00000008,8 = "%UserProfile%\Recent"
0x00000008,9 = "%UserProfile%\SendTo"
0x00000008,11 = "%UserProfile%\Start Menu"
0x00000008,13 = "%UserProfile%\My Documents\My Music"
0x00000008,14 = "%UserProfile%\My Documents\My Videos"
0x00000008,16 = "%UserProfile%\Desktop"
0x00000008,19 = "%UserProfile%\NetHood"
0x00000008,20 = "%WinDir%\Fonts"
0x00000008,21 = "%UserProfile%\Templates"
0x00000008,22 = "%AllUsersProfile%\Start Menu"
0x00000008,23 = "%AllUsersProfile%\Start Menu\Programs"
0x00000008,24 = "%AllUsersProfile%\Start Menu\Programs\Startup"
0x00000008,25 = "%AllUsersProfile%\Desktop"
0x00000008,26 = "%UserProfile%\Application Data" <- Perfect for QuickLauch
0x00000008,27 = "%UserProfile%\PrintHood"
0x00000008,28 = "%UserProfile%\Local Settings\Application Data"
0x00000008,31 = "%AllUsersProfile%\Favorites"
0x00000008,32 = "%UserProfile%\Local Settings\Temporary Internet Files"
0x00000008,33 = "%UserProfile%\Local Settings\Cookies"
0x00000008,34 = "%UserProfile%\Local Settings\History"
0x00000008,35 = "%AllUsersProfile%\Application Data"
0x00000008,36 = "%WinDir%"
0x00000008,37 = "%WinDir%\system32"
0x00000008,38 = "%ProgramFiles%"

0x00000008,39 = "%UserProfile%\My Documents\My Pictures"
0x00000008,40 = "%UserProfile%"
0x00000008,41 = "%WinDir%\system32"
0x00000008,43 = "%CommonProgramFiles%"
0x00000008,45 = "%AllUsersProfile%\Templates"
0x00000008,46 = "%AllUsersProfile%\Documents"
0x00000008,47 = "%AllUsersProfile%\Start Menu\Programs\Administrative Tools"
0x00000008,48 = "%UserProfile%\Start Menu\Programs\Administrative Tools"
0x00000008,53 = "%AllUsersProfile%\Documents\My Music"
0x00000008,54 = "%AllUsersProfile%\Documents\My Pictures"
0x00000008,55 = "%AllUsersProfile%\Documents\My Videos"
0x00000008,56 = "%WinDir%\Resources"
0x00000008,57 = "%WinDir%\Resources\0409"
0x00000008,59 = "%UserProfile%\Local Settings\Application Data\Microsoft\CD Burning"

Delete Advanced ProfileItems

0x0000000A,XX (Change the "8" to the letter "A" and "XX" with one of the above numbers corresponding to your shortuct)

Complete Advanced INF

```
[Version]
Signature="$CHICAGO$"
AdvancedINF=2.5,%BadAdvpackVer%

[SourceDisksNames]
10="testDisk",,0

[SourceDisksFiles]
newfile.txt=10
inseng.dll=10
adv.inf=10

[DefaultInstall]
;existing gen install INF options
Copyfiles=CopyFilesSection
RenFiles=RenFilesSection
DelFiles=DelFilesSection
UpdateInis=UpdateInisSection
UpdateIniFields=UpdateIniFieldsSection
AddReg=AddRegSection
DelReg=DelRegSection
Ini2Reg=Ini2RegSection
UpdateCfgSys=UpdateCfgSysSection
UpdateAutoBat=UpdateAutoBatSection

;advanced INF options
RequiredEngine=[SETUPAPI | SETUPX],%BadSetupEngineVer%
CustomDestination=CustomDestinationSection
RegisterOCXs=RegisterOCXsSection
UnregisterOCXs=RegisterOCXsSection
BeginPrompt=BeginPromptSection
EndPrompt=EndPromptSection
RunPreSetupCommands=RunPreSetupCommandsSection
RunPostSetupCommands=RunPostSetupCommandsSection
```

```
SmartReboot=[I | A | N]
DelDirs=DelDirsSection
Cleanup=1
CheckAdminRights=[1 | 0]

;advanced INF options needed for save/rollback only
ComponentName=%Name%
ComponentVersion=4.0
BackupReg=BackupRegSection
PreRollBack=GenInstallSection  ;optional parameter
BackupPath=%49000%\%UninstallData%  ;optional

;Per User install options
PerUserInstall=PerUserInst

[PerUserInst]
DisplayName=%WebInteg%
Version=5.0.0311.0
IsInstalled= [1==installed | 0==uninstalled]
ComponentID=IE4Shell_WIN ;ComponentID in ActiveSetup .CIF file
GUID={89820200-ECBD-11cf-8B85-00AA005B4395}
Locale=en
StubPath=rundll32.exe advpack.dll,LaunchINFSectionEx %17%\inst.inf,
InstSec,,36

[DestinationDirs]
CopyFilesSection=11
DelFilesSection=17
RenFilesSection=17

[CustomDestinationSection]
49000,49012,49011,49010=DestA,1

[DestA]
HKLM,SOFTWARE\Test,TestLoc,"%Prompt1%",%17%
;HKLM,SOFTWARE\test2,"Blah",'%Prompt2%',"c:\test"
;HKLM,SOFTWARE\Microsoft\test3,,'%Prompt3%',"%11%"

[RegisterOCXsSection]
@%inat%
%11%\textfxr.ocx
```

```
%11%\inseng.dll
%11%\adesktop.dll
%11%\shdocvw,I,shell
or
%11%\shdocvw,I,web

[AddRegSection]
HKLM,"SOFTWARE\Microsoft\Active Setup\Installed Components\%GUID
%",,,"My Component"
HKLM,"SOFTWARE\Microsoft\Active Setup\Installed Components\%GUID
%","IsInstalled",1,01,00,00,00
HKLM,"SOFTWARE\Microsoft\Active Setup\Installed Components\%GUID
%","Version",,"4,72,1605,0"

[DelRegSection]
HKLM,"SOFTWARE\Microsoft\Active Setup","SteppingMode","Y"

[DelDirsSection]
%49000%

[RunPreSetupCommandsSection]
calc.exe

[RunPostSetupCommandsSection]
DelFile c:\windows\inf\inseng.dll

[BeginPromptSection]
Prompt=%BeginPrompt%
ButtonType= [YESNO | OKCANCEL]
Title="Test INF Install"

[EndPromptSection]
Prompt=%EndPrompt%

[CopyFilesSection]
inseng.dll,,,16
adv.inf

[RenFilesSection]
newfile.txt,oldfile.txt
```

```
[DelFilesSection]
oldfile.txt,,,1

[UpdateInisSection]
win.ini,iniSection

[UpdateIniFieldsSection]
system.ini,iniSection,profile

[Ini2RegSection]
system.ini,iniSection,,HKLM,subkey

[UpdateCfgSysSection]
Buffers=40

[UpdateAutoBatSection]
TmpDir=C:\Temp

[BackupRegSection]
HKLM,"SOFTWARE\Microsoft\Active Setup"

[GenInstallSection]
AddReg=NewGenAddRegSection
DelReg=NewGenDelRegSection

[Strings]
BadAdvpackVer="Incorrect version of advpack.dll. Please get new
version from our web site."
BadSetupapiVer="Setupapi.dll is required to install on this system."
BadSetupEngineVer="Advpack.dll is required to install on this
system." BeginPrompt="Are you sure you want to proceed with install?"
EndPrompt="Install has completed successfully. NEW"
Prompt1="1: IS App''s location right?"
Prompt2="2: Where would you like to install?"
Prompt3="3: Where would you like to go today?"
UninstallData="Uninstall Information"
GUID="MyGuid"
```

Version Section

Version

[Version]
Signature="$Chicago$"
LayoutFile=*filename.inf*
AdvancedINF=*2.5, failure-dialog-string*

Defines the standard header for all Windows 95 or NT INF files. Note that if the signature is not $Chicago$ then Windows will not accept the INF file as an INF file for any of the classes of devices recognized by Windows 95 and NT.

Note the signature string recognition is case-insensitive. So, for example, you can use either $Chicago$ or $CHICAGO$.

filename.inf
Names the INF file that contains the layout information (source disks and files) required to install this component. This line is optional. If not given, the SourceDisksNames and SourceDisksFiles sections must be given in this INF.

AdvancedINF
Identifies the version of Advpack.dll that Internet Explorer 4.0 must load in order to parse this INF file. In this case, version 2.5 is required. If the version of Advpack.dll is not found, the *failure-dialog-string* is presented to the user.

SourceDisksNames Section

SourceDisksNames

[SourceDisksNames]
disk-ordinal="disk-description",disk-label,disk-serial-number

Identifies and names the disk(s) that contain the source files for file copying and renaming operations. Used along with SourceDisksFiles.

disk-ordinal
A unique number that identifies a source disk. If there is more than one source disk, each must have a unique ordinal.

disk-description
A string or a strings key describing the contents or purpose of the disk. The installer displays this string to the user to identify the disk. The description is enclosed in double-quotation marks.

disk-label
Volume label of the source disk that is set when the source disk is formatted.
This is also where you name the cab file if the files are in a cab.

disk-serial-number
Unused. Value must be 0.

This example identifies one source disk. The disk description is given as a strings key:

```
[SourceDisksNames]
55 = %ID1%, Instd1, 0

[Strings]
ID1="My Application Installation Disk 1"
```

SourceDisksFiles Section

SourceDisksFiles

[SourceDisksFiles]
filename=disk-number(,*subdir*)(,*file-size*)

Names the source files used during installation and identifies the source disks that contain the files.

filename
Name of the file on the source disk.

disk-number
Ordinal of the source disk that contains the file. This ordinal must be defined in the SourceDisksNames section, and must have a value greater than or equal to 1 (zero is not a valid disk-number parameter value).

subdir
Optional parameter that specifies the subdirectory on the source disk where the file resides. If this parameter is not used, the source disk root directory is the default.

file-size
Optional parameter that specifies the size of the file, in Bytes.

This example SourceDisksFiles section identifies a single source file, SRS01.386, on the disk having ordinal 1:

```
[SourceDisksFiles]
SRS01.386 = 1
```

Install Sections

Install and DefaultInstall

[install-section-name]
CopyFiles=*file-list-section*
RenFiles=*file-list-section*
DelFiles=*file-list-section*
UpdateInis=*update-ini-section*
UpdateIniFields=*update-inifields-section*
AddReg=*add-registry-section*
DelReg=*del-registry-section*
Ini2Reg=*ini-to-registry-section*
UpdateCfgSys=*update-config-section*
UpdateAutoBat=*update-autoexec-section*

Identifies the additional sections in the INF that contain installation information for the component.

Not all the types of items shown in the syntax above are needed or required in an **Install** section. If an item type is used, it must specify the name of a section in the INF. (An exception to this is the CopyFiles item, which may use the "@" character along with a filename to copy a single file without specifying a section name.) The section name must consist of printable characters.

Only one of each type of item can be used in any one **Install** section. More than one section name can be listed in an item, but each additional name must be preceded with a comma.

install-section-name
Naming the install section **[DefaultInstall]** will cause the install section to be executed when the "Install" verb is selected when the INF is right-clicked. It is also the section that is executed when selecting an INF as the setup option using the Cabpack wizard.

This example shows a typical **Install** section. It contains CopyFiles and AddReg items that identify the sections containing information about which files to install.

Example:
```
[MyApplication]
```

```
Copyfiles=MyAppWinFiles, MyAppSysFiles, @SRSutil.exe
AddReg=MyAppRegEntries
```

Note that in the above example, by renaming the [MyApplication] section to **[DefaultInstall]**, this Install section would be executed when the "Install" verb is selected when right-clicking the INF.

The CopyFiles item provides a special notation which will allow a single file to be copied directly from the copy line. An individual file can be copied by prefixing the file name with an @ symbol. The destination for any file copied using this notation will be the **DefaultDestDir** as defined in the DestinationDirs section.

The following example shows how to copy individual files:

```
CopyFiles=FileSection1,@myfile.txt,@anotherfile.txt,LastSectionName
```

Required Setup Version

RequiredEngine

```
RequiredEngine=<Value>,%ErrorString%
```

There are two values that may be assigned to a RequiredEngine parameter:

<Value>	Description	Usage
SETUPAPI	32-bit version of setup engine (advpack.dll)	Win95 is forced into 32-bit install, or displays error dialog. Standard WinNT setup engine.
SETUPX	16-bit version of setup engine	Cannot be used with WinNT, forces Win95 into 16-bit install.

For more information about SETUPAPI, check out setupapi.h.

SmartReboot

Overview

SmartReboot lets you choose whether or not to ask a user to reboot his system. The install engine determines if a reboot is needed for you.

Syntax

```
[InstallSection]
SmartReboot=&ltparameter>
```

Parameter

Parameter	Meaning
N	Never reboot
AS	Always reboot without asking
IS	Reboot without prompting if needed
A	Always ask user to reboot
I	Ask user to reboot if needed

Note

SmartReboot is ignored unless the INF is excuted via LaunchINFSection.

INF Cleanup

Cleanup

```
[DefaultInstall]
Cleanup=1
```

Function

This parameter located in an install section is useful for when you want to delete your INF when you uninstall, but you need your INF to finished uninstalling.

In the case where you need a reboot to finish deleting or unregistering a file because it is in use, then your INF may still be needed to do this. This function causes your INF to automatically be deleted when it has finished the uninstall.

Check Administrator Rights

CheckAdminRights

```
[DefaultInstall]
CheckAdminRights=<flag>
```

Flags

Flag	Function
0	Don't Check Rights -- Anyone Install
1	Verify that the User has Rights

ComponentName Parameter

ComponentName

ComponentName=*"My Component"*

This parameter is used by LaunchINFSectionEx to perform the rollback feature. The name *"My Component"* does not need to be localizable because the user will never see it, it is only used by LaunchINFSectionEx.

PreRollBack and GenInstall

PreRollBack

When using the rollback function in the Advanced INF Installer, this general install section can be called before the rollback is fired.

This allows the use of a seperate install section to prepare the users machine before the rollback happens.

Alternate Backup Path

BackupPath

```
[DefaultInstall]
BackupPath=%11%\MyText
```

If you would rather backup your files for rollback into a different directory than the directory used by default, you can specify a path with this parameter.

Specify Per User Stub

PerUserInstall

For each ActiveSetup enabled component, this option defines the component's states, version, locale and per-user stubpath under HKLM\Software\microsof\Active Setup\Installed Components\GUID registry key. When the user first time logon after installing the component, the commands pointed by StubPath may be executed if this version of component per-user stub has not been run for the user.

Syntax:

```
[InstallSection]
PerUserInstall=PerUserInstall

[PerUserInstall]
DisplayName=%WebInteg%
Version=5.0.0311.0
IsInstalled=1
ComponentID=IE4Shell_WIN
GUID={89820200-ECBD-11cf-8B85-00AA005B4395}
Locale=en
StubPath=rundll32.exe advpack.dll,LaunchINFSectionEx %11%\myinst.inf,
UserStub,,36

[Strings]
```

WebInteg = "MyComp Update"

Note:

```
The StubPath data can be any command line launched by CreateProcess
().
```

DestinationDirs Section

DestinationDirs

[DestinationDirs]
file-list-section=ldid(, subdir)
DefaultDestDir=*ldid(, subdir)*

The DestinationDirs section defines the destination directories for the operations specified in *file-list-section*, which is either a CopyFiles, RenFiles, or DelFiles section. Optionally, a default destination directory may be specified for any CopyFiles, RenFiles, or DelFiles section in the INF file that are not explicitly named in the DestinationDirs section.

file-list-section
Name of a CopyFiles, RenFiles, or DelFiles section. This name must be referred to in a Copyfiles, RenFiles, or DelFiles item in an Install section.

ldid
A logical disk identifier (LDID). Can be one of these values:

Value	Meaning
00	Null LDID. This LDID can be used to create a new LDID
01	Source Drive:\pathname
10	Machine directory (Maps to the Windows directory on a Server-Based Setup.)
11	System directory
12	IOSubsys directory
13	Command directory
17	INF Directory
18	Help directory
20	Fonts

21	Viewers
22	VMM32
23	Color directory
24	Root of drive containing the Windows directory
25	Windows directory
26	Guarenteed boot device for Windows (Winboot)
28	Host Winboot
30	Root directory of the boot drive
31	Root directory for Host drive of a virtual boot drive

subdir
Name of the directory, within the directory named by ldid, to be the destination directory.

The optional **DefaultDestDir** item provides a default destination for any CopyFiles items that use the direct copy (@filename) notation or any CopyFiles, RenFiles, or DelFiles sections not specified in the DestinationDirs section. If a DefaultDestDir is not used in a DestinationDirs section, then the default directory is set to LDID_WIN.

This example sets the destination directory for the MoveMiniPort section to WINDOWS\IOSUBSYS and sets the default directory for other sections to be the BIN directory on the boot drive:

```
[DestinationDirs]
MoveMiniPort=12 ; Destination for MoveMiniPort Section is windows
\iosubsys
DefaultDestDirs=30,bin ; Direct copies go to Boot:\bin
```

CustomDestination Section

CustomDestination

This is where you can either ask the user for a custom destination for your copy files, or you can obtain the destination from a reg key.

Reg Key Destination

```
[Version]
Signature="$Chicago$"
AdvancedINF=2.5,"You need a newer version of Advpack.dll"

[SourceDiskNames]
10="Title",,0

[SourceDisksFiles]
testfile.txt=10

[DefaultInstall]
CustomDestination=CustomDestinationSection
CopyFiles=CopyFilesSection

[CustomDestinationSection]
49001=DestA,1  ;<1=prompt user, 5=no prompt>

[DestA]
HKLM,"SOFTWARE\Test","InstLoc","Location Not Found in Reg. Use default?","C:\default\folder"

[DestinationDirs]
CopyFilesSection=49001

[CopyFilesSection]
testfile.txt
```

OCX Registration/UnRegistration

RegisterOCXs

IExpress's AdvPack INF file extensions can be used to properly register OCX's after installation. You can also unregister your OCXs before they are deleted upon uninstall.

INF File Sections

```
[Version]
Signature="$Chicago$"
AdvancedINF=2.5

[DefaultInstall]
RegisterOCXs=RegisterOCXSection

[Uninstall]
UnRegisterOCXs=RegisterOCXSection

[RegisterOCXSection]
%&ltLDID>%\&ltsubdir>\&ltOCX file name>,<flag,<parameter>>
```

Flag

Flag	Description
I	Call DllRegisterServer and DllInstall
N	Do not call DllRegisterServer, only DllInstall

Parameter

This becomes available when a flag is used. You may pass a string as a parameter for DllInstall.

Add Registry Section

AddReg

```
[DefaultInstall]
AddReg=AddRegSection

[AddRegSection]
reg-root-string,(subkey),(value-name),(flag),(value)
2nd-reg-root-string,(subkey),(value-name),(flag),(value)
```

Adds subkeys or value names to the registry, optionally setting the value. The *AddRegSection* name must appear in an AddReg-type item in an Install section such as *DefaultInstall*.

reg-root-string
Registry root name. Can be one of these values:

- **HKCR** = HKEY_CLASSES_ROOT
- **HKCU** = HKEY_CURRENT_USER
- **HKLM** = HKEY_LOCAL_MACHINE
- **HKU** = HKEY_USERS.
- **HKR** Means relative from the Key passed into GenInstallEx

subkey
Optional. Identifies the subkey to set. Has the form *key1\key2\key3*.... This parameter can be expressed as a replaceable string. For example, you could use %Subkey1% where the string to replace %Subkey1% is defined in the Strings section of the INF file.

value-name
Optional. Identifies the value name for the *subkey*. For string type, if *the value-name* parameter is left empty, the value of the subkey specified in the *subkey* parameter is set to a NULL string. Note that the *value-name* parameter can be expressed as a replaceable string. For example, you could use %Valname1% where the string to replace %Valname1% is defined in the Strings section of the INF file.

flag
Optional. Determines both the value type and whether the registry key is replaced if it already exists.

Value	Meaning
0	(Default) Value is an ANSI string. Replace key if it exists.
1	Value is a hexadecimal number. Replace key if it exists.
2	Value is an ANSI string. Do not replace key if it exists.
3	Value is a hexadecimal number. Do not replace key if it exists.

value
Optional. Value to set. Can be either an ANSI string or a number in hexadecimal notation and Intel format. Any item containing a binary value can be extended beyond the 128-byte line maximum by using a backslash (\) character. A string key of the form %*strkey*% can also be given. The *strkey* must be defined in the Strings section of the INF file. To use a % character in the line, use %%.

At least two fields are required, however one can be null thus at least one comma is required when using this form.

The two items in the example AddReg-type section below add two value names to the registry. Note that %25% will be expanded to the machine's Windows directory.

```
[MyAppRegEntries]
HKLM,Software\MyApp,ProgramName,,"My Application"
HKLM,Software\MyApp,"Program Location",,"%25%\MyApp.exe"
```

Note: Using an INF there is not a way to add dwords in Win95, but there is a way to trick the system to believe that you did:

Set a binary key to the value of **01,00,00,00**. This will reflect the same as a dword of one.

Delete Registry Section

DelReg

[del-registry-section]
reg-root-string, subkey, (value-name)
2nd-reg-root-string, subkey, (value-name)

Deletes a subkey or value name from the registry. The *del-registry-section* name must appear in an **DelReg** item in an Install section.

reg-root-string
Registry root name. Can be one of these values:

- **HKCR** = HKEY_CLASSES_ROOT
- **HKCU** = HKEY_CURRENT_USER
- **HKLM** = HKEY_LOCAL_MACHINE
- **HKU** = HKEY_USERS.
- **HKR** Means relative from the Key passed into GenInstallEx

subkey
Identifies the subkey to delete. Has the form *key1\key2\key3*... This parameter can be expressed as a replaceable string. For example, you could use %Subkey1% where the string to replace %Subkey1% is defined in the **Strings** section of the INF file.

value-name
Optional. Identifies the value name for the *subkey*. Note that the *value-name* parameter can be expressed as a replaceable string. For example, you could use %Valname1% where the string to replace % Valname1% is defined in the **Strings** section of the INF file.

This type of section can contain any number of items. Each item deletes one subkey or value name from the registry.

Delete Directories Section

DelDirs

This allows you to delete folders on a user's system. It will only work if the folder is empty.

Syntax

```
[DefaultInstall]
DelDirs=DelDirsSection

[DelDirsSection]
%11%/FolderX
```

This would delete empty folder "FolderX" in the systems folder.

Run PreSetup Applications

RunPreSetupCommands

You can launch any number of commands before an INF install section is executed. Standard LDIDs can be used in pre-install command sections, but Custom LDIDs can not.

Syntax:

```
[InstallSection]
RunPreSetupCommands=<CmdSection1>[:flag][,CmdSection2[:flag]

[CmdSection1]
myapp1.exe
myapp2.exe

[CmdSection2]
myapp1.exe
myapp2.exe
```

Flag:

Quiet 1
NoWait 2

```
Default flag is not quiet and wait for command return.
```

Note:

The RunPreSetupCommands section is executed after the begin prompt.

Run PostSetup Applications

RunPostSetupCommands

You can launch any number of commands after an INF section is executed. Both Standard and Custom LDIDs can be used in post-install command sections.

Syntax:

```
[InstallSection]
RunPostSetupCommands=Section1[:flag][,<Section2[:flag]>]

[Section1]
myapp1.exe
myapp2.exe

[Section2]
myapp3.exe
myapp4.exe
```

Flag

Flag	Meaning
1	Quiet Mode
2	No Delay
4	Delay Command

The delayed commands will be add to RunOnceEx or RunOnce branch based on the same logic as delayed register OCXs method. We check the *EXPLORER.EXE* version or presence of *IERUNONCE.DLL* to determine where to add the values. If added to RunOnceEx, starting subkey is 990.

Keep in mind that on Win95 systems without *IERUNONCE.DLL*, all of the delay commands will be added to RunOnce branch. Due RegEnum limits the order can not be guaranteed, so we recommend the

user use this feature in the IE 4.0 context.

Note:

The RunPostSetupCommands section is executed immediately before the end prompt.

User Prompting and Dialogs

BeginPrompt and EndPrompt

With user prompting, you can pop up a confirmation box before executing an INF. You can also inform users that an install/uninstall was completed with a dialog.

Syntax

```
[InstallSection]
BeginPrompt=BeginPromptSection
EndPrompt=EndPromptSection

[BeginPromptSection]
Prompt="<Prompt>"
ButtonType=<ButtonType>
Title="<Title>"

[EndPromptSection]
Prompt="<Prompt>"
```

Parameters

Parameter	Description
<Prompt>	String that is displayed in dialog box
<ButtonType>	**YESNO** = Buttons are Yes/No **OKCANC** = Buttons are OK/Cancel
<Title>	Title displayed on ALL dialog boxes ONLY if advpack is called via LaunchINFSection

Notes

Since the <Title> parameter in the BeginPrompt section is used for error dialog titles and in the

EndPrompt dialog box, you must add a BeginPrompt section to your INF with only a <Title> parameter so that your dialogs are displayed properly.

Both prompt sections are only executed if the INF is launched via LaunchINFSection.

Copy Files Section

CopyFiles

[CopyFiles-section-name]
destination-file1-name(, source-file1-name)(, temporary-file1-name)(,flag)
destination-file2-name(, source-file2-name)(, temporary-file2-name)(,flag)

A list of the names of files to be copied from a source disk to a destination directory. The **source disk** and **destination directory** associated with each file are specified in other sections of the INF file. The file-list-section name must appear in the CopyFiles item of an Install section.

Note that you can specify the copying of a single file in the CopyFiles item of the Install section itself, without building a CopyFiles section. To do this, use the special character "@" to force a single file copy. An example of using the "@" character in a CopyFiles-type item is in the Install section reference topic. Copying a single file in this way is somewhat limited because the source and destination filenames must be the same in this case and you cannot use a temporary file.

destination-file-name
Name of the destination file. If no source filename is given, this is also the name of the source file.

source-file-name
Name of the source file. If the source and destination filenames for the file copy operation are the same, this is not required.

temporary-file-name
Name of a temporary file for the file copy operation. The installer copies the source file but gives it the temporary file name. The next time the operating system starts, it renames the temporary file to the destination file name. This is useful for copying files to a destination which is currently open or in use by Windows.

If the file is not in use by Windows, use flag **8** to force it to use the temporary name. This will only work if the file already exists in in the target folder. To get around this, first copy the file into the folder, then use flag 8 to copy it again.

flag
Optional parameter used to perform special actions during the installation process. Multiple flags can be used by adding the values to create the combined flag. The following valid flags can be used:

Value	Meaning
1	On **CopyFiles**: Warn if user tries to skip file.
1	On **DelFiles**: If file is in use, queue up delayed delete in wininit.ini. Otherwise an in-use file will not be deleted.
2	Setup Critical: don't allow user to skip file.
4	Ignore version check and always copy file. This will overwrite a newer file.
8	Force Rename (trick engine into thinking that file is in use). **Note:** Only happens if file already exists on target.
16	If file already exists on target, don't copy.
32	Suppress version conflict dialog and don't overwrite newer files.

This example copies three files:

```
[CopyTheseFilesSec]
file11 ; copies file11
file22,, file23,8 ; copies file22, temporarily naming it file23
file31, file32 ; copies file32 to file31
```

All the source filenames used in this example must be defined in a SourceDisksFiles section and the logical disk numbers that appear in the SourceDisksFiles section must have been defined in a SourceDisksNames section. As an alternative, you can use a LAYOUT.INF file to supply this information.

You also must define the destination directory for each CopyFiles section using the DestinationDirs section. More than one destination directory requires more than one CopyFiles section.

Rename Files Section

RenFiles

[Rename-files-section-name]
new-file-name, old-file-name

Lists the names of files to be renamed. The name of the section must appear in a **Renfiles** item in an Install section of the INF file.

new-file-name
New name of the file.

old-file-name
Old name of the file.

This example renames the files file42 to file41, file52 to file51, and file62 to file61 in the system folder:

```
[DefaultInstall]
RenFiles=RenameOldFilesSec

[RenameOldFilesSec]
file41, file42
file51, file52
file61, file62

[DestinationDirs]
RenameOldFilesSec=11
```

The folder where all the old filenames exist must be defined in a DestinationDirs section as shown in the above example.

Delete Files Section

DelFiles

[File-list-section]
file-name(,,,flag)

A **DelFiles** section lists the names of files to be deleted. The *file-list-section* name must appear in the **Delfiles** item of an Install section.

file-name
Identifies a file to be deleted.

flag
Optional parameter used to force Windows 95 to delete the file named in the item if it is in use during the installation process. Set the *flag* parameter value to 1 to cause Windows 95 to queue the file deletion operation until the system has restarted. If a file marked with the *flag*=1 parameter setting cannot be deleted because it is in use, a system restart will occur after the device installation is complete.

If you do not use the *flag* parameter value equal to 1 along with a *file-name* parameter, then if that file is in use when the **DelFiles** section is executed that file will not be deleted from the system.

This example deletes three files in the system folder:

```
[DefaultInstall]
DelFiles=DeleteOldFilesSec

[DeleteOldFilesSec]
file1
file2
file3,,,1

[DestinationDirs]
DeleteOldFilesSec=11
```

Update INI File Section

UpdateInis

[Update-ini-section-name]
ini-file, ini-section, (old-ini-entry), (new-ini-entry), (flags)

Replaces, deletes, or adds complete entries in the given INI file. The section name, *update-ini-section-name*, must appear in the **UpdateInis** item in an Install section of the INF file.

ini-file
Name of the INI file containing the entry to change. For more information about specifying the INI filename, see the comments below.

ini-section
Name of the section containing the entry to change.

old-ini-entry
Optional. Usually in the form *Key=Value*.

new-ini-entry
Optional. Usually in the form *Key=Value*. Either the key or value may specify replaceable strings. For example, either the key or value specified in the *new-ini-entry* parameter may be %String1%, where the string that replaces %String1% is defined in the Strings section of the INF file.

flags
Optional action flags. Can be one of these values:

Value	Meaning
0	(Default) If *old-ini-entry* key is present in an INI file entry, that entry is replaced with *new-ini-entry*. Note that only the keys of the *old-ini-entry* parameter and the INF file entry must match, the value of each entry is ignored. To add *new-ini-entry* to the INI file unconditionally, set *old-ini-entry* to NULL. To delete *old-ini-entry* from the INI file unconditionally, set *new-ini-entry* to NULL.

1	If both key and value of *old-ini-entry* exist in an INI file entry, that entry is replaced with *new-ini-entry*. Note that the *old-ini-entry* parameter and the INF file entry must match on both key and value for the replacement to be made (this is in contrast to using an action flag value of 0, where only the keys must match for the replacement to be made).
2	If the key in the *old-ini-entry* parameter does not exist in the INI file, then no operation is performed on the INI file. If the key in the *old-ini-entry* parameter exists in an INI file entry and the key in the *new-ini-entry* parameter exists in an INI file entry, then the INI file entry that matches the key in the *new-ini-entry* parameter is deleted and the INI file entry that matches the *old-ini-entry* parameter is operated on in the following way: the key of the INI file entry is replaced with the key in the *new-ini-entry* parameter. If the key in the *old-ini-entry* parameter exists in an INI file entry and the key in the *new-ini-entry* parameter does not exist in an INI file entry, then an entry is added to the INI file made up of the key in the *new-ini-entry* parameter and the old value. Note that the match of the *old-ini-entry* parameter and an INI file entry is based on key alone, not key and value.
3	Same as flag parameter value of 2 described above, except match of the *old-ini-entry* parameter and an entry in the INF file is based on matching both key and value, not just the key.

The wild card character (*) can be used in specifying the key and value and they will be interpreted correctly.

The *ini-file* name can be a string or a strings key. A strings key has the form %*strkey*% where *strkey* is defined in the Strings section in the INF file. In either case, the name must be a valid filename.

The name should include the name of the directory containing the file, but the directory name should be given as a logical directory identifier (LDID) rather than an actual name. The installer replaces an LDID with an actual name during installation.

An LDID has the form %*ldid*% where *ldid* is one of the predefined identifiers or an identifier defined in the DestinationDirs section. Note that when the constants LDID_BOOT and LDID_BOOTHOST are replaced, the backslash is included in the path. For example LDID_BOOT may be replaced with C:\. However, in your INF file you can either use the backslash character or not. For example, either "%30% boot.ini" and "%30%\boot.ini" can be used to reference BOOT.INI in the root of the boot drive.

The following examples illustrate individual items in an Update INI File section of an INF file:

```
%11%\sample.ini, Section1,, Value1=2 ; adds new entry
%11%\sample.ini, Section2, Value3=*, ; deletes old entry
%11%\sample.ini, Section4, Value5=1, Value5=4 ; replaces old entry
```

The following set of items in an Update INI File-type section of an INF file work together to operate on the Boot section of SYSTEM.INI. The conditionality built into the *flags* parameter of the INF file items is used to add the entry "comm.drv=comm.drv" to the Boot section unless the entries "comm. drv=*vcoscomm.drv" or "comm.drv=*r0dmdcom.drv" exist in the Boot section, in which case the existing entry is preserved and the entry "comm.drv=comm.drv" is not added to the INI file. In other words, after the four INF file entries shown below are executed, there will be one "comm.drv=" entry in the Boot section of the INI file: "comm.drv=*vcoscomm.drv" or "comm.drv=*r0dmdcom.drv" or "comm.drv=comm.drv."

```
system.ini, boot, "comm.drv=*vcoscomm.drv","~CommDrvTemp~=*",3
system.ini, boot, "comm.drv=*r0dmdcom.drv","~CommDrvTemp~=*",3
system.ini, boot,,"comm.drv=comm.drv"
system.ini, boot, "~CommDrvTemp~=*","comm.drv=*",3
```

Update INI Fields Section

UpdateIniFields

[update-inifields-section-name]
ini-file, ini-section, profile-name,(old-field), (new-field),(flags)

Replaces, adds, and deletes fields in the value of a given INI entry. Unlike the Update INI File section type, this type of section replaces, adds, or deletes portions of a value in an INI file entry rather than the whole value. The section name, *update-inifields-section-name*, must appear in the **UpdateIniFields** item in an Install section of the INF file.

ini-file
Name of the INI file containing the entry to change. For more information about specifying the INI filename, see the topic that describes the Update INI File section type.

ini-section
Name of the INI file section containing the entry to change.

profile-name
Name of the entry to change.

old-field
Field value to delete.

new-field
Field value to add, if not already there.

flags
Specifies whether to treat the *old-field* and *new-field* parameters as if they have a wild card character or not and/or what separator character to use when appending a new field to an INI file entry. Can be any of these values:

Value	Meaning
0	(Default) Treat "*" character literally when matching fields, and not as a wild card character. Use blank (" ") as a separator when adding a new field to an entry.

1	Treat "*" character as a wild card character when matching fields. Use blank (" ") as a separator when adding a new field to an entry.
2	Treat "*" character literally when matching fields, and not as a wild card character. Use comma (",") as a separator when adding a new field to an entry.
3	Treat "*" character as a wild card character when matching fields. Use comma (",") as a separator when adding a new field to an entry.

Any comments in the INI file line are removed as they might not be applicable after changes. When looking for fields in the line in the INI file, spaces, tabs and commas are used as field delimiters. However, a space is used as the separator when the new field is appended to the line.

Ini File to Registry Section

Ini2Reg

[ini-to-registry-section]
ini-file, ini-section, (ini-key), reg-root-string, subkey(,flags)

Moves lines or sections from an INI file to the registry, creating or replacing a registry entry under the given key in the registry. The section name *ini-to-registry-section* must appear in an **Ini2Reg** item in an Install section of the INF file.

ini-file
Name of the INI file containing the key to copy. For more information about specifying the INI filename, see the comments in the Reference topic about the Update Ini File Section.

ini-section
Name of the section in the INI file containing the key to copy.

ini-key
Name of the key in the INI file to copy to the registry. If *ini-key* is empty the whole section is transferred to the specified registry key.

reg-root-string
Registry root name. Can be one of these values:

- **HKCR** = HKEY_CLASSES_ROOT
- **HKCU** = HKEY_CURRENT_USER
- **HKLM** = HKEY_LOCAL_MACHINE
- **HKU** = HKEY_USERS.
- **HKR** Means relative from the Key passed into GenInstallEx

subkey
Identifies the subkey to receive the value. Has the form *key1\key2\key3*...

flags
Indicates whether to delete the INI key after transfer to the registry and whether to overwrite the value in the registry if the registry key already exists. Can be one of these values:

Value	Meaning
0	(Default) Do not delete the INI entry from the INI file after moving the information in the entry to the registry. If the registry subkey already exists, do not replace its current value.
1	Delete the INI entry from the INI file after moving the information in the entry to the registry. If the registry subkey already exists, do not replace its current value
2	Do not delete the INI entry from the INI file after moving the information in the entry to the registry. If the registry subkey already exists, replace its current value with the value from the INI file entry.
3	Delete the INI entry from the INI file after moving the information in the entry to the registry. If the registry subkey already exists, replace its current value with the value from the INI file entry.

For example, suppose the following entry exists in the WIN.INI file:

```
[Windows]
CursorBlinkRate=15
```

If a CursorBlinkRate subkey does not exist under the Control Panel\Desktop, then the following item in an **Ini File to Registry** section creates the subkey, sets the value of the subkey to 15, and leaves the original line in WIN.INI unchanged:

```
win.ini,Windows,CursorBlinkRate,HKCU,"Control Panel\Desktop"
```

If the subkey already exists, the INF file item sets the value of the subkey to 15, and leaves the original line in WIN.INI unchanged.

Update Config.sys Section

UpdateCfgSys

[update-config-section]
Buffers=*legal-dos-buffer-value*
DelKey=*key*
DevAddDev=*driver-name,configkeyword(,flag)(,param-string)*
DevDelete=*device-driver-name*
DevRename=*current-dev-name,new-dev-name*
Files=*legal-dos-files-value*
PrefixPath=*ldid(,ldid)*
RemKey=*key*
Stacks=*dos-stacks-values*

Provides commands to add, delete, or rename commands in the CONFIG.SYS file. The section name, *update-config-section-name*, must appear in the **UpdateConfigSys** item in an Install section of the INF file.

Not all item types shown in the syntax above are needed or required. An Update Config.sys section may contain as many **DevRename**, **DevDelete**, **DevAddDev**, **DelKey**, and **RemKey** items as needed, but the **Buffers**, **Files**, and **Stacks** items may only be used once in a section. When processing an Update Config.sys section, the Installer processes all **DevRenames** items first, all **DevDelete** items second, and all **DevAddDev** items last. The syntax and meaning of each of the types of items that can be used in an Update Config.sys section are given later in this topic.

Buffers Item

Buffers=*legal-dos-buffer-value*
Sets the number of file buffers. As it does with the **Stacks** item, the Installer compares the existing value with the proposed value and always sets the file buffers to the larger of the two values.

DelKey Item

DelKey=*key*
Causes the CONFIG.SYS command with the specified key to be remarked out in the CONFIG.SYS file. For example, the INF file item:

```
DelKey=Break
```

would cause a Break=on command to be remarked out in the CONFIG.SYS file.

The **DelKey** item has the same effect as the **RemKey** item. There can be multiple **DelKey** and/or **RemKey** items in a section of the INF file.

key
The key of the CONFIG.SYS command to be remarked out.

DevAddDev Item

DevAddDev=*driver-name,configkeyword(,flag)(,param-string)*
Adds a **device** or **install** command to the CONFIG.SYS file.

driver-name
Name of the driver or executable file to add. The installer validates the filename extension, ensuring that it is SYS or EXE.

configkeyword
Command name. Can be device or install.

flag
Optional placement flag. If 0, the command is placed at the bottom of the file. If 1, it is placed at the top. If flag is not given, 0 is used by default.

param-string
Optional command parameters. Must be valid for the given device driver or executable file.

DevDelete Item

DevDelete=*device-driver-name*
Deletes any line containing the specified filename from the CONFIG.SYS file.

device-driver-name
Name of a file or device driver. The Installer searches the CONFIG.SYS file for the name and deletes any line containing it. Because MS-DOS does not permit implicit filename extensions in CONFIG.SYS, each *device-driver-name* must explicitly specify the filename extension.

This example **DevDelete** item in an Update Config.sys section deletes lines 1 and 3 but not line 2 of the example CONFIG.SYS file:

```
DevDelete=Foo.sys
;; lines in CONFIG.SYS
Device=Foo.sys ;; line #1
Install=foo.exe ;; line #2
Device=Foo.sys /d:b800 /I:3 ;; line #3
```

DevRename Item

DevRename=*current-dev-name,new-dev-name*
Renames a device driver in the CONFIG.SYS file.

current-dev-name
Name of the device driver or executable file to rename. The installer looks for the name on the right side of a **device** or **install** command in the CONFIG.SYS.

new-dev-name
New name for driver or executable file.

Files Item

Files=*legal-dos-files-value*
Sets the maximum number of open files in the CONFIG.SYS file. As it does with the **Stacks** item, the Installer compares the existing value with the proposed value and always sets the maximum number of open files to the larger of the two values.

legal-dos-files-value
A legal MS-DOS files value.

PrefixPath Item

PrefixPath=*ldid(,ldid)...*
Appends the path associated with the given LDID to the path command.

ldid
Can be any of the predefined LDID values or a new value defined in the INF. For a definition of all the predefined LDID values, see the "Reference" topic about the DestinationDirs section.

RemKey Item

RemKey=*key*
Causes the CONFIG.SYS command with the specified key to be remarked out in the CONFIG.SYS file.

For example, the INF file item:

```
RemKey=Break
```

would cause a Break=on command to be remarked out in the CONFIG.SYS file.

The **RemKey** item has the same effect as the **DelKey** item. There can be multiple **RemKey** and/or **DelKey** items in a section of the INF file.

key
The key of the CONFIG.SYS command to be remarked out.

Stacks Item

Stacks=*dos-stacks-values*
Sets the number and size of stacks in the CONFIG.SYS file. The Installer compares the existing value with the proposed value and always sets the stacks to the larger of the two values. For example, if CONFIG.SYS contains stacks=9,218 and an INF contains stacks=5,256, the installer sets to new value to stacks=9,256.

Update Autoexec.bat Section

UpdateAutoBat

[update-autoexec-section]
CmdAdd=*command-name(,command-parameters*)
CmdDelete=*command-name*
PrefixPath=*ldid(,ldid)*
RemOldPath=*ldid*(,*ldid*)
TmpDir=*ldid*(,*subdir*)
UnSet=*env-var-name*

Provides commands to manipulate lines in the AUTOEXEC.BAT file. The section name, *update-autoexec-section-name*, must appear in the **UpdateAutoBat** item in an Install section of the INF file.

Not all item types shown in the syntax above are needed or required in an Update Autoexec.bat section. The section can contain as many **CmdAdd**, **CmdDelete** and **UnSet** items as needed, but only one **PrefixPath**, **RemOldPath** and **TmpDir** items can be used in an INF file. The syntax and meaning of each of the item types is described later in this topic.
The installer processes all **CmdDelete** items before any **CmdAdd** items.

CmdAddItem

CmdAdd=command-name(,"command-parameters")
Adds the given command and optional command parameters to the AUTOEXEC.BAT file, at the end of the file.

command-name
Name of an executable file, with or without an extension. If the filename is also defined in the SourceDisksFiles and DestinationDirs sections of the INF file, the installer adds the appropriate path to the filename before writing it to the AUTOEXEC.BAT file.

command-parameters
A string enclosed in double quotation marks or a replaceable string like %String1% or %Myparam%, where the strings that replace %String1% and %Myparam% are defined in the Strings section of the INF file. The installer appends the string to the *command-name* before appending the line to the end of the AUTOEXEC.BAT file. The format of this line is dependent on the command line requirements of the given executable file.

CmdDelete Item

CmdDelete=*command-name*
Deletes any lines from AUTOEXEC.BAT that include the given command name. The installer searches for and deletes any occurrence of the given name that has a filename extension of EXE, COM, and BAT.

command-name
Name of an executable file without an extension.

PrefixPath Item

PrefixPath=*ldid(,ldid)...*
Appends the path associated with the given LDID to the path command.

ldid
Can be any of the predefined LDID values or a new value defined in the INF. For a definition of all the predefined LDID values, see the "Reference" topic about the DestinationDirs section.

RemOldPath Item

RemOldPath=*ldid*(,*ldid*)
Removes the path associated with the given LDID from the path command. For example, if the user installs the new version of Windows into c:\newwin and has an old copy of Windows in c:\windows, the following INF file item removes c:\windows from the path environmental variable:

```
RemOldPath=10
```

ldid
Can be any of the predefined LDID values or a new value defined in the INF. For a definition of all the predefined LDID values, see the "Reference" topic about the DestinationDirs section.

TmpDir Item

TmpDir=*ldid[,subdir]*
Creates a temporary directory within the directory given by the LDID, if it does not already exist.

ldid
Can be any of the predefined LDID values or a new value defined in the INF. For a definition of all the predefined LDID values, see the "Reference" topic about the DestinationDirs section.

subdir
A path name. If ldid\subdir does not already exist, it is created.

UnSet Item

UnSet=*env-var-name*
Removes any **set** command from the AUTOEXEC.BAT file that includes the given environment variable name.

Backup Registry Section

BackupReg

```
[DefaultInstall]
BackupReg=BackupRegSection

[BackupRegSection]
HKLM,"SOFTWARE\Microsoft\Active Setup"
```

This is a normal registry section similar to AddReg or DelReg. BackupReg is useful for saving the state of a registry key before perfoming an install/rollback.

General Install Section

GenInstall

This section is basically another install section, which normally is not the default install section. So in the case of PreRollBack, this new install section is called before rollback.

Strings Section

Strings

[Strings]
strings-key=value

Defines one or more strings keys. A strings key is a name that represents a string of printable characters. Although the Strings section is generally the last section in the INF files, a string key defined in this section may be used anywhere in the INF file that the corresponding string would be used. The Installer expands the string key to the given string and uses it for further processing. Using a strings key requires that it be enclosed in percent signs (%).

strings-key
A unique name consisting of letters and digits.

value
A string consisting of letters, digits, or other printable characters. It should be enclosed in double-quotation marks if the corresponding strings key is used in a type of item that requires double quotation marks.

The Strings section makes translation of strings for international markets easier by placing all strings that can be displayed at the user interface when the INF file is used in a single section of the INF file. Strings keys should be used whenever possible.

The following example shows the strings section for a sample INF file.

```
[Strings]
String0="My Application"
String1="My Application Readme File"
String2="CX2590 SCSI Adapter"
```

LayoutFile

```
/*++

Copyright (c) 1995-1997 Microsoft Corporation

Module Name:

   setupapi.h

Abstract:

   Public header file for Windows NT Setup services Dll.

Author:

   Ted Miller (tedm) 11-Jan-1995

Revision History:

--*/

#ifndef _INC_SETUPAPI
#define _INC_SETUPAPI

//
// Define API decoration for direct importing of DLL references.
//
#if !defined(_SETUPAPI_)
#define WINSETUPAPI DECLSPEC_IMPORT
#else
#define WINSETUPAPI
#endif

#include <pshpack1.h>   // Assume byte packing throughout

#ifdef __cplusplus
extern "C" {
#endif

#ifndef __LPGUID_DEFINED__
#define __LPGUID_DEFINED__
typedef GUID *LPGUID;
#endif

//
// Include commctrl.h for our use of HIMAGELIST and wizard support.
//
#include <commctrl.h>


//
// Define maximum string length constants as specified by
// Windows 95.
//
#define LINE_LEN                256 // Win95-compatible maximum for displayable
                                    // strings coming from a device INF.
```

```
#define MAX_INF_STRING_LENGTH      4096 // Actual maximum size of an INF string (including
                                        // string substitutions).
#define MAX_TITLE_LEN              60
#define MAX_INSTRUCTION_LEN        256
#define MAX_LABEL_LEN              30
#define MAX_SERVICE_NAME_LEN       256

//
// Define type for reference to loaded inf file
//
typedef PVOID HINF;

//
// Inf context structure. Applications must not interpret or
// overwrite values in these structures.
//
typedef struct _INFCONTEXT {
    PVOID Inf;
    PVOID CurrentInf;
    UINT Section;
    UINT Line;
} INFCONTEXT, *PINFCONTEXT;

//
// Inf file information structure.
//
typedef struct _SP_INF_INFORMATION {
    DWORD InfStyle;
    DWORD InfCount;
    BYTE VersionData[ANYSIZE_ARRAY];
} SP_INF_INFORMATION, *PSP_INF_INFORMATION;

//
// SP_INF_INFORMATION.InfStyle values
//
#define INF_STYLE_NONE  0       // unrecognized or non-existent
#define INF_STYLE_OLDNT 1       // winnt 3.x
#define INF_STYLE_WIN4  2       // Win95

//
// Target directory specs.
//
#define DIRID_ABSOLUTE         -1              // real 32-bit -1
#define DIRID_ABSOLUTE_16BIT    0xffff         // 16-bit -1 for compat w/setupx
#define DIRID_NULL              0
#define DIRID_SRCPATH           1
#define DIRID_WINDOWS           10
#define DIRID_SYSTEM            11             // system32
#define DIRID_DRIVERS           12
#define DIRID_IOSUBSYS          DIRID_DRIVERS
#define DIRID_INF               17
#define DIRID_HELP              18
#define DIRID_FONTS             20
#define DIRID_VIEWERS           21
#define DIRID_COLOR             23
#define DIRID_APPS              24
```

```
#define DIRID_SHARED           25
#define DIRID_BOOT             30

#define DIRID_SYSTEM16         50
#define DIRID_SPOOL            51
#define DIRID_SPOOLDRIVERS     52
#define DIRID_USERPROFILE      53
#define DIRID_LOADER           54

#define DIRID_DEFAULT          DIRID_SYSTEM

//
// First user-definable dirid. See SetupSetDirectoryId().
//
#define DIRID_USER             0x8000


//
// Setup callback notification routine type
//
typedef UINT (CALLBACK* PSP_FILE_CALLBACK_A)(
    IN PVOID Context,
    IN UINT Notification,
    IN UINT Param1,
    IN UINT Param2
    );

typedef UINT (CALLBACK* PSP_FILE_CALLBACK_W)(
    IN PVOID Context,
    IN UINT Notification,
    IN UINT Param1,
    IN UINT Param2
    );

#ifdef UNICODE
#define PSP_FILE_CALLBACK PSP_FILE_CALLBACK_W
#else
#define PSP_FILE_CALLBACK PSP_FILE_CALLBACK_A
#endif


//
// Operation/queue start/end notification. These are ordinal values.
//
#define SPFILENOTIFY_STARTQUEUE        0x00000001
#define SPFILENOTIFY_ENDQUEUE          0x00000002
#define SPFILENOTIFY_STARTSUBQUEUE     0x00000003
#define SPFILENOTIFY_ENDSUBQUEUE       0x00000004
#define SPFILENOTIFY_STARTDELETE       0x00000005
#define SPFILENOTIFY_ENDDELETE         0x00000006
#define SPFILENOTIFY_DELETEERROR       0x00000007
#define SPFILENOTIFY_STARTRENAME       0x00000008
#define SPFILENOTIFY_ENDRENAME         0x00000009
#define SPFILENOTIFY_RENAMEERROR       0x0000000a
#define SPFILENOTIFY_STARTCOPY         0x0000000b
#define SPFILENOTIFY_ENDCOPY           0x0000000c
```

```
#define SPFILENOTIFY_COPYERROR          0x0000000d
#define SPFILENOTIFY_NEEDMEDIA          0x0000000e
#define SPFILENOTIFY_QUEUESCAN          0x0000000f
//
// These are used with SetupIterateCabinet().
//
#define SPFILENOTIFY_CABINETINFO        0x00000010
#define SPFILENOTIFY_FILEINCABINET      0x00000011
#define SPFILENOTIFY_NEEDNEWCABINET     0x00000012
#define SPFILENOTIFY_FILEEXTRACTED      0x00000013

#define SPFILENOTIFY_FILEOPDELAYED      0x00000014

//
// Copy notification. These are bit flags that may be combined.
//
#define SPFILENOTIFY_LANGMISMATCH       0x00010000
#define SPFILENOTIFY_TARGETEXISTS       0x00020000
#define SPFILENOTIFY_TARGETNEWER        0x00040000

//
// File operation codes and callback outcomes.
//
#define FILEOP_COPY                     0
#define FILEOP_RENAME                   1
#define FILEOP_DELETE                   2

#define FILEOP_ABORT                    0
#define FILEOP_DOIT                     1
#define FILEOP_SKIP                     2
#define FILEOP_RETRY                    FILEOP_DOIT
#define FILEOP_NEWPATH                  4

//
// Flags in inf copy sections
//
#define COPYFLG_WARN_IF_SKIP            0x00000001      // warn if user tries to skip file
#define COPYFLG_NOSKIP                  0x00000002      // disallow skipping this file
#define COPYFLG_NOVERSIONCHECK          0x00000004      // ignore versions and overwrite target
#define COPYFLG_FORCE_FILE_IN_USE       0x00000008      // force file-in-use behavior
#define COPYFLG_NO_OVERWRITE            0x00000010      // do not copy if file exists on target
#define COPYFLG_NO_VERSION_DIALOG       0x00000020      // do not copy if target is newer
#define COPYFLG_REPLACEONLY             0x00000400      // copy only if file exists on target

//
// Flags in inf delete sections
// New flags go in high word
//
#define DELFLG_IN_USE                   0x00000001      // queue in-use file for delete
#define DELFLG_IN_USE1                  0x00010000      // high-word version of DELFLG_IN_USE

//
// Source and file paths. Used when notifying queue callback
// of SPFILENOTIFY_STARTxxx, SPFILENOTIFY_ENDxxx, and SPFILENOTIFY_xxxERROR.
//
typedef struct _FILEPATHS_A {
```

```
    PCSTR  Target;
    PCSTR  Source;  // not used for delete operations
    UINT   Win32Error;
    DWORD  Flags;   // such as SP_COPY_NOSKIP for copy errors
} FILEPATHS_A, *PFILEPATHS_A;

typedef struct _FILEPATHS_W {
    PCWSTR Target;
    PCWSTR Source;  // not used for delete operations
    UINT   Win32Error;
    DWORD  Flags;   // such as SP_COPY_NOSKIP for copy errors
} FILEPATHS_W, *PFILEPATHS_W;

#ifdef UNICODE
typedef FILEPATHS_W FILEPATHS;
typedef PFILEPATHS_W PFILEPATHS;
#else
typedef FILEPATHS_A FILEPATHS;
typedef PFILEPATHS_A PFILEPATHS;
#endif



//
// Structure used with SPFILENOTIFY_NEEDMEDIA
//
typedef struct _SOURCE_MEDIA_A {
    PCSTR Reserved;
    PCSTR Tagfile;         // may be NULL
    PCSTR Description;
    //
    // Pathname part and filename part of source file
    // that caused us to need the media.
    //
    PCSTR SourcePath;
    PCSTR SourceFile;
    DWORD Flags;           // subset of SP_COPY_xxx
} SOURCE_MEDIA_A, *PSOURCE_MEDIA_A;

typedef struct _SOURCE_MEDIA_W {
    PCWSTR Reserved;
    PCWSTR Tagfile;        // may be NULL
    PCWSTR Description;
    //
    // Pathname part and filename part of source file
    // that caused us to need the media.
    //
    PCWSTR SourcePath;
    PCWSTR SourceFile;
    DWORD  Flags;          // subset of SP_COPY_xxx
} SOURCE_MEDIA_W, *PSOURCE_MEDIA_W;

#ifdef UNICODE
typedef SOURCE_MEDIA_W SOURCE_MEDIA;
typedef PSOURCE_MEDIA_W PSOURCE_MEDIA;
#else
typedef SOURCE_MEDIA_A SOURCE_MEDIA;
```

```
typedef PSOURCE_MEDIA_A PSOURCE_MEDIA;
#endif

//
// Structure used with SPFILENOTIFY_CABINETINFO and
// SPFILENOTIFY_NEEDNEWCABINET
//
typedef struct _CABINET_INFO_A {
    PCSTR CabinetPath;
    PCSTR CabinetFile;
    PCSTR DiskName;
    USHORT SetId;
    USHORT CabinetNumber;
} CABINET_INFO_A, *PCABINET_INFO_A;

typedef struct _CABINET_INFO_W {
    PCWSTR CabinetPath;
    PCWSTR CabinetFile;
    PCWSTR DiskName;
    USHORT SetId;
    USHORT CabinetNumber;
} CABINET_INFO_W, *PCABINET_INFO_W;

#ifdef UNICODE
typedef CABINET_INFO_W CABINET_INFO;
typedef PCABINET_INFO_W PCABINET_INFO;
#else
typedef CABINET_INFO_A CABINET_INFO;
typedef PCABINET_INFO_A PCABINET_INFO;
#endif

//
// Structure used with SPFILENOTIFY_FILEINCABINET
//
typedef struct _FILE_IN_CABINET_INFO_A {
    PCSTR NameInCabinet;
    DWORD FileSize;
    DWORD Win32Error;
    WORD  DosDate;
    WORD  DosTime;
    WORD  DosAttribs;
    CHAR  FullTargetName[MAX_PATH];
} FILE_IN_CABINET_INFO_A, *PFILE_IN_CABINET_INFO_A;

typedef struct _FILE_IN_CABINET_INFO_W {
    PCWSTR NameInCabinet;
    DWORD  FileSize;
    DWORD  Win32Error;
    WORD   DosDate;
    WORD   DosTime;
    WORD   DosAttribs;
    WCHAR  FullTargetName[MAX_PATH];
} FILE_IN_CABINET_INFO_W, *PFILE_IN_CABINET_INFO_W;

#ifdef UNICODE
typedef FILE_IN_CABINET_INFO_W FILE_IN_CABINET_INFO;
```

```
typedef PFILE_IN_CABINET_INFO_W PFILE_IN_CABINET_INFO;
#else
typedef FILE_IN_CABINET_INFO_A FILE_IN_CABINET_INFO;
typedef PFILE_IN_CABINET_INFO_A PFILE_IN_CABINET_INFO;
#endif


//
// Define type for setup file queue
//
typedef PVOID HSPFILEQ;

//
// Define type for setup disk space list
//
typedef PVOID HDSKSPC;

//
// Define type for reference to device information set
//
typedef PVOID HDEVINFO;

//
// Device information structure (references a device instance
// that is a member of a device information set)
//
typedef struct _SP_DEVINFO_DATA {
   DWORD cbSize;
   GUID  ClassGuid;
   DWORD DevInst;    // DEVINST handle
   DWORD Reserved;
} SP_DEVINFO_DATA, *PSP_DEVINFO_DATA;

//
// Interface device information structure (references an interface
// device that is associated with the device information element
// that owns it).
//
typedef struct _SP_INTERFACE_DEVICE_DATA {
   DWORD cbSize;
   GUID  InterfaceClassGuid;
   DWORD Flags;
   DWORD Reserved;
} SP_INTERFACE_DEVICE_DATA, *PSP_INTERFACE_DEVICE_DATA;

//
// Flags for SP_INTERFACE_DEVICE_DATA.Flags field.
//
#define SPID_ACTIVE     0x00000001
#define SPID_DEFAULT    0x00000002
#define SPID_REMOVED    0x00000004


typedef struct _SP_INTERFACE_DEVICE_DETAIL_DATA_A {
   DWORD  cbSize;
   CHAR   DevicePath[ANYSIZE_ARRAY];
```

```
} SP_INTERFACE_DEVICE_DETAIL_DATA_A, *PSP_INTERFACE_DEVICE_DETAIL_DATA_A;

typedef struct _SP_INTERFACE_DEVICE_DETAIL_DATA_W {
    DWORD  cbSize;
    WCHAR  DevicePath[ANYSIZE_ARRAY];
} SP_INTERFACE_DEVICE_DETAIL_DATA_W, *PSP_INTERFACE_DEVICE_DETAIL_DATA_W;

#ifdef UNICODE
typedef SP_INTERFACE_DEVICE_DETAIL_DATA_W SP_INTERFACE_DEVICE_DETAIL_DATA;
typedef PSP_INTERFACE_DEVICE_DETAIL_DATA_W PSP_INTERFACE_DEVICE_DETAIL_DATA;
#else
typedef SP_INTERFACE_DEVICE_DETAIL_DATA_A SP_INTERFACE_DEVICE_DETAIL_DATA;
typedef PSP_INTERFACE_DEVICE_DETAIL_DATA_A PSP_INTERFACE_DEVICE_DETAIL_DATA;
#endif



//
// Class installer function codes
//
#define DIF_SELECTDEVICE           0x00000001
#define DIF_INSTALLDEVICE          0x00000002
#define DIF_ASSIGNRESOURCES        0x00000003
#define DIF_PROPERTIES             0x00000004
#define DIF_REMOVE                 0x00000005
#define DIF_FIRSTTIMESETUP         0x00000006
#define DIF_FOUNDDEVICE            0x00000007
#define DIF_SELECTCLASSDRIVERS     0x00000008
#define DIF_VALIDATECLASSDRIVERS   0x00000009
#define DIF_INSTALLCLASSDRIVERS    0x0000000A
#define DIF_CALCDISKSPACE          0x0000000B
#define DIF_DESTROYPRIVATEDATA     0x0000000C
#define DIF_VALIDATEDRIVER         0x0000000D
#define DIF_MOVEDEVICE             0x0000000E
#define DIF_DETECT                 0x0000000F
#define DIF_INSTALLWIZARD          0x00000010
#define DIF_DESTROYWIZARDDATA      0x00000011
#define DIF_PROPERTYCHANGE         0x00000012
#define DIF_ENABLECLASS            0x00000013
#define DIF_DETECTVERIFY           0x00000014
#define DIF_INSTALLDEVICEFILES     0x00000015
#define DIF_REGISTERDEVICE         0x00000016
#define DIF_INSTALLINTERFACES      0x00000017

typedef UINT       DI_FUNCTION;    // Function type for device installer



//
// Device installation parameters structure (associated with a
// particular device information element, or globally with a device
// information set)
//
typedef struct _SP_DEVINSTALL_PARAMS_A {
    DWORD           cbSize;
    DWORD           Flags;
    DWORD           FlagsEx;
    HWND            hwndParent;
```

```
    PSP_FILE_CALLBACK InstallMsgHandler;
    PVOID           InstallMsgHandlerContext;
    HSPFILEQ         FileQueue;
    DWORD           ClassInstallReserved;
    DWORD           Reserved;
    CHAR            DriverPath[MAX_PATH];
} SP_DEVINSTALL_PARAMS_A, *PSP_DEVINSTALL_PARAMS_A;

typedef struct _SP_DEVINSTALL_PARAMS_W {
    DWORD           cbSize;
    DWORD           Flags;
    DWORD           FlagsEx;
    HWND            hwndParent;
    PSP_FILE_CALLBACK InstallMsgHandler;
    PVOID           InstallMsgHandlerContext;
    HSPFILEQ         FileQueue;
    DWORD           ClassInstallReserved;
    DWORD           Reserved;
    WCHAR           DriverPath[MAX_PATH];
} SP_DEVINSTALL_PARAMS_W, *PSP_DEVINSTALL_PARAMS_W;

#ifdef UNICODE
typedef SP_DEVINSTALL_PARAMS_W SP_DEVINSTALL_PARAMS;
typedef PSP_DEVINSTALL_PARAMS_W PSP_DEVINSTALL_PARAMS;
#else
typedef SP_DEVINSTALL_PARAMS_A SP_DEVINSTALL_PARAMS;
typedef PSP_DEVINSTALL_PARAMS_A PSP_DEVINSTALL_PARAMS;
#endif



//
// SP_DEVINSTALL_PARAMS.Flags values
//
// Flags for choosing a device
//
#define DI_SHOWOEM              0x00000001L     // support Other... button
#define DI_SHOWCOMPAT           0x00000002L     // show compatibility list
#define DI_SHOWCLASS            0x00000004L     // show class list
#define DI_SHOWALL              0x00000007L     // both class & compat list shown
#define DI_NOVCP                0x00000008L     // don't create a new copy queue--use
                                // caller-supplied FileQueue
#define DI_DIDCOMPAT            0x00000010L     // Searched for compatible devices
#define DI_DIDCLASS             0x00000020L     // Searched for class devices
#define DI_AUTOASSIGNRES        0x00000040L     // No UI for resources if possible

// flags returned by DiInstallDevice to indicate need to reboot/restart
#define DI_NEEDRESTART          0x00000080L     // Reboot required to take effect
#define DI_NEEDREBOOT           0x00000100L     // ""

// flags for device installation
#define DI_NOBROWSE             0x00000200L     // no Browse... in InsertDisk

// Flags set by DiBuildDriverInfoList
#define DI_MULTMFGS             0x00000400L     // Set if multiple manufacturers in
                                // class driver list
```

```
// Flag indicates that device is disabled
#define DI_DISABLED             0x00000800L     // Set if device disabled

// Flags for Device/Class Properties
#define DI_GENERALPAGE_ADDED    0x00001000L
#define DI_RESOURCEPAGE_ADDED   0x00002000L

// Flag to indicate the setting properties for this Device (or class) caused a change
// so the Dev Mgr UI probably needs to be updatd.
#define DI_PROPERTIES_CHANGE    0x00004000L

// Flag to indicate that the sorting from the INF file should be used.
#define DI_INF_IS_SORTED        0x00008000L

// Flag to indicate that only the the INF specified by SP_DEVINSTALL_PARAMS.DriverPath
// should be searched.
#define DI_ENUMSINGLEINF        0x00010000L

// Flag that prevents ConfigMgr from removing/re-enumerating devices during device
// registration, installation, and deletion.
#define DI_DONOTCALLCONFIGMG    0x00020000L

// The following flag can be used to install a device disabled
#define DI_INSTALLDISABLED      0x00040000L

// Flag that causes SetupDiBuildDriverInfoList to build a device's compatible driver
// list from its existing class driver list, instead of the normal INF search.
#define DI_COMPAT_FROM_CLASS    0x00080000L

// This flag is set if the Class Install params should be used.
#define DI_CLASSINSTALLPARAMS   0x00100000L

// This flag is set if the caller of DiCallClassInstaller does NOT
// want the internal default action performed if the Class installer
// returns ERROR_DI_DO_DEFAULT.
#define DI_NODI_DEFAULTACTION   0x00200000L

// The setupx flag, DI_NOSYNCPROCESSING (0x00400000L) is not support in the Setup APIs.

// flags for device installation
#define DI_QUIETINSTALL         0x00800000L     // don't confuse the user with
                                // questions or excess info
#define DI_NOFILECOPY           0x01000000L     // No file Copy necessary
#define DI_FORCECOPY            0x02000000L     // Force files to be copied from install path
#define DI_DRIVERPAGE_ADDED     0x04000000L     // Prop provider added Driver page.
#define DI_USECI_SELECTSTRINGS  0x08000000L     // Use Class Installer Provided strings in the Select Device Dlg
#define DI_OVERRIDE_INFFLAGS    0x10000000L     // Override INF flags
#define DI_PROPS_NOCHANGEUSAGE  0x20000000L     // No Enable/Disable in General Props

#define DI_NOSELECTICONS        0x40000000L     // No small icons in select device dialogs

#define DI_NOWRITE_IDS          0x80000000L     // Don't write HW & Compat IDs on install


//
// SP_DEVINSTALL_PARAMS.FlagsEx values
```

```
//
#define DI_FLAGSEX_USEOLDINFSEARCH      0x00000001L  // Inf Search functions should not use Index Search
#define DI_FLAGSEX_AUTOSELECTRANK0      0x00000002L  // SetupDiSelectDevice doesn't prompt user if rank 0 match
#define DI_FLAGSEX_CI_FAILED            0x00000004L  // Failed to Load/Call class installer

#define DI_FLAGSEX_DIDINFOLIST          0x00000010L  // Did the Class Info List
#define DI_FLAGSEX_DIDCOMPATINFO        0x00000020L  // Did the Compat Info List

#define DI_FLAGSEX_FILTERCLASSES        0x00000040L
#define DI_FLAGSEX_SETFAILEDINSTALL     0x00000080L
#define DI_FLAGSEX_DEVICECHANGE         0x00000100L
#define DI_FLAGSEX_ALWAYSWRITEIDS       0x00000200L
#define DI_FLAGSEX_ALLOWEXCLUDEDDRVS    0x00000800L
#define DI_FLAGSEX_NOUIONQUERYREMOVE    0x00001000L
#define DI_FLAGSEX_USECLASSFORCOMPAT    0x00002000L  // Use the device's class when building compat drv list.
                                        // (Ignored if DI_COMPAT_FROM_CLASS flag is specified.)

#define DI_FLAGSEX_OLDINF_IN_CLASSLIST  0x00004000L  // Search legacy INFs when building class driver list.

#define DI_FLAGSEX_NO_DRVREG_MODIFY     0x00008000L  // Don't run AddReg and DelReg for device's software (driver) key.

//
// Class installation parameters header.  This must be the first field of any class install
// parameter structure.  The InstallFunction field must be set to the function code
// corresponding to the structure, and the cbSize field must be set to the size of the
// header structure.  E.g.,
//
// SP_ENABLECLASS_PARAMS EnableClassParams;
//
// EnableClassParams.ClassInstallHeader.cbSize = sizeof(SP_CLASSINSTALL_HEADER);
// EnableClassParams.ClassInstallHeader.InstallFunction = DIF_ENABLECLASS;
//
typedef struct _SP_CLASSINSTALL_HEADER {
   DWORD      cbSize;
   DI_FUNCTION InstallFunction;
} SP_CLASSINSTALL_HEADER, *PSP_CLASSINSTALL_HEADER;


//
// Structure corresponding to a DIF_ENABLECLASS install function.
//
typedef struct _SP_ENABLECLASS_PARAMS {
   SP_CLASSINSTALL_HEADER ClassInstallHeader;
   GUID                   ClassGuid;
   DWORD                  EnableMessage;
} SP_ENABLECLASS_PARAMS, *PSP_ENABLECLASS_PARAMS;

#define ENABLECLASS_QUERY   0
#define ENABLECLASS_SUCCESS 1
#define ENABLECLASS_FAILURE 2


//
// Structure corresponding to a DIF_MOVEDEVICE install function.
//
typedef struct _SP_MOVEDEV_PARAMS {
```

```
    SP_CLASSINSTALL_HEADER ClassInstallHeader;
    SP_DEVINFO_DATA        SourceDeviceInfoData;
} SP_MOVEDEV_PARAMS, *PSP_MOVEDEV_PARAMS;


//
// Values indicating a change in a device's state
//
#define DICS_ENABLE      0x00000001
#define DICS_DISABLE     0x00000002
#define DICS_PROPCHANGE  0x00000003
#define DICS_START       0x00000004
#define DICS_STOP        0x00000005
//
// Values specifying the scope of a device property change
//
#define DICS_FLAG_GLOBAL         0x00000001  // make change in all hardware profiles
#define DICS_FLAG_CONFIGSPECIFIC 0x00000002  // make change in specified profile only
#define DICS_FLAG_CONFIGGENERAL  0x00000004  // 1 or more hardware profile-specific
                                             // changes to follow.
//
// Structure corresponding to a DIF_PROPERTYCHANGE install function.
//
typedef struct _SP_PROPCHANGE_PARAMS {
    SP_CLASSINSTALL_HEADER ClassInstallHeader;
    DWORD                  StateChange;
    DWORD                  Scope;
    DWORD                  HwProfile;
} SP_PROPCHANGE_PARAMS, *PSP_PROPCHANGE_PARAMS;


//
// Structure corresponding to a DIF_REMOVE install function.
//
typedef struct _SP_REMOVEDEVICE_PARAMS {
    SP_CLASSINSTALL_HEADER ClassInstallHeader;
    DWORD Scope;
    DWORD HwProfile;
} SP_REMOVEDEVICE_PARAMS, *PSP_REMOVEDEVICE_PARAMS;

#define DI_REMOVEDEVICE_GLOBAL              0x00000001
#define DI_REMOVEDEVICE_CONFIGSPECIFIC      0x00000002


//
// Structure corresponding to a DIF_SELECTDEVICE install function.
//
typedef struct _SP_SELECTDEVICE_PARAMS_A {
    SP_CLASSINSTALL_HEADER ClassInstallHeader;
    CHAR                   Title[MAX_TITLE_LEN];
    CHAR                   Instructions[MAX_INSTRUCTION_LEN];
    CHAR                   ListLabel[MAX_LABEL_LEN];
    BYTE                   Reserved[2];              // DWORD size alignment
} SP_SELECTDEVICE_PARAMS_A, *PSP_SELECTDEVICE_PARAMS_A;

typedef struct _SP_SELECTDEVICE_PARAMS_W {
```

```
    SP_CLASSINSTALL_HEADER ClassInstallHeader;
    WCHAR               Title[MAX_TITLE_LEN];
    WCHAR               Instructions[MAX_INSTRUCTION_LEN];
    WCHAR               ListLabel[MAX_LABEL_LEN];
} SP_SELECTDEVICE_PARAMS_W, *PSP_SELECTDEVICE_PARAMS_W;

#ifdef UNICODE
typedef SP_SELECTDEVICE_PARAMS_W SP_SELECTDEVICE_PARAMS;
typedef PSP_SELECTDEVICE_PARAMS_W PSP_SELECTDEVICE_PARAMS;
#else
typedef SP_SELECTDEVICE_PARAMS_A SP_SELECTDEVICE_PARAMS;
typedef PSP_SELECTDEVICE_PARAMS_A PSP_SELECTDEVICE_PARAMS;
#endif



//
// 'Add New Device' installation wizard structure
//
// Structure corresponding to a DIF_INSTALLWIZARD install function.
// (NOTE: This structure is also applicable for DIF_DESTROYWIZARDDATA,
// but DIF_INSTALLWIZARD is the associated function code in the class
// installation parameter structure in both cases.)
//
// Define maximum number of dynamic wizard pages that can be added to
// hardware install wizard.
//
#define MAX_INSTALLWIZARD_DYNAPAGES             20

typedef struct _SP_INSTALLWIZARD_DATA {
    SP_CLASSINSTALL_HEADER ClassInstallHeader;
    DWORD               Flags;
    HPROPSHEETPAGE      DynamicPages[MAX_INSTALLWIZARD_DYNAPAGES];
    DWORD               NumDynamicPages;
    DWORD               DynamicPageFlags;
    DWORD               PrivateFlags;
    LPARAM              PrivateData;
    HWND                hwndWizardDlg;
} SP_INSTALLWIZARD_DATA, *PSP_INSTALLWIZARD_DATA;

//
// SP_INSTALLWIZARD_DATA.Flags values
//
#define NDW_INSTALLFLAG_DIDFACTDEFS         0x00000001
#define NDW_INSTALLFLAG_HARDWAREALLREADYIN  0x00000002
#define NDW_INSTALLFLAG_NEEDRESTART         DI_NEEDRESTART
#define NDW_INSTALLFLAG_NEEDREBOOT          DI_NEEDREBOOT
#define NDW_INSTALLFLAG_NEEDSHUTDOWN        0x00000200
#define NDW_INSTALLFLAG_EXPRESSINTRO        0x00000400
#define NDW_INSTALLFLAG_SKIPISDEVINSTALLED  0x00000800
#define NDW_INSTALLFLAG_NODETECTEDDEVS      0x00001000
#define NDW_INSTALLFLAG_INSTALLSPECIFIC     0x00002000
#define NDW_INSTALLFLAG_SKIPCLASSLIST       0x00004000
#define NDW_INSTALLFLAG_CI_PICKED_OEM       0x00008000
#define NDW_INSTALLFLAG_PCMCIAMODE          0x00010000
#define NDW_INSTALLFLAG_PCMCIADEVICE        0x00020000
#define NDW_INSTALLFLAG_USERCANCEL          0x00040000
```

```
#define NDW_INSTALLFLAG_KNOWNCLASS          0x00080000


//
// SP_INSTALLWIZARD_DATA.DynamicPageFlags values
//
// This flag is set if a Class installer has added pages to the
// install wizard.
//
#define DYNAWIZ_FLAG_PAGESADDED             0x00000001

//
// The following flags will control the button states when displaying
// the InstallDetectedDevs dialog.
//
#define DYNAWIZ_FLAG_INSTALLDET_NEXT        0x00000002
#define DYNAWIZ_FLAG_INSTALLDET_PREV        0x00000004

// Set this flag if you jump to the analyze page, and want it to
// handle conflicts for you.  NOTE.  You will not get control back
// in the event of a conflict if you set this flag.
//
// BUGBUG (lonnym): Not currently implemented!
//
#define DYNAWIZ_FLAG_ANALYZE_HANDLECONFLICT 0x00000008

//
// Define wizard page resource IDs to be used when adding custom pages
// to the hardware install wizard.
//
// Resource ID for the first page that the install wizard will go to after
// adding the class installer pages.
//
#define IDD_DYNAWIZ_FIRSTPAGE               10000

//
// Resource ID for the page that the Select Device page will go back to.
//
#define IDD_DYNAWIZ_SELECT_PREVPAGE         10001

//
// Resource ID for the page that the Select Device page will go forward to.
//
#define IDD_DYNAWIZ_SELECT_NEXTPAGE         10002

//
// Resource ID for the page that the Analyze dialog should go back to
// This will only be used in the event that there is a problem, and the user
// selects Back from the analyze proc.
//
#define IDD_DYNAWIZ_ANALYZE_PREVPAGE        10003

//
// Resource ID for the page that the Analyze dialog should go to if it
// continue from the analyze proc.  the wAnalyzeResult in the INSTALLDATA
// struct will contain the anaysis results.
```

```
//
#define IDD_DYNAWIZ_ANALYZE_NEXTPAGE           10004

//
// Resource ID for that page that the Install detected devices page will go
// back to.
//
#define IDD_DYNAWIZ_INSTALLDETECTED_PREVPAGE    10006

//
// Resource ID for the page that the Install detected devices page will go
// forward to.
//
#define IDD_DYNAWIZ_INSTALLDETECTED_NEXTPAGE    10007

//
// Resource ID for the page that the Install detected devices page will go
// to in the event that no devices are detected.
//
#define IDD_DYNAWIZ_INSTALLDETECTED_NODEVS      10008

//
// Resource ID of the hardware install wizard's select device page.
// This ID can be used to go directly to the hardware install wizard's select
// device page.
//
#define IDD_DYNAWIZ_SELECTDEV_PAGE              10009

//
// Resource ID of the hardware install wizard's device analysis page.
// This ID can be use to go directly to the hardware install wizard's analysis
// page.
//
#define IDD_DYNAWIZ_ANALYZEDEV_PAGE             10010

//
// Resource ID of the hardware install wizard's install detected devices page.
// This ID can be use to go directly to the hardware install wizard's install
// detected devices page.
//
#define IDD_DYNAWIZ_INSTALLDETECTEDDEVS_PAGE    10011

//
// Resource ID of the hardware install wizard's select class page.
// This ID can be use to go directly to the hardware install wizard's select
// class page.
//
#define IDD_DYNAWIZ_SELECTCLASS_PAGE           10012


//
// Driver information structure (member of a driver info list that may be associated
// with a particular device instance, or (globally) with a device information set)
//
typedef struct _SP_DRVINFO_DATA_A {
    DWORD cbSize;
```

```
    DWORD DriverType;
    DWORD Reserved;
    CHAR  Description[LINE_LEN];
    CHAR  MfgName[LINE_LEN];
    CHAR  ProviderName[LINE_LEN];
} SP_DRVINFO_DATA_A, *PSP_DRVINFO_DATA_A;

typedef struct _SP_DRVINFO_DATA_W {
    DWORD cbSize;
    DWORD DriverType;
    DWORD Reserved;
    WCHAR Description[LINE_LEN];
    WCHAR MfgName[LINE_LEN];
    WCHAR ProviderName[LINE_LEN];
} SP_DRVINFO_DATA_W, *PSP_DRVINFO_DATA_W;

#ifdef UNICODE
typedef SP_DRVINFO_DATA_W SP_DRVINFO_DATA;
typedef PSP_DRVINFO_DATA_W PSP_DRVINFO_DATA;
#else
typedef SP_DRVINFO_DATA_A SP_DRVINFO_DATA;
typedef PSP_DRVINFO_DATA_A PSP_DRVINFO_DATA;
#endif


//
// Driver information details structure (provides detailed information about a
// particular driver information structure)
//
typedef struct _SP_DRVINFO_DETAIL_DATA_A {
    DWORD    cbSize;
    FILETIME InfDate;
    DWORD    CompatIDsOffset;
    DWORD    CompatIDsLength;
    DWORD    Reserved;
    CHAR     SectionName[LINE_LEN];
    CHAR     InfFileName[MAX_PATH];
    CHAR     DrvDescription[LINE_LEN];
    CHAR     HardwareID[ANYSIZE_ARRAY];
} SP_DRVINFO_DETAIL_DATA_A, *PSP_DRVINFO_DETAIL_DATA_A;

typedef struct _SP_DRVINFO_DETAIL_DATA_W {
    DWORD    cbSize;
    FILETIME InfDate;
    DWORD    CompatIDsOffset;
    DWORD    CompatIDsLength;
    DWORD    Reserved;
    WCHAR    SectionName[LINE_LEN];
    WCHAR    InfFileName[MAX_PATH];
    WCHAR    DrvDescription[LINE_LEN];
    WCHAR    HardwareID[ANYSIZE_ARRAY];
} SP_DRVINFO_DETAIL_DATA_W, *PSP_DRVINFO_DETAIL_DATA_W;

#ifdef UNICODE
typedef SP_DRVINFO_DETAIL_DATA_W SP_DRVINFO_DETAIL_DATA;
typedef PSP_DRVINFO_DETAIL_DATA_W PSP_DRVINFO_DETAIL_DATA;
```

```
#else
typedef SP_DRVINFO_DETAIL_DATA_A SP_DRVINFO_DETAIL_DATA;
typedef PSP_DRVINFO_DETAIL_DATA_A PSP_DRVINFO_DETAIL_DATA;
#endif



//
// Driver installation parameters (associated with a particular driver
// information element)
//
typedef struct _SP_DRVINSTALL_PARAMS {
   DWORD cbSize;
   DWORD Rank;
   DWORD Flags;
   DWORD PrivateData;
   DWORD Reserved;
} SP_DRVINSTALL_PARAMS, *PSP_DRVINSTALL_PARAMS;

//
// SP_DRVINSTALL_PARAMS.Flags values
//
#define DNF_DUPDESC         0x00000001   // Multiple providers have same desc
#define DNF_OLDDRIVER       0x00000002   // Driver node specifies old/current driver
#define DNF_EXCLUDEFROMLIST 0x00000004   // If set, this driver node will not be
                                         // displayed in any driver select dialogs.
#define DNF_NODRIVER        0x00000008   // if we want to install no driver
                                         // (e.g no mouse drv)
#define DNF_LEGACYINF       0x00000010   // this driver node comes from an old-style INF



//
// Setup callback routine for comparing detection signatures
//
typedef DWORD (CALLBACK* PSP_DETSIG_CMPPROC)(
   IN HDEVINFO         DeviceInfoSet,
   IN PSP_DEVINFO_DATA NewDeviceData,
   IN PSP_DEVINFO_DATA ExistingDeviceData,
   IN PVOID            CompareContext      OPTIONAL
   );



//
// Structure containing class image list information.
//
typedef struct _SP_CLASSIMAGELIST_DATA {
   DWORD      cbSize;
   HIMAGELIST ImageList;
   DWORD      Reserved;
} SP_CLASSIMAGELIST_DATA, *PSP_CLASSIMAGELIST_DATA;



//
// Structure to be passed as first parameter (LPVOID lpv) to ExtensionPropSheetPageProc
// entry point in setupapi.dll.  Used to retrieve a handle for a specified property page.
//
typedef struct _SP_PROPSHEETPAGE_REQUEST {
```

```
    DWORD          cbSize;
    DWORD          PageRequested;
    HDEVINFO       DeviceInfoSet;
    PSP_DEVINFO_DATA DeviceInfoData;
} SP_PROPSHEETPAGE_REQUEST, *PSP_PROPSHEETPAGE_REQUEST;

//
// Property sheet codes used in SP_PROPSHEETPAGE_REQUEST.PageRequested
//
#define SPPSR_SELECT_DEVICE_RESOURCES  1



//
// Setupapi-specific error codes
//
// Inf parse outcomes
//
#define ERROR_EXPECTED_SECTION_NAME  (APPLICATION_ERROR_MASK|ERROR_SEVERITY_ERROR|0)
#define ERROR_BAD_SECTION_NAME_LINE  (APPLICATION_ERROR_MASK|ERROR_SEVERITY_ERROR|1)
#define ERROR_SECTION_NAME_TOO_LONG  (APPLICATION_ERROR_MASK|ERROR_SEVERITY_ERROR|2)
#define ERROR_GENERAL_SYNTAX         (APPLICATION_ERROR_MASK|ERROR_SEVERITY_ERROR|3)
//
// Inf runtime errors
//
#define ERROR_WRONG_INF_STYLE        (APPLICATION_ERROR_MASK|ERROR_SEVERITY_ERROR|0x100)
#define ERROR_SECTION_NOT_FOUND      (APPLICATION_ERROR_MASK|ERROR_SEVERITY_ERROR|0x101)
#define ERROR_LINE_NOT_FOUND         (APPLICATION_ERROR_MASK|ERROR_SEVERITY_ERROR|0x102)
//
// Device Installer errors
//
#define ERROR_NO_ASSOCIATED_CLASS        (APPLICATION_ERROR_MASK|ERROR_SEVERITY_ERROR|0x200)
#define ERROR_CLASS_MISMATCH             (APPLICATION_ERROR_MASK|ERROR_SEVERITY_ERROR|0x201)
#define ERROR_DUPLICATE_FOUND            (APPLICATION_ERROR_MASK|ERROR_SEVERITY_ERROR|0x202)
#define ERROR_NO_DRIVER_SELECTED         (APPLICATION_ERROR_MASK|ERROR_SEVERITY_ERROR|0x203)
#define ERROR_KEY_DOES_NOT_EXIST         (APPLICATION_ERROR_MASK|ERROR_SEVERITY_ERROR|0x204)
#define ERROR_INVALID_DEVINST_NAME       (APPLICATION_ERROR_MASK|ERROR_SEVERITY_ERROR|0x205)
#define ERROR_INVALID_CLASS              (APPLICATION_ERROR_MASK|ERROR_SEVERITY_ERROR|0x206)
#define ERROR_DEVINST_ALREADY_EXISTS     (APPLICATION_ERROR_MASK|ERROR_SEVERITY_ERROR|0x207)
#define ERROR_DEVINFO_NOT_REGISTERED     (APPLICATION_ERROR_MASK|ERROR_SEVERITY_ERROR|0x208)
#define ERROR_INVALID_REG_PROPERTY       (APPLICATION_ERROR_MASK|ERROR_SEVERITY_ERROR|0x209)
#define ERROR_NO_INF                     (APPLICATION_ERROR_MASK|ERROR_SEVERITY_ERROR|0x20A)
#define ERROR_NO_SUCH_DEVINST            (APPLICATION_ERROR_MASK|ERROR_SEVERITY_ERROR|0x20B)
#define ERROR_CANT_LOAD_CLASS_ICON       (APPLICATION_ERROR_MASK|ERROR_SEVERITY_ERROR|0x20C)
#define ERROR_INVALID_CLASS_INSTALLER    (APPLICATION_ERROR_MASK|ERROR_SEVERITY_ERROR|0x20D)
#define ERROR_DI_DO_DEFAULT              (APPLICATION_ERROR_MASK|ERROR_SEVERITY_ERROR|0x20E)
#define ERROR_DI_NOFILECOPY              (APPLICATION_ERROR_MASK|ERROR_SEVERITY_ERROR|0x20F)
#define ERROR_INVALID_HWPROFILE          (APPLICATION_ERROR_MASK|ERROR_SEVERITY_ERROR|0x210)
#define ERROR_NO_DEVICE_SELECTED         (APPLICATION_ERROR_MASK|ERROR_SEVERITY_ERROR|0x211)
#define ERROR_DEVINFO_LIST_LOCKED        (APPLICATION_ERROR_MASK|ERROR_SEVERITY_ERROR|0x212)
#define ERROR_DEVINFO_DATA_LOCKED        (APPLICATION_ERROR_MASK|ERROR_SEVERITY_ERROR|0x213)
#define ERROR_DI_BAD_PATH                (APPLICATION_ERROR_MASK|ERROR_SEVERITY_ERROR|0x214)
#define ERROR_NO_CLASSINSTALL_PARAMS     (APPLICATION_ERROR_MASK|ERROR_SEVERITY_ERROR|0x215)
#define ERROR_FILEQUEUE_LOCKED           (APPLICATION_ERROR_MASK|ERROR_SEVERITY_ERROR|0x216)
#define ERROR_BAD_SERVICE_INSTALLSECT    (APPLICATION_ERROR_MASK|ERROR_SEVERITY_ERROR|0x217)
#define ERROR_NO_CLASS_DRIVER_LIST       (APPLICATION_ERROR_MASK|ERROR_SEVERITY_ERROR|0x218)
#define ERROR_NO_ASSOCIATED_SERVICE      (APPLICATION_ERROR_MASK|ERROR_SEVERITY_ERROR|0x219)
```

```
#define ERROR_NO_DEFAULT_INTERFACE_DEVICE (APPLICATION_ERROR_MASK|ERROR_SEVERITY_ERROR|0x21A)
#define ERROR_INTERFACE_DEVICE_ACTIVE     (APPLICATION_ERROR_MASK|ERROR_SEVERITY_ERROR|0x21B)
#define ERROR_INTERFACE_DEVICE_REMOVED    (APPLICATION_ERROR_MASK|ERROR_SEVERITY_ERROR|0x21C)
#define ERROR_BAD_INTERFACE_INSTALLSECT   (APPLICATION_ERROR_MASK|ERROR_SEVERITY_ERROR|0x21D)
#define ERROR_NO_SUCH_INTERFACE_CLASS     (APPLICATION_ERROR_MASK|ERROR_SEVERITY_ERROR|0x21E)
#define ERROR_INVALID_REFERENCE_STRING    (APPLICATION_ERROR_MASK|ERROR_SEVERITY_ERROR|0x21F)
#define ERROR_INVALID_MACHINENAME         (APPLICATION_ERROR_MASK|ERROR_SEVERITY_ERROR|0x220)
#define ERROR_REMOTE_COMM_FAILURE         (APPLICATION_ERROR_MASK|ERROR_SEVERITY_ERROR|0x221)
#define ERROR_MACHINE_UNAVAILABLE         (APPLICATION_ERROR_MASK|ERROR_SEVERITY_ERROR|0x222)
#define ERROR_NO_CONFIGMGR_SERVICES       (APPLICATION_ERROR_MASK|ERROR_SEVERITY_ERROR|0x223)


WINSETUPAPI
BOOL
WINAPI
SetupGetInfInformationA(
   IN  LPCVOID           InfSpec,
   IN  DWORD             SearchControl,
   OUT PSP_INF_INFORMATION ReturnBuffer,     OPTIONAL
   IN  DWORD             ReturnBufferSize,
   OUT PDWORD              RequiredSize      OPTIONAL
   );

WINSETUPAPI
BOOL
WINAPI
SetupGetInfInformationW(
   IN  LPCVOID           InfSpec,
   IN  DWORD             SearchControl,
   OUT PSP_INF_INFORMATION ReturnBuffer,     OPTIONAL
   IN  DWORD             ReturnBufferSize,
   OUT PDWORD              RequiredSize      OPTIONAL
   );

//
// SearchControl flags for SetupGetInfInformation
//
#define INFINFO_INF_SPEC_IS_HINF        1
#define INFINFO_INF_NAME_IS_ABSOLUTE    2
#define INFINFO_DEFAULT_SEARCH          3
#define INFINFO_REVERSE_DEFAULT_SEARCH  4
#define INFINFO_INF_PATH_LIST_SEARCH    5

#ifdef UNICODE
#define SetupGetInfInformation SetupGetInfInformationW
#else
#define SetupGetInfInformation SetupGetInfInformationA
#endif


WINSETUPAPI
BOOL
WINAPI
SetupQueryInfFileInformationA(
   IN  PSP_INF_INFORMATION InfInformation,
   IN  UINT                InfIndex,
```

```
    OUT PSTR              ReturnBuffer,     OPTIONAL
    IN  DWORD             ReturnBufferSize,
    OUT PDWORD            RequiredSize      OPTIONAL
    );

WINSETUPAPI
BOOL
WINAPI
SetupQueryInfFileInformationW(
    IN  PSP_INF_INFORMATION InfInformation,
    IN  UINT              InfIndex,
    OUT PWSTR             ReturnBuffer,     OPTIONAL
    IN  DWORD             ReturnBufferSize,
    OUT PDWORD            RequiredSize      OPTIONAL
    );

#ifdef UNICODE
#define SetupQueryInfFileInformation SetupQueryInfFileInformationW
#else
#define SetupQueryInfFileInformation SetupQueryInfFileInformationA
#endif


WINSETUPAPI
BOOL
WINAPI
SetupQueryInfVersionInformationA(
    IN  PSP_INF_INFORMATION InfInformation,
    IN  UINT              InfIndex,
    IN  PCSTR             Key,              OPTIONAL
    OUT PSTR              ReturnBuffer,     OPTIONAL
    IN  DWORD             ReturnBufferSize,
    OUT PDWORD            RequiredSize      OPTIONAL
    );

WINSETUPAPI
BOOL
WINAPI
SetupQueryInfVersionInformationW(
    IN  PSP_INF_INFORMATION InfInformation,
    IN  UINT              InfIndex,
    IN  PCWSTR            Key,              OPTIONAL
    OUT PWSTR             ReturnBuffer,     OPTIONAL
    IN  DWORD             ReturnBufferSize,
    OUT PDWORD            RequiredSize      OPTIONAL
    );

#ifdef UNICODE
#define SetupQueryInfVersionInformation SetupQueryInfVersionInformationW
#else
#define SetupQueryInfVersionInformation SetupQueryInfVersionInformationA
#endif


WINSETUPAPI
BOOL
```

```
WINAPI
SetupGetInfFileListA(
    IN  PCSTR  DirectoryPath,    OPTIONAL
    IN  DWORD  InfStyle,
    OUT PSTR   ReturnBuffer,     OPTIONAL
    IN  DWORD  ReturnBufferSize,
    OUT PDWORD RequiredSize      OPTIONAL
    );

WINSETUPAPI
BOOL
WINAPI
SetupGetInfFileListW(
    IN  PCWSTR DirectoryPath,    OPTIONAL
    IN  DWORD  InfStyle,
    OUT PWSTR  ReturnBuffer,     OPTIONAL
    IN  DWORD  ReturnBufferSize,
    OUT PDWORD RequiredSize      OPTIONAL
    );

#ifdef UNICODE
#define SetupGetInfFileList SetupGetInfFileListW
#else
#define SetupGetInfFileList SetupGetInfFileListA
#endif


WINSETUPAPI
HINF
WINAPI
SetupOpenInfFileW(
    IN  PCWSTR FileName,
    IN  PCWSTR InfClass,    OPTIONAL
    IN  DWORD  InfStyle,
    OUT PUINT  ErrorLine    OPTIONAL
    );

WINSETUPAPI
HINF
WINAPI
SetupOpenInfFileA(
    IN  PCSTR FileName,
    IN  PCSTR InfClass,     OPTIONAL
    IN  DWORD InfStyle,
    OUT PUINT ErrorLine     OPTIONAL
    );

#ifdef UNICODE
#define SetupOpenInfFile SetupOpenInfFileW
#else
#define SetupOpenInfFile SetupOpenInfFileA
#endif


WINSETUPAPI
HINF
```

```
WINAPI
SetupOpenMasterInf(
    VOID
    );


WINSETUPAPI
BOOL
WINAPI
SetupOpenAppendInfFileW(
    IN  PCWSTR FileName,    OPTIONAL
    IN  HINF   InfHandle,
    OUT PUINT  ErrorLine    OPTIONAL
    );

WINSETUPAPI
BOOL
WINAPI
SetupOpenAppendInfFileA(
    IN  PCSTR FileName,     OPTIONAL
    IN  HINF  InfHandle,
    OUT PUINT ErrorLine     OPTIONAL
    );

#ifdef UNICODE
#define SetupOpenAppendInfFile SetupOpenAppendInfFileW
#else
#define SetupOpenAppendInfFile SetupOpenAppendInfFileA
#endif


WINSETUPAPI
VOID
WINAPI
SetupCloseInfFile(
    IN HINF InfHandle
    );


WINSETUPAPI
BOOL
WINAPI
SetupFindFirstLineA(
    IN  HINF       InfHandle,
    IN  PCSTR      Section,
    IN  PCSTR      Key,         OPTIONAL
    OUT PINFCONTEXT Context
    );

WINSETUPAPI
BOOL
WINAPI
SetupFindFirstLineW(
    IN  HINF       InfHandle,
    IN  PCWSTR     Section,
    IN  PCWSTR     Key,         OPTIONAL
```

```
    OUT PINFCONTEXT Context
    );

#ifdef UNICODE
#define SetupFindFirstLine SetupFindFirstLineW
#else
#define SetupFindFirstLine SetupFindFirstLineA
#endif


WINSETUPAPI
BOOL
WINAPI
SetupFindNextLine(
    IN  PINFCONTEXT ContextIn,
    OUT PINFCONTEXT ContextOut
    );


WINSETUPAPI
BOOL
WINAPI
SetupFindNextMatchLineA(
    IN  PINFCONTEXT ContextIn,
    IN  PCSTR       Key,        OPTIONAL
    OUT PINFCONTEXT ContextOut
    );

WINSETUPAPI
BOOL
WINAPI
SetupFindNextMatchLineW(
    IN  PINFCONTEXT ContextIn,
    IN  PCWSTR      Key,        OPTIONAL
    OUT PINFCONTEXT ContextOut
    );

#ifdef UNICODE
#define SetupFindNextMatchLine SetupFindNextMatchLineW
#else
#define SetupFindNextMatchLine SetupFindNextMatchLineA
#endif


WINSETUPAPI
BOOL
WINAPI
SetupGetLineByIndexA(
    IN  HINF        InfHandle,
    IN  PCSTR       Section,
    IN  DWORD       Index,
    OUT PINFCONTEXT Context
    );

WINSETUPAPI
BOOL
```

```
WINAPI
SetupGetLineByIndexW(
   IN  HINF       InfHandle,
   IN  PCWSTR     Section,
   IN  DWORD      Index,
   OUT PINFCONTEXT Context
   );

#ifdef UNICODE
#define SetupGetLineByIndex SetupGetLineByIndexW
#else
#define SetupGetLineByIndex SetupGetLineByIndexA
#endif


WINSETUPAPI
LONG
WINAPI
SetupGetLineCountA(
   IN HINF  InfHandle,
   IN PCSTR Section
   );

WINSETUPAPI
LONG
WINAPI
SetupGetLineCountW(
   IN HINF   InfHandle,
   IN PCWSTR Section
   );

#ifdef UNICODE
#define SetupGetLineCount SetupGetLineCountW
#else
#define SetupGetLineCount SetupGetLineCountA
#endif


WINSETUPAPI
BOOL
WINAPI
SetupGetLineTextA(
   IN  PINFCONTEXT Context,         OPTIONAL
   IN  HINF        InfHandle,       OPTIONAL
   IN  PCSTR       Section,         OPTIONAL
   IN  PCSTR       Key,             OPTIONAL
   OUT PSTR        ReturnBuffer,    OPTIONAL
   IN  DWORD       ReturnBufferSize,
   OUT PDWORD      RequiredSize     OPTIONAL
   );

WINSETUPAPI
BOOL
WINAPI
SetupGetLineTextW(
   IN  PINFCONTEXT Context,         OPTIONAL
```

```
    IN  HINF        InfHandle,      OPTIONAL
    IN  PCWSTR      Section,        OPTIONAL
    IN  PCWSTR      Key,            OPTIONAL
    OUT PWSTR       ReturnBuffer,   OPTIONAL
    IN  DWORD       ReturnBufferSize,
    OUT PDWORD      RequiredSize    OPTIONAL
    );

#ifdef UNICODE
#define SetupGetLineText SetupGetLineTextW
#else
#define SetupGetLineText SetupGetLineTextA
#endif


WINSETUPAPI
DWORD
WINAPI
SetupGetFieldCount(
    IN PINFCONTEXT Context
    );


WINSETUPAPI
BOOL
WINAPI
SetupGetStringFieldA(
    IN  PINFCONTEXT Context,
    IN  DWORD       FieldIndex,
    OUT PSTR        ReturnBuffer,   OPTIONAL
    IN  DWORD       ReturnBufferSize,
    OUT PDWORD      RequiredSize    OPTIONAL
    );

WINSETUPAPI
BOOL
WINAPI
SetupGetStringFieldW(
    IN  PINFCONTEXT Context,
    IN  DWORD       FieldIndex,
    OUT PWSTR       ReturnBuffer,   OPTIONAL
    IN  DWORD       ReturnBufferSize,
    OUT PDWORD      RequiredSize    OPTIONAL
    );

#ifdef UNICODE
#define SetupGetStringField SetupGetStringFieldW
#else
#define SetupGetStringField SetupGetStringFieldA
#endif


WINSETUPAPI
BOOL
WINAPI
SetupGetIntField(
```

```
    IN  PINFCONTEXT Context,
    IN  DWORD       FieldIndex,
    OUT PINT        IntegerValue
    );


WINSETUPAPI
BOOL
WINAPI
SetupGetMultiSzFieldA(
    IN  PINFCONTEXT Context,
    IN  DWORD       FieldIndex,
    OUT PSTR        ReturnBuffer,     OPTIONAL
    IN  DWORD       ReturnBufferSize,
    OUT LPDWORD     RequiredSize      OPTIONAL
    );

WINSETUPAPI
BOOL
WINAPI
SetupGetMultiSzFieldW(
    IN  PINFCONTEXT Context,
    IN  DWORD       FieldIndex,
    OUT PWSTR       ReturnBuffer,     OPTIONAL
    IN  DWORD       ReturnBufferSize,
    OUT LPDWORD     RequiredSize      OPTIONAL
    );

#ifdef UNICODE
#define SetupGetMultiSzField SetupGetMultiSzFieldW
#else
#define SetupGetMultiSzField SetupGetMultiSzFieldA
#endif


WINSETUPAPI
BOOL
WINAPI
SetupGetBinaryField(
    IN  PINFCONTEXT Context,
    IN  DWORD       FieldIndex,
    OUT PBYTE       ReturnBuffer,     OPTIONAL
    IN  DWORD       ReturnBufferSize,
    OUT LPDWORD     RequiredSize      OPTIONAL
    );


WINSETUPAPI
DWORD
WINAPI
SetupGetFileCompressionInfoA(
    IN  PCSTR   SourceFileName,
    OUT PSTR   *ActualSourceFileName,
    OUT PDWORD  SourceFileSize,
    OUT PDWORD  TargetFileSize,
    OUT PUINT   CompressionType
```

```
    );

WINSETUPAPI
DWORD
WINAPI
SetupGetFileCompressionInfoW(
    IN  PCWSTR  SourceFileName,
    OUT PWSTR  *ActualSourceFileName,
    OUT PDWORD  SourceFileSize,
    OUT PDWORD  TargetFileSize,
    OUT PUINT   CompressionType
    );

#ifdef UNICODE
#define SetupGetFileCompressionInfo SetupGetFileCompressionInfoW
#else
#define SetupGetFileCompressionInfo SetupGetFileCompressionInfoA
#endif


//
// Compression types
//
#define FILE_COMPRESSION_NONE       0
#define FILE_COMPRESSION_WINLZA     1
#define FILE_COMPRESSION_MSZIP      2


WINSETUPAPI
DWORD
WINAPI
SetupDecompressOrCopyFileA(
    IN PCSTR SourceFileName,
    IN PCSTR TargetFileName,
    IN PUINT CompressionType OPTIONAL
    );

WINSETUPAPI
DWORD
WINAPI
SetupDecompressOrCopyFileW(
    IN PCWSTR SourceFileName,
    IN PCWSTR TargetFileName,
    IN PUINT  CompressionType OPTIONAL
    );

#ifdef UNICODE
#define SetupDecompressOrCopyFile SetupDecompressOrCopyFileW
#else
#define SetupDecompressOrCopyFile SetupDecompressOrCopyFileA
#endif


WINSETUPAPI
BOOL
WINAPI
```

```
SetupGetSourceFileLocationA(
    IN  HINF       InfHandle,
    IN  PINFCONTEXT InfContext,      OPTIONAL
    IN  PCSTR      FileName,         OPTIONAL
    OUT PUINT      SourceId,
    OUT PSTR       ReturnBuffer,     OPTIONAL
    IN  DWORD      ReturnBufferSize,
    OUT PDWORD     RequiredSize      OPTIONAL
    );

WINSETUPAPI
BOOL
WINAPI
SetupGetSourceFileLocationW(
    IN  HINF       InfHandle,
    IN  PINFCONTEXT InfContext,      OPTIONAL
    IN  PCWSTR     FileName,         OPTIONAL
    OUT PUINT      SourceId,
    OUT PWSTR      ReturnBuffer,     OPTIONAL
    IN  DWORD      ReturnBufferSize,
    OUT PDWORD     RequiredSize      OPTIONAL
    );

#ifdef UNICODE
#define SetupGetSourceFileLocation SetupGetSourceFileLocationW
#else
#define SetupGetSourceFileLocation SetupGetSourceFileLocationA
#endif


WINSETUPAPI
BOOL
WINAPI
SetupGetSourceFileSizeA(
    IN  HINF       InfHandle,
    IN  PINFCONTEXT InfContext,     OPTIONAL
    IN  PCSTR      FileName,        OPTIONAL
    IN  PCSTR      Section,         OPTIONAL
    OUT PDWORD     FileSize,
    IN  UINT       RoundingFactor  OPTIONAL
    );

WINSETUPAPI
BOOL
WINAPI
SetupGetSourceFileSizeW(
    IN  HINF       InfHandle,
    IN  PINFCONTEXT InfContext,     OPTIONAL
    IN  PCWSTR     FileName,        OPTIONAL
    IN  PCWSTR     Section,         OPTIONAL
    OUT PDWORD     FileSize,
    IN  UINT       RoundingFactor  OPTIONAL
    );

#ifdef UNICODE
#define SetupGetSourceFileSize SetupGetSourceFileSizeW
```

```
#else
#define SetupGetSourceFileSize SetupGetSourceFileSizeA
#endif


WINSETUPAPI
BOOL
WINAPI
SetupGetTargetPathA(
    IN  HINF       InfHandle,
    IN  PINFCONTEXT InfContext,      OPTIONAL
    IN  PCSTR      Section,          OPTIONAL
    OUT PSTR       ReturnBuffer,     OPTIONAL
    IN  DWORD      ReturnBufferSize,
    OUT PDWORD     RequiredSize      OPTIONAL
    );

WINSETUPAPI
BOOL
WINAPI
SetupGetTargetPathW(
    IN  HINF       InfHandle,
    IN  PINFCONTEXT InfContext,      OPTIONAL
    IN  PCWSTR     Section,          OPTIONAL
    OUT PWSTR      ReturnBuffer,     OPTIONAL
    IN  DWORD      ReturnBufferSize,
    OUT PDWORD     RequiredSize      OPTIONAL
    );

#ifdef UNICODE
#define SetupGetTargetPath SetupGetTargetPathW
#else
#define SetupGetTargetPath SetupGetTargetPathA
#endif


//
// Define flags for SourceList APIs.
//
#define SRCLIST_TEMPORARY       0x00000001
#define SRCLIST_NOBROWSE        0x00000002
#define SRCLIST_SYSTEM          0x00000010
#define SRCLIST_USER            0x00000020
#define SRCLIST_SYSIFADMIN      0x00000040
#define SRCLIST_SUBDIRS         0x00000100
#define SRCLIST_APPEND          0x00000200
#define SRCLIST_NOSTRIPPLATFORM 0x00000400


WINSETUPAPI
BOOL
WINAPI
SetupSetSourceListA(
    IN DWORD  Flags,
    IN PCSTR *SourceList,
    IN UINT   SourceCount
```

```
    );

WINSETUPAPI
BOOL
WINAPI
SetupSetSourceListW(
    IN DWORD   Flags,
    IN PCWSTR *SourceList,
    IN UINT    SourceCount
    );

#ifdef UNICODE
#define SetupSetSourceList SetupSetSourceListW
#else
#define SetupSetSourceList SetupSetSourceListA
#endif


WINSETUPAPI
BOOL
WINAPI
SetupCancelTemporarySourceList(
    VOID
    );


WINSETUPAPI
BOOL
WINAPI
SetupAddToSourceListA(
    IN DWORD Flags,
    IN PCSTR Source
    );

WINSETUPAPI
BOOL
WINAPI
SetupAddToSourceListW(
    IN DWORD  Flags,
    IN PCWSTR Source
    );

#ifdef UNICODE
#define SetupAddToSourceList SetupAddToSourceListW
#else
#define SetupAddToSourceList SetupAddToSourceListA
#endif


WINSETUPAPI
BOOL
WINAPI
SetupRemoveFromSourceListA(
    IN DWORD Flags,
    IN PCSTR Source
    );
```

```
WINSETUPAPI
BOOL
WINAPI
SetupRemoveFromSourceListW(
   IN DWORD  Flags,
   IN PCWSTR Source
   );

#ifdef UNICODE
#define SetupRemoveFromSourceList SetupRemoveFromSourceListW
#else
#define SetupRemoveFromSourceList SetupRemoveFromSourceListA
#endif


WINSETUPAPI
BOOL
WINAPI
SetupQuerySourceListA(
   IN  DWORD   Flags,
   OUT PCSTR **List,
   OUT PUINT   Count
   );

WINSETUPAPI
BOOL
WINAPI
SetupQuerySourceListW(
   IN  DWORD    Flags,
   OUT PCWSTR **List,
   OUT PUINT    Count
   );

#ifdef UNICODE
#define SetupQuerySourceList SetupQuerySourceListW
#else
#define SetupQuerySourceList SetupQuerySourceListA
#endif


WINSETUPAPI
BOOL
WINAPI
SetupFreeSourceListA(
   IN OUT PCSTR **List,
   IN     UINT    Count
   );

WINSETUPAPI
BOOL
WINAPI
SetupFreeSourceListW(
   IN OUT PCWSTR **List,
   IN     UINT     Count
   );
```

```
#ifdef UNICODE
#define SetupFreeSourceList SetupFreeSourceListW
#else
#define SetupFreeSourceList SetupFreeSourceListA
#endif


WINSETUPAPI
UINT
WINAPI
SetupPromptForDiskA(
    IN  HWND   hwndParent,
    IN  PCSTR  DialogTitle,      OPTIONAL
    IN  PCSTR  DiskName,         OPTIONAL
    IN  PCSTR  PathToSource,     OPTIONAL
    IN  PCSTR  FileSought,
    IN  PCSTR  TagFile,          OPTIONAL
    IN  DWORD  DiskPromptStyle,
    OUT PSTR   PathBuffer,
    IN  DWORD  PathBufferSize,
    OUT PDWORD PathRequiredSize
    );

WINSETUPAPI
UINT
WINAPI
SetupPromptForDiskW(
    IN  HWND   hwndParent,
    IN  PCWSTR DialogTitle,      OPTIONAL
    IN  PCWSTR DiskName,         OPTIONAL
    IN  PCWSTR PathToSource,     OPTIONAL
    IN  PCWSTR FileSought,
    IN  PCWSTR TagFile,          OPTIONAL
    IN  DWORD  DiskPromptStyle,
    OUT PWSTR  PathBuffer,
    IN  DWORD  PathBufferSize,
    OUT PDWORD PathRequiredSize
    );

#ifdef UNICODE
#define SetupPromptForDisk SetupPromptForDiskW
#else
#define SetupPromptForDisk SetupPromptForDiskA
#endif


WINSETUPAPI
UINT
WINAPI
SetupCopyErrorA(
    IN  HWND   hwndParent,
    IN  PCSTR  DialogTitle,     OPTIONAL
    IN  PCSTR  DiskName,        OPTIONAL
    IN  PCSTR  PathToSource,
    IN  PCSTR  SourceFile,
```

```
    IN  PCSTR  TargetPathFile,  OPTIONAL
    IN  UINT   Win32ErrorCode,
    IN  DWORD  Style,
    OUT PSTR   PathBuffer,      OPTIONAL
    IN  DWORD  PathBufferSize,
    OUT PDWORD PathRequiredSize OPTIONAL
    );

WINSETUPAPI
UINT
WINAPI
SetupCopyErrorW(
    IN  HWND   hwndParent,
    IN  PCWSTR DialogTitle,     OPTIONAL
    IN  PCWSTR DiskName,        OPTIONAL
    IN  PCWSTR PathToSource,
    IN  PCWSTR SourceFile,
    IN  PCWSTR TargetPathFile,  OPTIONAL
    IN  UINT   Win32ErrorCode,
    IN  DWORD  Style,
    OUT PWSTR  PathBuffer,      OPTIONAL
    IN  DWORD  PathBufferSize,
    OUT PDWORD PathRequiredSize OPTIONAL
    );

#ifdef UNICODE
#define SetupCopyError SetupCopyErrorW
#else
#define SetupCopyError SetupCopyErrorA
#endif


WINSETUPAPI
UINT
WINAPI
SetupRenameErrorA(
    IN  HWND   hwndParent,
    IN  PCSTR  DialogTitle,     OPTIONAL
    IN  PCSTR  SourceFile,
    IN  PCSTR  TargetFile,
    IN  UINT   Win32ErrorCode,
    IN  DWORD  Style
    );

WINSETUPAPI
UINT
WINAPI
SetupRenameErrorW(
    IN  HWND   hwndParent,
    IN  PCWSTR DialogTitle,     OPTIONAL
    IN  PCWSTR SourceFile,
    IN  PCWSTR TargetFile,
    IN  UINT   Win32ErrorCode,
    IN  DWORD  Style
    );
```

```
#ifdef UNICODE
#define SetupRenameError SetupRenameErrorW
#else
#define SetupRenameError SetupRenameErrorA
#endif


WINSETUPAPI
UINT
WINAPI
SetupDeleteErrorA(
    IN  HWND   hwndParent,
    IN  PCSTR  DialogTitle,     OPTIONAL
    IN  PCSTR  File,
    IN  UINT   Win32ErrorCode,
    IN  DWORD  Style
    );

WINSETUPAPI
UINT
WINAPI
SetupDeleteErrorW(
    IN  HWND   hwndParent,
    IN  PCWSTR DialogTitle,     OPTIONAL
    IN  PCWSTR File,
    IN  UINT   Win32ErrorCode,
    IN  DWORD  Style
    );

#ifdef UNICODE
#define SetupDeleteError SetupDeleteErrorW
#else
#define SetupDeleteError SetupDeleteErrorA
#endif


//
// Styles for SetupPromptForDisk, SetupCopyError,
// SetupRenameError, SetupDeleteError
//
#define IDF_NOBROWSE        0x00000001
#define IDF_NOSKIP          0x00000002
#define IDF_NODETAILS       0x00000004
#define IDF_NOCOMPRESSED    0x00000008
#define IDF_CHECKFIRST      0x00000100
#define IDF_NOBEEP          0x00000200
#define IDF_NOFOREGROUND    0x00000400
#define IDF_WARNIFSKIP      0x00000800
#define IDF_OEMDISK         0x80000000

//
// Return values for SetupPromptForDisk, SetupCopyError,
// SetupRenameError, SetupDeleteError
//
#define DPROMPT_SUCCESS         0
#define DPROMPT_CANCEL          1
```

```
#define DPROMPT_SKIPFILE       2
#define DPROMPT_BUFFERTOOSMALL  3
#define DPROMPT_OUTOFMEMORY     4


WINSETUPAPI
BOOL
WINAPI
SetupSetDirectoryIdA(
   IN HINF  InfHandle,
   IN DWORD Id,          OPTIONAL
   IN PCSTR Directory     OPTIONAL
   );

WINSETUPAPI
BOOL
WINAPI
SetupSetDirectoryIdW(
   IN HINF   InfHandle,
   IN DWORD  Id,         OPTIONAL
   IN PCWSTR Directory    OPTIONAL
   );

#ifdef UNICODE
#define SetupSetDirectoryId SetupSetDirectoryIdW
#else
#define SetupSetDirectoryId SetupSetDirectoryIdA
#endif


WINSETUPAPI
BOOL
WINAPI
SetupSetDirectoryIdExA(
   IN HINF  InfHandle,
   IN DWORD Id,          OPTIONAL
   IN PCSTR Directory,    OPTIONAL
   IN DWORD Flags,
   IN DWORD Reserved1,
   IN PVOID Reserved2
   );

WINSETUPAPI
BOOL
WINAPI
SetupSetDirectoryIdExW(
   IN HINF   InfHandle,
   IN DWORD  Id,         OPTIONAL
   IN PCWSTR Directory,   OPTIONAL
   IN DWORD  Flags,
   IN DWORD  Reserved1,
   IN PVOID  Reserved2
   );

#ifdef UNICODE
#define SetupSetDirectoryIdEx SetupSetDirectoryIdExW
```

```
#else
#define SetupSetDirectoryIdEx SetupSetDirectoryIdExA
#endif

//
// Flags for SetupSetDirectoryIdEx
//
#define SETDIRID_NOT_FULL_PATH      0x00000001


WINSETUPAPI
BOOL
WINAPI
SetupGetSourceInfoA(
    IN  HINF   InfHandle,
    IN  UINT   SourceId,
    IN  UINT   InfoDesired,
    OUT PSTR   ReturnBuffer,     OPTIONAL
    IN  DWORD  ReturnBufferSize,
    OUT PDWORD RequiredSize      OPTIONAL
    );

WINSETUPAPI
BOOL
WINAPI
SetupGetSourceInfoW(
    IN  HINF   InfHandle,
    IN  UINT   SourceId,
    IN  UINT   InfoDesired,
    OUT PWSTR  ReturnBuffer,     OPTIONAL
    IN  DWORD  ReturnBufferSize,
    OUT PDWORD RequiredSize      OPTIONAL
    );

#ifdef UNICODE
#define SetupGetSourceInfo SetupGetSourceInfoW
#else
#define SetupGetSourceInfo SetupGetSourceInfoA
#endif

//
// InfoDesired values for SetupGetSourceInfo
//
#define SRCINFO_PATH           1
#define SRCINFO_TAGFILE        2
#define SRCINFO_DESCRIPTION    3


WINSETUPAPI
BOOL
WINAPI
SetupInstallFileA(
    IN HINF            InfHandle,       OPTIONAL
    IN PINFCONTEXT     InfContext,      OPTIONAL
    IN PCSTR           SourceFile,      OPTIONAL
    IN PCSTR           SourcePathRoot,  OPTIONAL
```

```
    IN PCSTR             DestinationName,   OPTIONAL
    IN DWORD             CopyStyle,
    IN PSP_FILE_CALLBACK_A CopyMsgHandler,    OPTIONAL
    IN PVOID             Context          OPTIONAL
    );

WINSETUPAPI
BOOL
WINAPI
SetupInstallFileW(
    IN HINF              InfHandle,        OPTIONAL
    IN PINFCONTEXT       InfContext,       OPTIONAL
    IN PCWSTR            SourceFile,       OPTIONAL
    IN PCWSTR            SourcePathRoot,   OPTIONAL
    IN PCWSTR            DestinationName,  OPTIONAL
    IN DWORD             CopyStyle,
    IN PSP_FILE_CALLBACK_W CopyMsgHandler,   OPTIONAL
    IN PVOID             Context          OPTIONAL
    );

#ifdef UNICODE
#define SetupInstallFile SetupInstallFileW
#else
#define SetupInstallFile SetupInstallFileA
#endif

WINSETUPAPI
BOOL
WINAPI
SetupInstallFileExA(
    IN  HINF             InfHandle,        OPTIONAL
    IN  PINFCONTEXT      InfContext,       OPTIONAL
    IN  PCSTR            SourceFile,       OPTIONAL
    IN  PCSTR            SourcePathRoot,   OPTIONAL
    IN  PCSTR            DestinationName,  OPTIONAL
    IN  DWORD            CopyStyle,
    IN  PSP_FILE_CALLBACK_A CopyMsgHandler,   OPTIONAL
    IN  PVOID            Context,          OPTIONAL
    OUT PBOOL            FileWasInUse
    );

WINSETUPAPI
BOOL
WINAPI
SetupInstallFileExW(
    IN  HINF             InfHandle,        OPTIONAL
    IN  PINFCONTEXT      InfContext,       OPTIONAL
    IN  PCWSTR           SourceFile,       OPTIONAL
    IN  PCWSTR           SourcePathRoot,   OPTIONAL
    IN  PCWSTR           DestinationName,  OPTIONAL
    IN  DWORD            CopyStyle,
    IN  PSP_FILE_CALLBACK_W CopyMsgHandler,   OPTIONAL
    IN  PVOID            Context,          OPTIONAL
    OUT PBOOL            FileWasInUse
    );
```

```
#ifdef UNICODE
#define SetupInstallFileEx SetupInstallFileExW
#else
#define SetupInstallFileEx SetupInstallFileExA
#endif

//
// CopyStyle values for copy and queue-related APIs
//
#define SP_COPY_DELETESOURCE       0x0000001   // delete source file on successful copy
#define SP_COPY_REPLACEONLY        0x0000002   // copy only if target file already present
#define SP_COPY_NEWER              0x0000004   // copy only if source file newer than target
#define SP_COPY_NOOVERWRITE        0x0000008   // copy only if target doesn't exist
#define SP_COPY_NODECOMP           0x0000010   // don't decompress source file while copying
#define SP_COPY_LANGUAGEAWARE      0x0000020   // don't overwrite file of different language
#define SP_COPY_SOURCE_ABSOLUTE    0x0000040   // SourceFile is a full source path
#define SP_COPY_SOURCEPATH_ABSOLUTE 0x0000080   // SourcePathRoot is the full path
#define SP_COPY_IN_USE_NEEDS_REBOOT 0x0000100   // System needs reboot if file in use
#define SP_COPY_FORCE_IN_USE       0x0000200   // Force target-in-use behavior
#define SP_COPY_NOSKIP             0x0000400   // Skip is disallowed for this file or section
#define SP_FLAG_CABINETCONTINUATION 0x0000800   // Used with need media notification
#define SP_COPY_FORCE_NOOVERWRITE   0x0001000   // like NOOVERWRITE but no callback nofitication
#define SP_COPY_FORCE_NEWER        0x0002000   // like NEWER but no callback nofitication
#define SP_COPY_WARNIFSKIP         0x0004000   // system critical file: warn if user tries to skip
#define SP_COPY_NOBROWSE           0x0008000   // Browsing is disallowed for this file or section


WINSETUPAPI
HSPFILEQ
WINAPI
SetupOpenFileQueue(
   VOID
   );

WINSETUPAPI
BOOL
WINAPI
SetupCloseFileQueue(
   IN HSPFILEQ QueueHandle
   );


WINSETUPAPI
BOOL
WINAPI
SetupSetPlatformPathOverrideA(
   IN PCSTR Override   OPTIONAL
   );

WINSETUPAPI
BOOL
WINAPI
SetupSetPlatformPathOverrideW(
   IN PCWSTR Override  OPTIONAL
   );
```

```
#ifdef UNICODE
#define SetupSetPlatformPathOverride SetupSetPlatformPathOverrideW
#else
#define SetupSetPlatformPathOverride SetupSetPlatformPathOverrideA
#endif


WINSETUPAPI
BOOL
WINAPI
SetupQueueCopyA(
    IN HSPFILEQ QueueHandle,
    IN PCSTR    SourceRootPath,
    IN PCSTR    SourcePath,         OPTIONAL
    IN PCSTR    SourceFilename,
    IN PCSTR    SourceDescription,  OPTIONAL
    IN PCSTR    SourceTagfile,      OPTIONAL
    IN PCSTR    TargetDirectory,
    IN PCSTR    TargetFilename,     OPTIONAL
    IN DWORD    CopyStyle
    );

WINSETUPAPI
BOOL
WINAPI
SetupQueueCopyW(
    IN HSPFILEQ QueueHandle,
    IN PCWSTR   SourceRootPath,
    IN PCWSTR   SourcePath,         OPTIONAL
    IN PCWSTR   SourceFilename,
    IN PCWSTR   SourceDescription,  OPTIONAL
    IN PCWSTR   SourceTagfile,      OPTIONAL
    IN PCWSTR   TargetDirectory,
    IN PCWSTR   TargetFilename,     OPTIONAL
    IN DWORD    CopyStyle
    );

#ifdef UNICODE
#define SetupQueueCopy SetupQueueCopyW
#else
#define SetupQueueCopy SetupQueueCopyA
#endif


WINSETUPAPI
BOOL
WINAPI
SetupQueueDefaultCopyA(
    IN HSPFILEQ QueueHandle,
    IN HINF     InfHandle,
    IN PCSTR    SourceRootPath,
    IN PCSTR    SourceFilename,
    IN PCSTR    TargetFilename,
    IN DWORD    CopyStyle
    );
```

```
WINSETUPAPI
BOOL
WINAPI
SetupQueueDefaultCopyW(
    IN HSPFILEQ QueueHandle,
    IN HINF     InfHandle,
    IN PCWSTR   SourceRootPath,
    IN PCWSTR   SourceFilename,
    IN PCWSTR   TargetFilename,
    IN DWORD    CopyStyle
    );

#ifdef UNICODE
#define SetupQueueDefaultCopy SetupQueueDefaultCopyW
#else
#define SetupQueueDefaultCopy SetupQueueDefaultCopyA
#endif


WINSETUPAPI
BOOL
WINAPI
SetupQueueCopySectionA(
    IN HSPFILEQ QueueHandle,
    IN PCSTR    SourceRootPath,
    IN HINF     InfHandle,
    IN HINF     ListInfHandle,   OPTIONAL
    IN PCSTR    Section,
    IN DWORD    CopyStyle
    );

WINSETUPAPI
BOOL
WINAPI
SetupQueueCopySectionW(
    IN HSPFILEQ QueueHandle,
    IN PCWSTR   SourceRootPath,
    IN HINF     InfHandle,
    IN HINF     ListInfHandle,   OPTIONAL
    IN PCWSTR   Section,
    IN DWORD    CopyStyle
    );

#ifdef UNICODE
#define SetupQueueCopySection SetupQueueCopySectionW
#else
#define SetupQueueCopySection SetupQueueCopySectionA
#endif


WINSETUPAPI
BOOL
WINAPI
SetupQueueDeleteA(
    IN HSPFILEQ QueueHandle,
    IN PCSTR    PathPart1,
```

```
    IN PCSTR    PathPart2       OPTIONAL
    );

WINSETUPAPI
BOOL
WINAPI
SetupQueueDeleteW(
    IN HSPFILEQ QueueHandle,
    IN PCWSTR   PathPart1,
    IN PCWSTR   PathPart2       OPTIONAL
    );

#ifdef UNICODE
#define SetupQueueDelete SetupQueueDeleteW
#else
#define SetupQueueDelete SetupQueueDeleteA
#endif


WINSETUPAPI
BOOL
WINAPI
SetupQueueDeleteSectionA(
    IN HSPFILEQ QueueHandle,
    IN HINF     InfHandle,
    IN HINF     ListInfHandle,   OPTIONAL
    IN PCSTR    Section
    );

WINSETUPAPI
BOOL
WINAPI
SetupQueueDeleteSectionW(
    IN HSPFILEQ QueueHandle,
    IN HINF     InfHandle,
    IN HINF     ListInfHandle,   OPTIONAL
    IN PCWSTR   Section
    );

#ifdef UNICODE
#define SetupQueueDeleteSection SetupQueueDeleteSectionW
#else
#define SetupQueueDeleteSection SetupQueueDeleteSectionA
#endif


WINSETUPAPI
BOOL
WINAPI
SetupQueueRenameA(
    IN HSPFILEQ QueueHandle,
    IN PCSTR    SourcePath,
    IN PCSTR    SourceFilename, OPTIONAL
    IN PCSTR    TargetPath,     OPTIONAL
    IN PCSTR    TargetFilename
    );
```

```
WINSETUPAPI
BOOL
WINAPI
SetupQueueRenameW(
    IN HSPFILEQ QueueHandle,
    IN PCWSTR   SourcePath,
    IN PCWSTR   SourceFilename, OPTIONAL
    IN PCWSTR   TargetPath,     OPTIONAL
    IN PCWSTR   TargetFilename
    );

#ifdef UNICODE
#define SetupQueueRename SetupQueueRenameW
#else
#define SetupQueueRename SetupQueueRenameA
#endif


WINSETUPAPI
BOOL
WINAPI
SetupQueueRenameSectionA(
    IN HSPFILEQ QueueHandle,
    IN HINF     InfHandle,
    IN HINF     ListInfHandle,   OPTIONAL
    IN PCSTR    Section
    );

WINSETUPAPI
BOOL
WINAPI
SetupQueueRenameSectionW(
    IN HSPFILEQ QueueHandle,
    IN HINF     InfHandle,
    IN HINF     ListInfHandle,   OPTIONAL
    IN PCWSTR   Section
    );

#ifdef UNICODE
#define SetupQueueRenameSection SetupQueueRenameSectionW
#else
#define SetupQueueRenameSection SetupQueueRenameSectionA
#endif


WINSETUPAPI
BOOL
WINAPI
SetupCommitFileQueueA(
    IN HWND             Owner,        OPTIONAL
    IN HSPFILEQ         QueueHandle,
    IN PSP_FILE_CALLBACK_A MsgHandler,
    IN PVOID            Context
    );
```

```
WINSETUPAPI
BOOL
WINAPI
SetupCommitFileQueueW(
    IN HWND              Owner,        OPTIONAL
    IN HSPFILEQ          QueueHandle,
    IN PSP_FILE_CALLBACK_W MsgHandler,
    IN PVOID             Context
    );

#ifdef UNICODE
#define SetupCommitFileQueue SetupCommitFileQueueW
#else
#define SetupCommitFileQueue SetupCommitFileQueueA
#endif


WINSETUPAPI
BOOL
WINAPI
SetupScanFileQueueA(
    IN  HSPFILEQ          FileQueue,
    IN  DWORD             Flags,
    IN  HWND              Window,          OPTIONAL
    IN  PSP_FILE_CALLBACK_A CallbackRoutine,   OPTIONAL
    IN  PVOID             CallbackContext,   OPTIONAL
    OUT PDWORD            Result
    );

WINSETUPAPI
BOOL
WINAPI
SetupScanFileQueueW(
    IN  HSPFILEQ          FileQueue,
    IN  DWORD             Flags,
    IN  HWND              Window,          OPTIONAL
    IN  PSP_FILE_CALLBACK_W CallbackRoutine,   OPTIONAL
    IN  PVOID             CallbackContext,   OPTIONAL
    OUT PDWORD            Result
    );

#ifdef UNICODE
#define SetupScanFileQueue SetupScanFileQueueW
#else
#define SetupScanFileQueue SetupScanFileQueueA
#endif

//
// Define flags for SetupScanFileQueue.
//
#define SPQ_SCAN_FILE_PRESENCE  0x00000001
#define SPQ_SCAN_FILE_VALIDITY  0x00000002
#define SPQ_SCAN_USE_CALLBACK   0x00000004
#define SPQ_SCAN_INFORM_USER    0x00000010

//
```

```
// Define flags used with Param2 for SPFILENOTIFY_QUEUESCAN
//
#define SPQ_DELAYED_COPY       0x00000001  // file was in use; registered for delayed copy


//
// Disk space list APIs
//
WINSETUPAPI
HDSKSPC
WINAPI
SetupCreateDiskSpaceListA(
   IN PVOID Reserved1,
   IN DWORD Reserved2,
   IN UINT  Flags
   );

WINSETUPAPI
HDSKSPC
WINAPI
SetupCreateDiskSpaceListW(
   IN PVOID Reserved1,
   IN DWORD Reserved2,
   IN UINT  Flags
   );

#ifdef UNICODE
#define SetupCreateDiskSpaceList SetupCreateDiskSpaceListW
#else
#define SetupCreateDiskSpaceList SetupCreateDiskSpaceListA
#endif

//
// Flags for SetupCreateDiskSpaceList
//
#define SPDSL_IGNORE_DISK      0x00000001  // ignore deletes and on-disk files in copies


WINSETUPAPI
HDSKSPC
WINAPI
SetupDuplicateDiskSpaceListA(
   IN HDSKSPC DiskSpace,
   IN PVOID   Reserved1,
   IN DWORD   Reserved2,
   IN UINT    Flags
   );

WINSETUPAPI
HDSKSPC
WINAPI
SetupDuplicateDiskSpaceListW(
   IN HDSKSPC DiskSpace,
   IN PVOID   Reserved1,
   IN DWORD   Reserved2,
   IN UINT    Flags
```

```
    );

#ifdef UNICODE
#define SetupDuplicateDiskSpaceList SetupDuplicateDiskSpaceListW
#else
#define SetupDuplicateDiskSpaceList SetupDuplicateDiskSpaceListA
#endif


WINSETUPAPI
BOOL
WINAPI
SetupDestroyDiskSpaceList(
    IN OUT HDSKSPC DiskSpace
    );


WINSETUPAPI
BOOL
WINAPI
SetupQueryDrivesInDiskSpaceListA(
    IN  HDSKSPC DiskSpace,
    OUT PSTR    ReturnBuffer,      OPTIONAL
    IN  DWORD   ReturnBufferSize,
    OUT PDWORD  RequiredSize       OPTIONAL
    );

WINSETUPAPI
BOOL
WINAPI
SetupQueryDrivesInDiskSpaceListW(
    IN  HDSKSPC DiskSpace,
    OUT PWSTR   ReturnBuffer,      OPTIONAL
    IN  DWORD   ReturnBufferSize,
    OUT PDWORD  RequiredSize       OPTIONAL
    );

#ifdef UNICODE
#define SetupQueryDrivesInDiskSpaceList SetupQueryDrivesInDiskSpaceListW
#else
#define SetupQueryDrivesInDiskSpaceList SetupQueryDrivesInDiskSpaceListA
#endif


WINSETUPAPI
BOOL
WINAPI
SetupQuerySpaceRequiredOnDriveA(
    IN  HDSKSPC   DiskSpace,
    IN  PCSTR     DriveSpec,
    OUT LONGLONG *SpaceRequired,
    IN  PVOID     Reserved1,
    IN  UINT      Reserved2
    );

WINSETUPAPI
```

```
BOOL
WINAPI
SetupQuerySpaceRequiredOnDriveW(
   IN  HDSKSPC   DiskSpace,
   IN  PCWSTR    DriveSpec,
   OUT LONGLONG *SpaceRequired,
   IN  PVOID     Reserved1,
   IN  UINT      Reserved2
   );

#ifdef UNICODE
#define SetupQuerySpaceRequiredOnDrive SetupQuerySpaceRequiredOnDriveW
#else
#define SetupQuerySpaceRequiredOnDrive SetupQuerySpaceRequiredOnDriveA
#endif


WINSETUPAPI
BOOL
WINAPI
SetupAdjustDiskSpaceListA(
   IN HDSKSPC  DiskSpace,
   IN LPCSTR   DriveRoot,
   IN LONGLONG Amount,
   IN PVOID    Reserved1,
   IN UINT     Reserved2
   );

WINSETUPAPI
BOOL
WINAPI
SetupAdjustDiskSpaceListW(
   IN HDSKSPC  DiskSpace,
   IN LPCWSTR  DriveRoot,
   IN LONGLONG Amount,
   IN PVOID    Reserved1,
   IN UINT     Reserved2
   );

#ifdef UNICODE
#define SetupAdjustDiskSpaceList SetupAdjustDiskSpaceListW
#else
#define SetupAdjustDiskSpaceList SetupAdjustDiskSpaceListA
#endif


WINSETUPAPI
BOOL
WINAPI
SetupAddToDiskSpaceListA(
   IN HDSKSPC  DiskSpace,
   IN PCSTR    TargetFilespec,
   IN LONGLONG FileSize,
   IN UINT     Operation,
   IN PVOID    Reserved1,
   IN UINT     Reserved2
```

```
    );

WINSETUPAPI
BOOL
WINAPI
SetupAddToDiskSpaceListW(
    IN HDSKSPC  DiskSpace,
    IN PCWSTR   TargetFilespec,
    IN LONGLONG FileSize,
    IN UINT     Operation,
    IN PVOID    Reserved1,
    IN UINT     Reserved2
    );

#ifdef UNICODE
#define SetupAddToDiskSpaceList SetupAddToDiskSpaceListW
#else
#define SetupAddToDiskSpaceList SetupAddToDiskSpaceListA
#endif


WINSETUPAPI
BOOL
WINAPI
SetupAddSectionToDiskSpaceListA(
    IN HDSKSPC DiskSpace,
    IN HINF    InfHandle,
    IN HINF    ListInfHandle,  OPTIONAL
    IN PCSTR   SectionName,
    IN UINT    Operation,
    IN PVOID   Reserved1,
    IN UINT    Reserved2
    );

WINSETUPAPI
BOOL
WINAPI
SetupAddSectionToDiskSpaceListW(
    IN HDSKSPC DiskSpace,
    IN HINF    InfHandle,
    IN HINF    ListInfHandle,  OPTIONAL
    IN PCWSTR  SectionName,
    IN UINT    Operation,
    IN PVOID   Reserved1,
    IN UINT    Reserved2
    );

#ifdef UNICODE
#define SetupAddSectionToDiskSpaceList SetupAddSectionToDiskSpaceListW
#else
#define SetupAddSectionToDiskSpaceList SetupAddSectionToDiskSpaceListA
#endif


WINSETUPAPI
BOOL
```

```
WINAPI
SetupAddInstallSectionToDiskSpaceListA(
    IN HDSKSPC DiskSpace,
    IN HINF    InfHandle,
    IN HINF    LayoutInfHandle,    OPTIONAL
    IN PCSTR   SectionName,
    IN PVOID   Reserved1,
    IN UINT    Reserved2
    );

WINSETUPAPI
BOOL
WINAPI
SetupAddInstallSectionToDiskSpaceListW(
    IN HDSKSPC DiskSpace,
    IN HINF    InfHandle,
    IN HINF    LayoutInfHandle,    OPTIONAL
    IN PCWSTR  SectionName,
    IN PVOID   Reserved1,
    IN UINT    Reserved2
    );

#ifdef UNICODE
#define SetupAddInstallSectionToDiskSpaceList SetupAddInstallSectionToDiskSpaceListW
#else
#define SetupAddInstallSectionToDiskSpaceList SetupAddInstallSectionToDiskSpaceListA
#endif


WINSETUPAPI
BOOL
WINAPI
SetupRemoveFromDiskSpaceListA(
    IN HDSKSPC DiskSpace,
    IN PCSTR   TargetFilespec,
    IN UINT    Operation,
    IN PVOID   Reserved1,
    IN UINT    Reserved2
    );

WINSETUPAPI
BOOL
WINAPI
SetupRemoveFromDiskSpaceListW(
    IN HDSKSPC DiskSpace,
    IN PCWSTR  TargetFilespec,
    IN UINT    Operation,
    IN PVOID   Reserved1,
    IN UINT    Reserved2
    );

#ifdef UNICODE
#define SetupRemoveFromDiskSpaceList SetupRemoveFromDiskSpaceListW
#else
#define SetupRemoveFromDiskSpaceList SetupRemoveFromDiskSpaceListA
#endif
```

```
WINSETUPAPI
BOOL
WINAPI
SetupRemoveSectionFromDiskSpaceListA(
    IN HDSKSPC DiskSpace,
    IN HINF    InfHandle,
    IN HINF    ListInfHandle,  OPTIONAL
    IN PCSTR   SectionName,
    IN UINT    Operation,
    IN PVOID   Reserved1,
    IN UINT    Reserved2
    );

WINSETUPAPI
BOOL
WINAPI
SetupRemoveSectionFromDiskSpaceListW(
    IN HDSKSPC DiskSpace,
    IN HINF    InfHandle,
    IN HINF    ListInfHandle,  OPTIONAL
    IN PCWSTR  SectionName,
    IN UINT    Operation,
    IN PVOID   Reserved1,
    IN UINT    Reserved2
    );

#ifdef UNICODE
#define SetupRemoveSectionFromDiskSpaceList SetupRemoveSectionFromDiskSpaceListW
#else
#define SetupRemoveSectionFromDiskSpaceList SetupRemoveSectionFromDiskSpaceListA
#endif


WINSETUPAPI
BOOL
WINAPI
SetupRemoveInstallSectionFromDiskSpaceListA(
    IN HDSKSPC DiskSpace,
    IN HINF    InfHandle,
    IN HINF    LayoutInfHandle,    OPTIONAL
    IN PCSTR   SectionName,
    IN PVOID   Reserved1,
    IN UINT    Reserved2
    );

WINSETUPAPI
BOOL
WINAPI
SetupRemoveInstallSectionFromDiskSpaceListW(
    IN HDSKSPC DiskSpace,
    IN HINF    InfHandle,
    IN HINF    LayoutInfHandle,    OPTIONAL
    IN PCWSTR  SectionName,
    IN PVOID   Reserved1,
```

```
    IN UINT    Reserved2
    );

#ifdef UNICODE
#define SetupRemoveInstallSectionFromDiskSpaceList SetupRemoveInstallSectionFromDiskSpaceListW
#else
#define SetupRemoveInstallSectionFromDiskSpaceList SetupRemoveInstallSectionFromDiskSpaceListA
#endif



//
// Cabinet APIs
//

WINSETUPAPI
BOOL
WINAPI
SetupIterateCabinetA(
    IN  PCSTR             CabinetFile,
    IN  DWORD               Reserved,
    IN  PSP_FILE_CALLBACK_A MsgHandler,
    IN  PVOID             Context
    );

WINSETUPAPI
BOOL
WINAPI
SetupIterateCabinetW(
    IN  PCWSTR            CabinetFile,
    IN  DWORD             Reserved,
    IN  PSP_FILE_CALLBACK_W MsgHandler,
    IN  PVOID             Context
    );

#ifdef UNICODE
#define SetupIterateCabinet SetupIterateCabinetW
#else
#define SetupIterateCabinet SetupIterateCabinetA
#endif



WINSETUPAPI
INT
WINAPI
SetupPromptReboot(
    IN HSPFILEQ FileQueue,  OPTIONAL
    IN HWND     Owner,
    IN BOOL     ScanOnly
    );

//
// Define flags that are returned by SetupPromptReboot
//
#define SPFILEQ_FILE_IN_USE         0x00000001
#define SPFILEQ_REBOOT_RECOMMENDED  0x00000002
#define SPFILEQ_REBOOT_IN_PROGRESS  0x00000004
```

```
WINSETUPAPI
PVOID
WINAPI
SetupInitDefaultQueueCallback(
    IN HWND OwnerWindow
    );

WINSETUPAPI
PVOID
WINAPI
SetupInitDefaultQueueCallbackEx(
    IN HWND  OwnerWindow,
    IN HWND  AlternateProgressWindow, OPTIONAL
    IN UINT  ProgressMessage,
    IN DWORD Reserved1,
    IN PVOID Reserved2
    );

WINSETUPAPI
VOID
WINAPI
SetupTermDefaultQueueCallback(
    IN PVOID Context
    );

WINSETUPAPI
UINT
WINAPI
SetupDefaultQueueCallbackA(
    IN PVOID Context,
    IN UINT  Notification,
    IN UINT  Param1,
    IN UINT  Param2
    );

WINSETUPAPI
UINT
WINAPI
SetupDefaultQueueCallbackW(
    IN PVOID Context,
    IN UINT  Notification,
    IN UINT  Param1,
    IN UINT  Param2
    );

#ifdef UNICODE
#define SetupDefaultQueueCallback SetupDefaultQueueCallbackW
#else
#define SetupDefaultQueueCallback SetupDefaultQueueCallbackA
#endif


//
// Flags for AddReg section lines in INF.  The corresponding value
```

```
// is <ValueType> in the AddReg line format given below:
//
// <RegRootString>,<SubKey>,<ValueName>,<ValueType>,<Value>...
//
// The low word contains basic flags concerning the general data type
// and AddReg action. The high word contains values that more specifically
// identify the data type of the registry value.  The high word is ignored
// by the 16-bit Windows 95 SETUPX APIs.
//
#define FLG_ADDREG_BINVALUETYPE     ( 0x00000001 )
#define FLG_ADDREG_NOCLOBBER        ( 0x00000002 )
#define FLG_ADDREG_DELVAL           ( 0x00000004 )
#define FLG_ADDREG_APPEND           ( 0x00000008 ) // Currently supported only
                                    // for REG_MULTI_SZ values.
#define FLG_ADDREG_KEYONLY          ( 0x00000010 ) // Just create the key, ignore value

#define FLG_ADDREG_TYPE_MASK        ( 0xFFFF0000 | FLG_ADDREG_BINVALUETYPE )
#define FLG_ADDREG_TYPE_SZ          ( 0x00000000                           )
#define FLG_ADDREG_TYPE_MULTI_SZ    ( 0x00010000                           )
#define FLG_ADDREG_TYPE_EXPAND_SZ   ( 0x00020000                           )
#define FLG_ADDREG_TYPE_BINARY      ( 0x00000000 | FLG_ADDREG_BINVALUETYPE )
#define FLG_ADDREG_TYPE_DWORD       ( 0x00010000 | FLG_ADDREG_BINVALUETYPE )
#define FLG_ADDREG_TYPE_NONE        ( 0x00020000 | FLG_ADDREG_BINVALUETYPE )

//
// The INF may supply any arbitrary data type ordinal in the highword except
// for the following: REG_NONE, REG_SZ, REG_EXPAND_SZ, REG_MULTI_SZ.  If this
// technique is used, then the data is given in binary format, one byte per
// field.
//


WINSETUPAPI
BOOL
WINAPI
SetupInstallFromInfSectionA(
    IN HWND             Owner,
    IN HINF             InfHandle,
    IN PCSTR            SectionName,
    IN UINT             Flags,
    IN HKEY             RelativeKeyRoot,   OPTIONAL
    IN PCSTR            SourceRootPath,    OPTIONAL
    IN UINT             CopyFlags,
    IN PSP_FILE_CALLBACK_A MsgHandler,
    IN PVOID            Context,
    IN HDEVINFO         DeviceInfoSet,     OPTIONAL
    IN PSP_DEVINFO_DATA    DeviceInfoData     OPTIONAL
    );

WINSETUPAPI
BOOL
WINAPI
SetupInstallFromInfSectionW(
    IN HWND             Owner,
    IN HINF             InfHandle,
    IN PCWSTR           SectionName,
```

```
    IN UINT             Flags,
    IN HKEY             RelativeKeyRoot,   OPTIONAL
    IN PCWSTR           SourceRootPath,    OPTIONAL
    IN UINT             CopyFlags,
    IN PSP_FILE_CALLBACK_W MsgHandler,
    IN PVOID            Context,
    IN HDEVINFO         DeviceInfoSet,     OPTIONAL
    IN PSP_DEVINFO_DATA    DeviceInfoData     OPTIONAL
    );

#ifdef UNICODE
#define SetupInstallFromInfSection SetupInstallFromInfSectionW
#else
#define SetupInstallFromInfSection SetupInstallFromInfSectionA
#endif

//
// Flags for SetupInstallFromInfSection
//
#define SPINST_LOGCONFIG        0x00000001
#define SPINST_INIFILES         0x00000002
#define SPINST_REGISTRY         0x00000004
#define SPINST_INI2REG          0x00000008
#define SPINST_FILES            0x00000010
#define SPINST_ALL              0x0000001f


WINSETUPAPI
BOOL
WINAPI
SetupInstallFilesFromInfSectionA(
    IN HINF     InfHandle,
    IN HINF     LayoutInfHandle,    OPTIONAL
    IN HSPFILEQ FileQueue,
    IN PCSTR    SectionName,
    IN PCSTR    SourceRootPath,     OPTIONAL
    IN UINT     CopyFlags
    );

WINSETUPAPI
BOOL
WINAPI
SetupInstallFilesFromInfSectionW(
    IN HINF     InfHandle,
    IN HINF     LayoutInfHandle,    OPTIONAL
    IN HSPFILEQ FileQueue,
    IN PCWSTR   SectionName,
    IN PCWSTR   SourceRootPath,     OPTIONAL
    IN UINT     CopyFlags
    );

#ifdef UNICODE
#define SetupInstallFilesFromInfSection SetupInstallFilesFromInfSectionW
#else
#define SetupInstallFilesFromInfSection SetupInstallFilesFromInfSectionA
#endif
```

```
WINSETUPAPI
BOOL
WINAPI
SetupInstallServicesFromInfSectionA(
    IN HINF   InfHandle,
    IN PCSTR  SectionName,
    IN DWORD  Flags
    );

WINSETUPAPI
BOOL
WINAPI
SetupInstallServicesFromInfSectionW(
    IN HINF   InfHandle,
    IN PCWSTR SectionName,
    IN DWORD  Flags
    );

#ifdef UNICODE
#define SetupInstallServicesFromInfSection SetupInstallServicesFromInfSectionW
#else
#define SetupInstallServicesFromInfSection SetupInstallServicesFromInfSectionA
#endif

//
// Flags for SetupInstallServicesFromInfSection.  These flags are also used in
// the flags field of an AddService line in a device INF.  However, in that case,
// additional flags are permitted that are not used by this API.  These flags
// are marked as such below.
//
#define SPSVCINST_TAGTOFRONT    (0x00000001)  // move service's tag to front of its group order list
#define SPSVCINST_ASSOCSERVICE  (0x00000002)  // associate this service with the device being installed
                                              // (flag is ignored by SetupInstallServicesFromInfSection)

//
// Define handle type for Setup file log.
//
typedef PVOID HSPFILELOG;

WINSETUPAPI
HSPFILELOG
WINAPI
SetupInitializeFileLogA(
    IN PCSTR LogFileName,   OPTIONAL
    IN DWORD Flags
    );

WINSETUPAPI
HSPFILELOG
WINAPI
SetupInitializeFileLogW(
    IN PCWSTR LogFileName,  OPTIONAL
    IN DWORD  Flags
    );
```

```
#ifdef UNICODE
#define SetupInitializeFileLog SetupInitializeFileLogW
#else
#define SetupInitializeFileLog SetupInitializeFileLogA
#endif

//
// Flags for SetupInitializeFileLog
//
#define SPFILELOG_SYSTEMLOG     0x00000001  // use system log -- must be Administrator
#define SPFILELOG_FORCENEW      0x00000002  // not valid with SPFILELOG_SYSTEMLOG
#define SPFILELOG_QUERYONLY     0x00000004  // allows non-administrators to read system log


WINSETUPAPI
BOOL
WINAPI
SetupTerminateFileLog(
    IN HSPFILELOG FileLogHandle
    );


WINSETUPAPI
BOOL
WINAPI
SetupLogFileA(
    IN HSPFILELOG FileLogHandle,
    IN PCSTR      LogSectionName,   OPTIONAL
    IN PCSTR      SourceFilename,
    IN PCSTR      TargetFilename,
    IN DWORD      Checksum,         OPTIONAL
    IN PCSTR      DiskTagfile,      OPTIONAL
    IN PCSTR      DiskDescription,  OPTIONAL
    IN PCSTR      OtherInfo,        OPTIONAL
    IN DWORD      Flags
    );

WINSETUPAPI
BOOL
WINAPI
SetupLogFileW(
    IN HSPFILELOG FileLogHandle,
    IN PCWSTR     LogSectionName,   OPTIONAL
    IN PCWSTR     SourceFilename,
    IN PCWSTR     TargetFilename,
    IN DWORD      Checksum,         OPTIONAL
    IN PCWSTR     DiskTagfile,      OPTIONAL
    IN PCWSTR     DiskDescription,  OPTIONAL
    IN PCWSTR     OtherInfo,        OPTIONAL
    IN DWORD      Flags
    );

#ifdef UNICODE
#define SetupLogFile SetupLogFileW
#else
```

```
#define SetupLogFile SetupLogFileA
#endif

//
// Flags for SetupLogFile
//
#define SPFILELOG_OEMFILE   0x00000001


WINSETUPAPI
BOOL
WINAPI
SetupRemoveFileLogEntryA(
   IN HSPFILELOG FileLogHandle,
   IN PCSTR      LogSectionName,   OPTIONAL
   IN PCSTR      TargetFilename    OPTIONAL
   );

WINSETUPAPI
BOOL
WINAPI
SetupRemoveFileLogEntryW(
   IN HSPFILELOG FileLogHandle,
   IN PCWSTR     LogSectionName,   OPTIONAL
   IN PCWSTR     TargetFilename    OPTIONAL
   );

#ifdef UNICODE
#define SetupRemoveFileLogEntry SetupRemoveFileLogEntryW
#else
#define SetupRemoveFileLogEntry SetupRemoveFileLogEntryA
#endif


//
// Items retrievable from SetupQueryFileLog()
//
typedef enum {
   SetupFileLogSourceFilename,
   SetupFileLogChecksum,
   SetupFileLogDiskTagfile,
   SetupFileLogDiskDescription,
   SetupFileLogOtherInfo,
   SetupFileLogMax
} SetupFileLogInfo;

WINSETUPAPI
BOOL
WINAPI
SetupQueryFileLogA(
   IN  HSPFILELOG      FileLogHandle,
   IN  PCSTR           LogSectionName,   OPTIONAL
   IN  PCSTR           TargetFilename,
   IN  SetupFileLogInfo DesiredInfo,
   OUT PSTR            DataOut,          OPTIONAL
   IN  DWORD           ReturnBufferSize,
```

```
    OUT PDWORD          RequiredSize     OPTIONAL
    );

WINSETUPAPI
BOOL
WINAPI
SetupQueryFileLogW(
    IN  HSPFILELOG      FileLogHandle,
    IN  PCWSTR          LogSectionName,  OPTIONAL
    IN  PCWSTR          TargetFilename,
    IN  SetupFileLogInfo DesiredInfo,
    OUT PWSTR           DataOut,         OPTIONAL
    IN  DWORD           ReturnBufferSize,
    OUT PDWORD          RequiredSize     OPTIONAL
    );

#ifdef UNICODE
#define SetupQueryFileLog SetupQueryFileLogW
#else
#define SetupQueryFileLog SetupQueryFileLogA
#endif


//
// Device Installer APIs
//

WINSETUPAPI
HDEVINFO
WINAPI
SetupDiCreateDeviceInfoList(
    IN LPGUID ClassGuid, OPTIONAL
    IN HWND   hwndParent OPTIONAL
    );


WINSETUPAPI
BOOL
WINAPI
SetupDiGetDeviceInfoListClass(
    IN  HDEVINFO DeviceInfoSet,
    OUT LPGUID   ClassGuid
    );


//
// Flags for SetupDiCreateDeviceInfo
//
#define DICD_GENERATE_ID        0x00000001
#define DICD_INHERIT_CLASSDRVS  0x00000002

WINSETUPAPI
BOOL
WINAPI
SetupDiCreateDeviceInfoA(
    IN  HDEVINFO        DeviceInfoSet,
```

```
    IN  PCSTR           DeviceName,
    IN  LPGUID          ClassGuid,
    IN  PCSTR           DeviceDescription, OPTIONAL
    IN  HWND            hwndParent,       OPTIONAL
    IN  DWORD           CreationFlags,
    OUT PSP_DEVINFO_DATA DeviceInfoData     OPTIONAL
    );

WINSETUPAPI
BOOL
WINAPI
SetupDiCreateDeviceInfoW(
    IN  HDEVINFO        DeviceInfoSet,
    IN  PCWSTR          DeviceName,
    IN  LPGUID          ClassGuid,
    IN  PCWSTR          DeviceDescription, OPTIONAL
    IN  HWND            hwndParent,       OPTIONAL
    IN  DWORD           CreationFlags,
    OUT PSP_DEVINFO_DATA DeviceInfoData     OPTIONAL
    );

#ifdef UNICODE
#define SetupDiCreateDeviceInfo SetupDiCreateDeviceInfoW
#else
#define SetupDiCreateDeviceInfo SetupDiCreateDeviceInfoA
#endif


//
// Flags for SetupDiOpenDeviceInfo
//
#define DIOD_INHERIT_CLASSDRVS  0x00000002
#define DIOD_CANCEL_REMOVE      0x00000004

WINSETUPAPI
BOOL
WINAPI
SetupDiOpenDeviceInfoA(
    IN  HDEVINFO        DeviceInfoSet,
    IN  PCSTR           DeviceInstanceId,
    IN  HWND            hwndParent,      OPTIONAL
    IN  DWORD           OpenFlags,
    OUT PSP_DEVINFO_DATA DeviceInfoData    OPTIONAL
    );

WINSETUPAPI
BOOL
WINAPI
SetupDiOpenDeviceInfoW(
    IN  HDEVINFO        DeviceInfoSet,
    IN  PCWSTR          DeviceInstanceId,
    IN  HWND            hwndParent,      OPTIONAL
    IN  DWORD           OpenFlags,
    OUT PSP_DEVINFO_DATA DeviceInfoData    OPTIONAL
    );
```

```
#ifdef UNICODE
#define SetupDiOpenDeviceInfo SetupDiOpenDeviceInfoW
#else
#define SetupDiOpenDeviceInfo SetupDiOpenDeviceInfoA
#endif


WINSETUPAPI
BOOL
WINAPI
SetupDiGetDeviceInstanceIdA(
    IN  HDEVINFO         DeviceInfoSet,
    IN  PSP_DEVINFO_DATA DeviceInfoData,
    OUT PSTR             DeviceInstanceId,
    IN  DWORD            DeviceInstanceIdSize,
    OUT PDWORD           RequiredSize         OPTIONAL
    );

WINSETUPAPI
BOOL
WINAPI
SetupDiGetDeviceInstanceIdW(
    IN  HDEVINFO         DeviceInfoSet,
    IN  PSP_DEVINFO_DATA DeviceInfoData,
    OUT PWSTR            DeviceInstanceId,
    IN  DWORD            DeviceInstanceIdSize,
    OUT PDWORD           RequiredSize         OPTIONAL
    );

#ifdef UNICODE
#define SetupDiGetDeviceInstanceId SetupDiGetDeviceInstanceIdW
#else
#define SetupDiGetDeviceInstanceId SetupDiGetDeviceInstanceIdA
#endif


WINSETUPAPI
BOOL
WINAPI
SetupDiDeleteDeviceInfo(
    IN HDEVINFO         DeviceInfoSet,
    IN PSP_DEVINFO_DATA DeviceInfoData
    );


WINSETUPAPI
BOOL
WINAPI
SetupDiEnumDeviceInfo(
    IN  HDEVINFO         DeviceInfoSet,
    IN  DWORD            MemberIndex,
    OUT PSP_DEVINFO_DATA DeviceInfoData
    );


WINSETUPAPI
```

```
BOOL
WINAPI
SetupDiDestroyDeviceInfoList(
    IN HDEVINFO DeviceInfoSet
    );


WINSETUPAPI
BOOL
WINAPI
SetupDiEnumInterfaceDevice(
    IN  HDEVINFO                 DeviceInfoSet,
    IN  PSP_DEVINFO_DATA         DeviceInfoData,     OPTIONAL
    IN  LPGUID                   InterfaceClassGuid,
    IN  DWORD                    MemberIndex,
    OUT PSP_INTERFACE_DEVICE_DATA InterfaceDeviceData
    );


WINSETUPAPI
BOOL
WINAPI
SetupDiCreateInterfaceDeviceA(
    IN  HDEVINFO                 DeviceInfoSet,
    IN  PSP_DEVINFO_DATA         DeviceInfoData,
    IN  LPGUID                   InterfaceClassGuid,
    IN  PCSTR                    ReferenceString,    OPTIONAL
    IN  DWORD                    CreationFlags,
    OUT PSP_INTERFACE_DEVICE_DATA InterfaceDeviceData OPTIONAL
    );

WINSETUPAPI
BOOL
WINAPI
SetupDiCreateInterfaceDeviceW(
    IN  HDEVINFO                 DeviceInfoSet,
    IN  PSP_DEVINFO_DATA         DeviceInfoData,
    IN  LPGUID                   InterfaceClassGuid,
    IN  PCWSTR                   ReferenceString,    OPTIONAL
    IN  DWORD                    CreationFlags,
    OUT PSP_INTERFACE_DEVICE_DATA InterfaceDeviceData OPTIONAL
    );

#ifdef UNICODE
#define SetupDiCreateInterfaceDevice SetupDiCreateInterfaceDeviceW
#else
#define SetupDiCreateInterfaceDevice SetupDiCreateInterfaceDeviceA
#endif


WINSETUPAPI
BOOL
WINAPI
SetupDiOpenInterfaceDeviceA(
    IN  HDEVINFO                 DeviceInfoSet,
    IN  PCSTR                    DevicePath,
```

```
    IN  DWORD                   OpenFlags,
    OUT PSP_INTERFACE_DEVICE_DATA InterfaceDeviceData OPTIONAL
    );

WINSETUPAPI
BOOL
WINAPI
SetupDiOpenInterfaceDeviceW(
    IN  HDEVINFO                DeviceInfoSet,
    IN  PCWSTR                  DevicePath,
    IN  DWORD                   OpenFlags,
    OUT PSP_INTERFACE_DEVICE_DATA InterfaceDeviceData OPTIONAL
    );

#ifdef UNICODE
#define SetupDiOpenInterfaceDevice SetupDiOpenInterfaceDeviceW
#else
#define SetupDiOpenInterfaceDevice SetupDiOpenInterfaceDeviceA
#endif


WINSETUPAPI
BOOL
WINAPI
SetupDiDeleteInterfaceDeviceData(
    IN HDEVINFO                 DeviceInfoSet,
    IN PSP_INTERFACE_DEVICE_DATA InterfaceDeviceData
    );


WINSETUPAPI
BOOL
WINAPI
SetupDiRemoveInterfaceDevice(
    IN     HDEVINFO             DeviceInfoSet,
    IN OUT PSP_INTERFACE_DEVICE_DATA InterfaceDeviceData
    );


WINSETUPAPI
BOOL
WINAPI
SetupDiGetInterfaceDeviceDetailA(
    IN  HDEVINFO                    DeviceInfoSet,
    IN  PSP_INTERFACE_DEVICE_DATA       InterfaceDeviceData,
    OUT PSP_INTERFACE_DEVICE_DETAIL_DATA_A InterfaceDeviceDetailData,     OPTIONAL
    IN  DWORD                       InterfaceDeviceDetailDataSize,
    OUT PDWORD                      RequiredSize,               OPTIONAL
    OUT PSP_DEVINFO_DATA                DeviceInfoData              OPTIONAL
    );

WINSETUPAPI
BOOL
WINAPI
SetupDiGetInterfaceDeviceDetailW(
    IN  HDEVINFO                    DeviceInfoSet,
```

```
    IN  PSP_INTERFACE_DEVICE_DATA         InterfaceDeviceData,
    OUT PSP_INTERFACE_DEVICE_DETAIL_DATA_W InterfaceDeviceDetailData,     OPTIONAL
    IN  DWORD                          InterfaceDeviceDetailDataSize,
    OUT PDWORD                           RequiredSize,               OPTIONAL
    OUT PSP_DEVINFO_DATA                 DeviceInfoData              OPTIONAL
    );

#ifdef UNICODE
#define SetupDiGetInterfaceDeviceDetail SetupDiGetInterfaceDeviceDetailW
#else
#define SetupDiGetInterfaceDeviceDetail SetupDiGetInterfaceDeviceDetailA
#endif


//
// Default install handler for DIF_INSTALLINTERFACES.
//
WINSETUPAPI
BOOL
WINAPI
SetupDiInstallInterfaceDevices(
    IN HDEVINFO         DeviceInfoSet,
    IN PSP_DEVINFO_DATA DeviceInfoData
    );


//
// Default install handler for DIF_REGISTERDEVICE
//

//
// Flags for SetupDiRegisterDeviceInfo
//
#define SPRDI_FIND_DUPS       0x00000001

WINSETUPAPI
BOOL
WINAPI
SetupDiRegisterDeviceInfo(
    IN     HDEVINFO          DeviceInfoSet,
    IN OUT PSP_DEVINFO_DATA   DeviceInfoData,
    IN     DWORD             Flags,
    IN     PSP_DETSIG_CMPPROC CompareProc,      OPTIONAL
    IN     PVOID             CompareContext,   OPTIONAL
    OUT    PSP_DEVINFO_DATA   DupDeviceInfoData OPTIONAL
    );


//
// Ordinal values distinguishing between class drivers and
// device drivers.
// (Passed in 'DriverType' parameter of driver information list APIs)
//
#define SPDIT_NODRIVER          0x00000000
#define SPDIT_CLASSDRIVER       0x00000001
#define SPDIT_COMPATDRIVER      0x00000002
```

```
WINSETUPAPI
BOOL
WINAPI
SetupDiBuildDriverInfoList(
    IN     HDEVINFO        DeviceInfoSet,
    IN OUT PSP_DEVINFO_DATA DeviceInfoData, OPTIONAL
    IN     DWORD           DriverType
    );


WINSETUPAPI
BOOL
WINAPI
SetupDiCancelDriverInfoSearch(
    IN HDEVINFO DeviceInfoSet
    );


WINSETUPAPI
BOOL
WINAPI
SetupDiEnumDriverInfoA(
    IN  HDEVINFO          DeviceInfoSet,
    IN  PSP_DEVINFO_DATA   DeviceInfoData, OPTIONAL
    IN  DWORD             DriverType,
    IN  DWORD             MemberIndex,
    OUT PSP_DRVINFO_DATA_A DriverInfoData
    );

WINSETUPAPI
BOOL
WINAPI
SetupDiEnumDriverInfoW(
    IN  HDEVINFO          DeviceInfoSet,
    IN  PSP_DEVINFO_DATA   DeviceInfoData, OPTIONAL
    IN  DWORD             DriverType,
    IN  DWORD             MemberIndex,
    OUT PSP_DRVINFO_DATA_W DriverInfoData
    );

#ifdef UNICODE
#define SetupDiEnumDriverInfo SetupDiEnumDriverInfoW
#else
#define SetupDiEnumDriverInfo SetupDiEnumDriverInfoA
#endif


WINSETUPAPI
BOOL
WINAPI
SetupDiGetSelectedDriverA(
    IN  HDEVINFO          DeviceInfoSet,
    IN  PSP_DEVINFO_DATA   DeviceInfoData, OPTIONAL
    OUT PSP_DRVINFO_DATA_A DriverInfoData
    );
```

```
WINSETUPAPI
BOOL
WINAPI
SetupDiGetSelectedDriverW(
   IN  HDEVINFO          DeviceInfoSet,
   IN  PSP_DEVINFO_DATA   DeviceInfoData, OPTIONAL
   OUT PSP_DRVINFO_DATA_W DriverInfoData
   );

#ifdef UNICODE
#define SetupDiGetSelectedDriver SetupDiGetSelectedDriverW
#else
#define SetupDiGetSelectedDriver SetupDiGetSelectedDriverA
#endif


WINSETUPAPI
BOOL
WINAPI
SetupDiSetSelectedDriverA(
   IN     HDEVINFO          DeviceInfoSet,
   IN     PSP_DEVINFO_DATA   DeviceInfoData, OPTIONAL
   IN OUT PSP_DRVINFO_DATA_A DriverInfoData  OPTIONAL
   );

WINSETUPAPI
BOOL
WINAPI
SetupDiSetSelectedDriverW(
   IN     HDEVINFO          DeviceInfoSet,
   IN     PSP_DEVINFO_DATA   DeviceInfoData, OPTIONAL
   IN OUT PSP_DRVINFO_DATA_W DriverInfoData  OPTIONAL
   );

#ifdef UNICODE
#define SetupDiSetSelectedDriver SetupDiSetSelectedDriverW
#else
#define SetupDiSetSelectedDriver SetupDiSetSelectedDriverA
#endif


WINSETUPAPI
BOOL
WINAPI
SetupDiGetDriverInfoDetailA(
   IN  HDEVINFO                DeviceInfoSet,
   IN  PSP_DEVINFO_DATA        DeviceInfoData,         OPTIONAL
   IN  PSP_DRVINFO_DATA_A      DriverInfoData,
   OUT PSP_DRVINFO_DETAIL_DATA_A DriverInfoDetailData,    OPTIONAL
   IN  DWORD                   DriverInfoDetailDataSize,
   OUT PDWORD                  RequiredSize            OPTIONAL
   );

WINSETUPAPI
BOOL
```

```
WINAPI
SetupDiGetDriverInfoDetailW(
    IN  HDEVINFO                DeviceInfoSet,
    IN  PSP_DEVINFO_DATA        DeviceInfoData,         OPTIONAL
    IN  PSP_DRVINFO_DATA_W      DriverInfoData,
    OUT PSP_DRVINFO_DETAIL_DATA_W DriverInfoDetailData,     OPTIONAL
    IN  DWORD                   DriverInfoDetailDataSize,
    OUT PDWORD                  RequiredSize            OPTIONAL
    );

#ifdef UNICODE
#define SetupDiGetDriverInfoDetail SetupDiGetDriverInfoDetailW
#else
#define SetupDiGetDriverInfoDetail SetupDiGetDriverInfoDetailA
#endif


WINSETUPAPI
BOOL
WINAPI
SetupDiDestroyDriverInfoList(
    IN HDEVINFO        DeviceInfoSet,
    IN PSP_DEVINFO_DATA DeviceInfoData, OPTIONAL
    IN DWORD           DriverType
    );


//
// Flags controlling what is included in the device information set built
// by SetupDiGetClassDevs
//
#define DIGCF_DEFAULT          0x00000001  // only valid with DIGCF_INTERFACEDEVICE
#define DIGCF_PRESENT          0x00000002
#define DIGCF_ALLCLASSES       0x00000004
#define DIGCF_PROFILE          0x00000008
#define DIGCF_INTERFACEDEVICE   0x00000010

WINSETUPAPI
HDEVINFO
WINAPI
SetupDiGetClassDevsA(
    IN LPGUID ClassGuid,  OPTIONAL
    IN PCSTR  Enumerator, OPTIONAL
    IN HWND   hwndParent, OPTIONAL
    IN DWORD  Flags
    );

WINSETUPAPI
HDEVINFO
WINAPI
SetupDiGetClassDevsW(
    IN LPGUID ClassGuid,  OPTIONAL
    IN PCWSTR Enumerator, OPTIONAL
    IN HWND   hwndParent, OPTIONAL
    IN DWORD  Flags
    );
```

```
#ifdef UNICODE
#define SetupDiGetClassDevs SetupDiGetClassDevsW
#else
#define SetupDiGetClassDevs SetupDiGetClassDevsA
#endif


WINSETUPAPI
BOOL
WINAPI
SetupDiGetINFClassA(
    IN  PCSTR  InfName,
    OUT LPGUID ClassGuid,
    OUT PSTR   ClassName,
    IN  DWORD  ClassNameSize,
    OUT PDWORD RequiredSize   OPTIONAL
    );

WINSETUPAPI
BOOL
WINAPI
SetupDiGetINFClassW(
    IN  PCWSTR InfName,
    OUT LPGUID ClassGuid,
    OUT PWSTR  ClassName,
    IN  DWORD  ClassNameSize,
    OUT PDWORD RequiredSize   OPTIONAL
    );

#ifdef UNICODE
#define SetupDiGetINFClass SetupDiGetINFClassW
#else
#define SetupDiGetINFClass SetupDiGetINFClassA
#endif


//
// Flags controlling exclusion from the class information list built
// by SetupDiBuildClassInfoList
//
#define DIBCI_NOINSTALLCLASS   0x00000001
#define DIBCI_NODISPLAYCLASS   0x00000002

WINSETUPAPI
BOOL
WINAPI
SetupDiBuildClassInfoList(
    IN  DWORD  Flags,
    OUT LPGUID ClassGuidList,
    IN  DWORD  ClassGuidListSize,
    OUT PDWORD RequiredSize
    );


WINSETUPAPI
```

```
BOOL
WINAPI
SetupDiGetClassDescriptionA(
    IN  LPGUID ClassGuid,
    OUT PSTR   ClassDescription,
    IN  DWORD  ClassDescriptionSize,
    OUT PDWORD RequiredSize         OPTIONAL
    );

WINSETUPAPI
BOOL
WINAPI
SetupDiGetClassDescriptionW(
    IN  LPGUID ClassGuid,
    OUT PWSTR  ClassDescription,
    IN  DWORD  ClassDescriptionSize,
    OUT PDWORD RequiredSize         OPTIONAL
    );

#ifdef UNICODE
#define SetupDiGetClassDescription SetupDiGetClassDescriptionW
#else
#define SetupDiGetClassDescription SetupDiGetClassDescriptionA
#endif


WINSETUPAPI
BOOL
WINAPI
SetupDiCallClassInstaller(
    IN DI_FUNCTION      InstallFunction,
    IN HDEVINFO         DeviceInfoSet,
    IN PSP_DEVINFO_DATA DeviceInfoData OPTIONAL
    );


//
// Default install handler for DIF_SELECTDEVICE
//
WINSETUPAPI
BOOL
WINAPI
SetupDiSelectDevice(
    IN     HDEVINFO         DeviceInfoSet,
    IN OUT PSP_DEVINFO_DATA DeviceInfoData OPTIONAL
    );


//
// Default install handler for DIF_INSTALLDEVICE
//
WINSETUPAPI
BOOL
WINAPI
SetupDiInstallDevice(
    IN     HDEVINFO         DeviceInfoSet,
```

```
    IN OUT PSP_DEVINFO_DATA DeviceInfoData
    );


//
// Default install handler for DIF_INSTALLDEVICEFILES
//
WINSETUPAPI
BOOL
WINAPI
SetupDiInstallDriverFiles(
    IN HDEVINFO         DeviceInfoSet,
    IN PSP_DEVINFO_DATA DeviceInfoData OPTIONAL
    );


//
// Default install handler for DIF_REMOVE
//
WINSETUPAPI
BOOL
WINAPI
SetupDiRemoveDevice(
    IN     HDEVINFO         DeviceInfoSet,
    IN OUT PSP_DEVINFO_DATA DeviceInfoData
    );


//
// Default install handler for DIF_MOVEDEVICE
//
WINSETUPAPI
BOOL
WINAPI
SetupDiMoveDuplicateDevice(
    IN HDEVINFO         DeviceInfoSet,
    IN PSP_DEVINFO_DATA DestinationDeviceInfoData
    );


//
// Default install handler for DIF_PROPERTYCHANGE
//
WINSETUPAPI
BOOL
WINAPI
SetupDiChangeState(
    IN     HDEVINFO         DeviceInfoSet,
    IN OUT PSP_DEVINFO_DATA DeviceInfoData
    );


WINSETUPAPI
BOOL
WINAPI
SetupDiInstallClassA(
```

```
    IN HWND     hwndParent,  OPTIONAL
    IN PCSTR    InfFileName,
    IN DWORD    Flags,
    IN HSPFILEQ FileQueue    OPTIONAL
    );

WINSETUPAPI
BOOL
WINAPI
SetupDiInstallClassW(
    IN HWND     hwndParent,  OPTIONAL
    IN PCWSTR   InfFileName,
    IN DWORD    Flags,
    IN HSPFILEQ FileQueue    OPTIONAL
    );

#ifdef UNICODE
#define SetupDiInstallClass SetupDiInstallClassW
#else
#define SetupDiInstallClass SetupDiInstallClassA
#endif


WINSETUPAPI
BOOL
WINAPI
SetupDiInstallClassExA(
    IN HWND     hwndParent,        OPTIONAL
    IN PCSTR    InfFileName,       OPTIONAL
    IN DWORD    Flags,
    IN HSPFILEQ FileQueue,         OPTIONAL
    IN LPGUID   InterfaceClassGuid, OPTIONAL
    IN PVOID    Reserved
    );

WINSETUPAPI
BOOL
WINAPI
SetupDiInstallClassExW(
    IN HWND     hwndParent,        OPTIONAL
    IN PCWSTR   InfFileName,       OPTIONAL
    IN DWORD    Flags,
    IN HSPFILEQ FileQueue,         OPTIONAL
    IN LPGUID   InterfaceClassGuid, OPTIONAL
    IN PVOID    Reserved
    );

#ifdef UNICODE
#define SetupDiInstallClassEx SetupDiInstallClassExW
#else
#define SetupDiInstallClassEx SetupDiInstallClassExA
#endif


WINSETUPAPI
HKEY
```

```
WINAPI
SetupDiOpenClassRegKey(
    IN LPGUID ClassGuid, OPTIONAL
    IN REGSAM samDesired
    );



//
// Flags for SetupDiOpenClassRegKeyEx
//
#define DIOCR_INSTALLER   0x00000001    // class installer registry branch
#define DIOCR_INTERFACE   0x00000002    // interface class registry branch

WINSETUPAPI
HKEY
WINAPI
SetupDiOpenClassRegKeyExA(
    IN LPGUID ClassGuid,  OPTIONAL
    IN REGSAM samDesired,
    IN DWORD  Flags,
    IN PCSTR  MachineName OPTIONAL
    );

WINSETUPAPI
HKEY
WINAPI
SetupDiOpenClassRegKeyExW(
    IN LPGUID ClassGuid,  OPTIONAL
    IN REGSAM samDesired,
    IN DWORD  Flags,
    IN PCWSTR MachineName OPTIONAL
    );

#ifdef UNICODE
#define SetupDiOpenClassRegKeyEx SetupDiOpenClassRegKeyExW
#else
#define SetupDiOpenClassRegKeyEx SetupDiOpenClassRegKeyExA
#endif



WINSETUPAPI
HKEY
WINAPI
SetupDiCreateInterfaceDeviceRegKeyA(
    IN HDEVINFO                 DeviceInfoSet,
    IN PSP_INTERFACE_DEVICE_DATA InterfaceDeviceData,
    IN DWORD                    Reserved,
    IN REGSAM                   samDesired,
    IN HINF                     InfHandle,          OPTIONAL
    IN PCSTR                    InfSectionName      OPTIONAL
    );

WINSETUPAPI
HKEY
WINAPI
SetupDiCreateInterfaceDeviceRegKeyW(
```

```
    IN HDEVINFO                 DeviceInfoSet,
    IN PSP_INTERFACE_DEVICE_DATA InterfaceDeviceData,
    IN DWORD                    Reserved,
    IN REGSAM                   samDesired,
    IN HINF                     InfHandle,          OPTIONAL
    IN PCWSTR                   InfSectionName      OPTIONAL
    );

#ifdef UNICODE
#define SetupDiCreateInterfaceDeviceRegKey SetupDiCreateInterfaceDeviceRegKeyW
#else
#define SetupDiCreateInterfaceDeviceRegKey SetupDiCreateInterfaceDeviceRegKeyA
#endif


WINSETUPAPI
HKEY
WINAPI
SetupDiOpenInterfaceDeviceRegKey(
    IN HDEVINFO                 DeviceInfoSet,
    IN PSP_INTERFACE_DEVICE_DATA InterfaceDeviceData,
    IN DWORD                    Reserved,
    IN REGSAM                   samDesired
    );


WINSETUPAPI
BOOL
WINAPI
SetupDiDeleteInterfaceDeviceRegKey(
    IN HDEVINFO                 DeviceInfoSet,
    IN PSP_INTERFACE_DEVICE_DATA InterfaceDeviceData,
    IN DWORD                    Reserved
    );


//
// KeyType values for SetupDiCreateDevRegKey, SetupDiOpenDevRegKey, and
// SetupDiDeleteDevRegKey.
//
#define DIREG_DEV       0x00000001         // Open/Create/Delete device key
#define DIREG_DRV       0x00000002         // Open/Create/Delete driver key
#define DIREG_BOTH      0x00000004         // Delete both driver and Device key

WINSETUPAPI
HKEY
WINAPI
SetupDiCreateDevRegKeyA(
    IN HDEVINFO         DeviceInfoSet,
    IN PSP_DEVINFO_DATA DeviceInfoData,
    IN DWORD            Scope,
    IN DWORD            HwProfile,
    IN DWORD            KeyType,
    IN HINF             InfHandle,      OPTIONAL
    IN PCSTR            InfSectionName  OPTIONAL
    );
```

```
WINSETUPAPI
HKEY
WINAPI
SetupDiCreateDevRegKeyW(
    IN HDEVINFO        DeviceInfoSet,
    IN PSP_DEVINFO_DATA DeviceInfoData,
    IN DWORD           Scope,
    IN DWORD           HwProfile,
    IN DWORD           KeyType,
    IN HINF            InfHandle,      OPTIONAL
    IN PCWSTR          InfSectionName  OPTIONAL
    );

#ifdef UNICODE
#define SetupDiCreateDevRegKey SetupDiCreateDevRegKeyW
#else
#define SetupDiCreateDevRegKey SetupDiCreateDevRegKeyA
#endif


WINSETUPAPI
HKEY
WINAPI
SetupDiOpenDevRegKey(
    IN HDEVINFO        DeviceInfoSet,
    IN PSP_DEVINFO_DATA DeviceInfoData,
    IN DWORD           Scope,
    IN DWORD           HwProfile,
    IN DWORD           KeyType,
    IN REGSAM          samDesired
    );


WINSETUPAPI
BOOL
WINAPI
SetupDiDeleteDevRegKey(
    IN HDEVINFO        DeviceInfoSet,
    IN PSP_DEVINFO_DATA DeviceInfoData,
    IN DWORD           Scope,
    IN DWORD           HwProfile,
    IN DWORD           KeyType
    );


WINSETUPAPI
BOOL
WINAPI
SetupDiGetHwProfileList(
    OUT PDWORD HwProfileList,
    IN  DWORD  HwProfileListSize,
    OUT PDWORD RequiredSize,
    OUT PDWORD CurrentlyActiveIndex OPTIONAL
    );
```

```
//
// Device registry property codes
// (Codes marked as read-only (R) may only be used for
// SetupDiGetDeviceRegistryProperty)
//
// These values should cover the same set of registry properties
// as defined by the CM_DRP codes in cfgmgr32.h.
//
#define SPDRP_DEVICEDESC                (0x00000000)  // DeviceDesc (R/W)
#define SPDRP_HARDWAREID                (0x00000001)  // HardwareID (R/W)
#define SPDRP_COMPATIBLEIDS             (0x00000002)  // CompatibleIDs (R/W)
#define SPDRP_NTDEVICEPATHS             (0x00000003)  // Unsupported, DO NOT USE
#define SPDRP_SERVICE                   (0x00000004)  // Service (R/W)
#define SPDRP_CONFIGURATION             (0x00000005)  // Configuration (R)
#define SPDRP_CONFIGURATIONVECTOR       (0x00000006)  // ConfigurationVector (R)
#define SPDRP_CLASS                     (0x00000007)  // Class (R--tied to ClassGUID)
#define SPDRP_CLASSGUID                 (0x00000008)  // ClassGUID (R/W)
#define SPDRP_DRIVER                    (0x00000009)  // Driver (R/W)
#define SPDRP_CONFIGFLAGS               (0x0000000A)  // ConfigFlags (R/W)
#define SPDRP_MFG                       (0x0000000B)  // Mfg (R/W)
#define SPDRP_FRIENDLYNAME              (0x0000000C)  // FriendlyName (R/W)
#define SPDRP_LOCATION_INFORMATION      (0x0000000D)  // LocationInformation (R/W)
#define SPDRP_PHYSICAL_DEVICE_OBJECT_NAME (0x0000000E)  // PhysicalDeviceObjectName (R)
#define SPDRP_CAPABILITIES              (0x0000000F)  // Capabilities (R)
#define SPDRP_UI_NUMBER                 (0x00000010)  // UiNumber (R)
#define SPDRP_UPPERFILTERS              (0x00000011)  // UpperFilters (R/W)
#define SPDRP_LOWERFILTERS              (0x00000012)  // LowerFilters (R/W)
#define SPDRP_POWERENABLE               (0x00000013)  // PowerEnable (R/W)
#define SPDRP_MAXIMUM_PROPERTY          (0x00000014)  // Upper bound on ordinals

WINSETUPAPI
BOOL
WINAPI
SetupDiGetDeviceRegistryPropertyA(
    IN  HDEVINFO       DeviceInfoSet,
    IN  PSP_DEVINFO_DATA DeviceInfoData,
    IN  DWORD          Property,
    OUT PDWORD         PropertyRegDataType, OPTIONAL
    OUT PBYTE          PropertyBuffer,
    IN  DWORD          PropertyBufferSize,
    OUT PDWORD         RequiredSize       OPTIONAL
    );

WINSETUPAPI
BOOL
WINAPI
SetupDiGetDeviceRegistryPropertyW(
    IN  HDEVINFO       DeviceInfoSet,
    IN  PSP_DEVINFO_DATA DeviceInfoData,
    IN  DWORD          Property,
    OUT PDWORD         PropertyRegDataType, OPTIONAL
    OUT PBYTE          PropertyBuffer,
    IN  DWORD          PropertyBufferSize,
    OUT PDWORD         RequiredSize       OPTIONAL
    );
```

```
#ifdef UNICODE
#define SetupDiGetDeviceRegistryProperty SetupDiGetDeviceRegistryPropertyW
#else
#define SetupDiGetDeviceRegistryProperty SetupDiGetDeviceRegistryPropertyA
#endif


WINSETUPAPI
BOOL
WINAPI
SetupDiSetDeviceRegistryPropertyA(
    IN     HDEVINFO         DeviceInfoSet,
    IN OUT PSP_DEVINFO_DATA DeviceInfoData,
    IN     DWORD            Property,
    IN     CONST BYTE*      PropertyBuffer,
    IN     DWORD            PropertyBufferSize
    );

WINSETUPAPI
BOOL
WINAPI
SetupDiSetDeviceRegistryPropertyW(
    IN     HDEVINFO         DeviceInfoSet,
    IN OUT PSP_DEVINFO_DATA DeviceInfoData,
    IN     DWORD            Property,
    IN     CONST BYTE*      PropertyBuffer,
    IN     DWORD            PropertyBufferSize
    );

#ifdef UNICODE
#define SetupDiSetDeviceRegistryProperty SetupDiSetDeviceRegistryPropertyW
#else
#define SetupDiSetDeviceRegistryProperty SetupDiSetDeviceRegistryPropertyA
#endif


WINSETUPAPI
BOOL
WINAPI
SetupDiGetDeviceInstallParamsA(
    IN  HDEVINFO               DeviceInfoSet,
    IN  PSP_DEVINFO_DATA       DeviceInfoData,         OPTIONAL
    OUT PSP_DEVINSTALL_PARAMS_A DeviceInstallParams
    );

WINSETUPAPI
BOOL
WINAPI
SetupDiGetDeviceInstallParamsW(
    IN  HDEVINFO               DeviceInfoSet,
    IN  PSP_DEVINFO_DATA       DeviceInfoData,         OPTIONAL
    OUT PSP_DEVINSTALL_PARAMS_W DeviceInstallParams
    );

#ifdef UNICODE
```

```
#define SetupDiGetDeviceInstallParams SetupDiGetDeviceInstallParamsW
#else
#define SetupDiGetDeviceInstallParams SetupDiGetDeviceInstallParamsA
#endif



WINSETUPAPI
BOOL
WINAPI
SetupDiGetClassInstallParamsA(
    IN  HDEVINFO               DeviceInfoSet,
    IN  PSP_DEVINFO_DATA       DeviceInfoData,        OPTIONAL
    OUT PSP_CLASSINSTALL_HEADER ClassInstallParams,    OPTIONAL
    IN  DWORD                  ClassInstallParamsSize,
    OUT PDWORD                 RequiredSize           OPTIONAL
    );

WINSETUPAPI
BOOL
WINAPI
SetupDiGetClassInstallParamsW(
    IN  HDEVINFO               DeviceInfoSet,
    IN  PSP_DEVINFO_DATA       DeviceInfoData,        OPTIONAL
    OUT PSP_CLASSINSTALL_HEADER ClassInstallParams,    OPTIONAL
    IN  DWORD                  ClassInstallParamsSize,
    OUT PDWORD                 RequiredSize           OPTIONAL
    );

#ifdef UNICODE
#define SetupDiGetClassInstallParams SetupDiGetClassInstallParamsW
#else
#define SetupDiGetClassInstallParams SetupDiGetClassInstallParamsA
#endif



WINSETUPAPI
BOOL
WINAPI
SetupDiSetDeviceInstallParamsA(
    IN HDEVINFO               DeviceInfoSet,
    IN PSP_DEVINFO_DATA       DeviceInfoData,    OPTIONAL
    IN PSP_DEVINSTALL_PARAMS_A DeviceInstallParams
    );

WINSETUPAPI
BOOL
WINAPI
SetupDiSetDeviceInstallParamsW(
    IN HDEVINFO               DeviceInfoSet,
    IN PSP_DEVINFO_DATA       DeviceInfoData,    OPTIONAL
    IN PSP_DEVINSTALL_PARAMS_W DeviceInstallParams
    );

#ifdef UNICODE
#define SetupDiSetDeviceInstallParams SetupDiSetDeviceInstallParamsW
#else
```

```
#define SetupDiSetDeviceInstallParams SetupDiSetDeviceInstallParamsA
#endif


WINSETUPAPI
BOOL
WINAPI
SetupDiSetClassInstallParamsA(
    IN HDEVINFO               DeviceInfoSet,
    IN PSP_DEVINFO_DATA       DeviceInfoData,       OPTIONAL
    IN PSP_CLASSINSTALL_HEADER ClassInstallParams,  OPTIONAL
    IN DWORD                  ClassInstallParamsSize
    );

WINSETUPAPI
BOOL
WINAPI
SetupDiSetClassInstallParamsW(
    IN HDEVINFO               DeviceInfoSet,
    IN PSP_DEVINFO_DATA       DeviceInfoData,       OPTIONAL
    IN PSP_CLASSINSTALL_HEADER ClassInstallParams,  OPTIONAL
    IN DWORD                  ClassInstallParamsSize
    );

#ifdef UNICODE
#define SetupDiSetClassInstallParams SetupDiSetClassInstallParamsW
#else
#define SetupDiSetClassInstallParams SetupDiSetClassInstallParamsA
#endif


WINSETUPAPI
BOOL
WINAPI
SetupDiGetDriverInstallParamsA(
    IN  HDEVINFO              DeviceInfoSet,
    IN  PSP_DEVINFO_DATA      DeviceInfoData,     OPTIONAL
    IN  PSP_DRVINFO_DATA_A    DriverInfoData,
    OUT PSP_DRVINSTALL_PARAMS DriverInstallParams
    );

WINSETUPAPI
BOOL
WINAPI
SetupDiGetDriverInstallParamsW(
    IN  HDEVINFO              DeviceInfoSet,
    IN  PSP_DEVINFO_DATA      DeviceInfoData,     OPTIONAL
    IN  PSP_DRVINFO_DATA_W    DriverInfoData,
    OUT PSP_DRVINSTALL_PARAMS DriverInstallParams
    );

#ifdef UNICODE
#define SetupDiGetDriverInstallParams SetupDiGetDriverInstallParamsW
#else
#define SetupDiGetDriverInstallParams SetupDiGetDriverInstallParamsA
#endif
```

```
WINSETUPAPI
BOOL
WINAPI
SetupDiSetDriverInstallParamsA(
    IN HDEVINFO             DeviceInfoSet,
    IN PSP_DEVINFO_DATA     DeviceInfoData,     OPTIONAL
    IN PSP_DRVINFO_DATA_A   DriverInfoData,
    IN PSP_DRVINSTALL_PARAMS DriverInstallParams
    );

WINSETUPAPI
BOOL
WINAPI
SetupDiSetDriverInstallParamsW(
    IN HDEVINFO             DeviceInfoSet,
    IN PSP_DEVINFO_DATA     DeviceInfoData,     OPTIONAL
    IN PSP_DRVINFO_DATA_W   DriverInfoData,
    IN PSP_DRVINSTALL_PARAMS DriverInstallParams
    );

#ifdef UNICODE
#define SetupDiSetDriverInstallParams SetupDiSetDriverInstallParamsW
#else
#define SetupDiSetDriverInstallParams SetupDiSetDriverInstallParamsA
#endif


WINSETUPAPI
BOOL
WINAPI
SetupDiLoadClassIcon(
    IN  LPGUID  ClassGuid,
    OUT HICON  *LargeIcon,    OPTIONAL
    OUT PINT    MiniIconIndex OPTIONAL
    );


//
// Flags controlling the drawing of mini-icons
//
#define DMI_MASK      0x00000001
#define DMI_BKCOLOR   0x00000002
#define DMI_USERECT   0x00000004

WINSETUPAPI
INT
WINAPI
SetupDiDrawMiniIcon(
    IN HDC   hdc,
    IN RECT  rc,
    IN INT   MiniIconIndex,
    IN DWORD Flags
    );
```

```
WINSETUPAPI
BOOL
WINAPI
SetupDiGetClassBitmapIndex(
   IN  LPGUID ClassGuid,
   OUT PINT   MiniIconIndex
   );


WINSETUPAPI
BOOL
WINAPI
SetupDiGetClassImageList(
   OUT PSP_CLASSIMAGELIST_DATA ClassImageListData
   );


WINSETUPAPI
BOOL
WINAPI
SetupDiGetClassImageIndex(
   IN  PSP_CLASSIMAGELIST_DATA ClassImageListData,
   IN  LPGUID                  ClassGuid,
   OUT PINT                    ImageIndex
   );


WINSETUPAPI
BOOL
WINAPI
SetupDiDestroyClassImageList(
   IN PSP_CLASSIMAGELIST_DATA ClassImageListData
   );


//
// Flags for the SetupDiGetClassDevPropertySheets API
//
#define DIGCDP_FLAG_BASIC          0x00000001
#define DIGCDP_FLAG_ADVANCED       0x00000002

WINSETUPAPI
BOOL
WINAPI
SetupDiGetClassDevPropertySheetsA(
   IN HDEVINFO           DeviceInfoSet,
   IN PSP_DEVINFO_DATA   DeviceInfoData,     OPTIONAL
   IN LPPROPSHEETHEADERA PropertySheetHeader,
   IN DWORD              Flags
   );

WINSETUPAPI
BOOL
WINAPI
SetupDiGetClassDevPropertySheetsW(
```

```
    IN HDEVINFO         DeviceInfoSet,
    IN PSP_DEVINFO_DATA   DeviceInfoData,     OPTIONAL
    IN LPPROPSHEETHEADERW PropertySheetHeader,
    IN DWORD            Flags
    );

#ifdef UNICODE
#define SetupDiGetClassDevPropertySheets SetupDiGetClassDevPropertySheetsW
#else
#define SetupDiGetClassDevPropertySheets SetupDiGetClassDevPropertySheetsA
#endif


WINSETUPAPI
BOOL
WINAPI
SetupDiAskForOEMDisk(
    IN HDEVINFO         DeviceInfoSet,
    IN PSP_DEVINFO_DATA DeviceInfoData OPTIONAL
    );


WINSETUPAPI
BOOL
WINAPI
SetupDiSelectOEMDrv(
    IN     HWND           hwndParent,    OPTIONAL
    IN     HDEVINFO       DeviceInfoSet,
    IN OUT PSP_DEVINFO_DATA DeviceInfoData OPTIONAL
    );


WINSETUPAPI
BOOL
WINAPI
SetupDiClassNameFromGuidA(
    IN  LPGUID ClassGuid,
    OUT PSTR   ClassName,
    IN  DWORD  ClassNameSize,
    OUT PDWORD RequiredSize   OPTIONAL
    );

WINSETUPAPI
BOOL
WINAPI
SetupDiClassNameFromGuidW(
    IN  LPGUID ClassGuid,
    OUT PWSTR  ClassName,
    IN  DWORD  ClassNameSize,
    OUT PDWORD RequiredSize   OPTIONAL
    );

#ifdef UNICODE
#define SetupDiClassNameFromGuid SetupDiClassNameFromGuidW
#else
#define SetupDiClassNameFromGuid SetupDiClassNameFromGuidA
```

```
#endif


WINSETUPAPI
BOOL
WINAPI
SetupDiClassGuidsFromNameA(
    IN  PCSTR  ClassName,
    OUT LPGUID ClassGuidList,
    IN  DWORD  ClassGuidListSize,
    OUT PDWORD RequiredSize
    );

WINSETUPAPI
BOOL
WINAPI
SetupDiClassGuidsFromNameW(
    IN  PCWSTR ClassName,
    OUT LPGUID ClassGuidList,
    IN  DWORD  ClassGuidListSize,
    OUT PDWORD RequiredSize
    );

#ifdef UNICODE
#define SetupDiClassGuidsFromName SetupDiClassGuidsFromNameW
#else
#define SetupDiClassGuidsFromName SetupDiClassGuidsFromNameA
#endif


WINSETUPAPI
BOOL
WINAPI
SetupDiGetHwProfileFriendlyNameA(
    IN  DWORD  HwProfile,
    OUT PSTR   FriendlyName,
    IN  DWORD  FriendlyNameSize,
    OUT PDWORD RequiredSize     OPTIONAL
    );

WINSETUPAPI
BOOL
WINAPI
SetupDiGetHwProfileFriendlyNameW(
    IN  DWORD  HwProfile,
    OUT PWSTR  FriendlyName,
    IN  DWORD  FriendlyNameSize,
    OUT PDWORD RequiredSize     OPTIONAL
    );

#ifdef UNICODE
#define SetupDiGetHwProfileFriendlyName SetupDiGetHwProfileFriendlyNameW
#else
#define SetupDiGetHwProfileFriendlyName SetupDiGetHwProfileFriendlyNameA
#endif
```

```
//
// PageType values for SetupDiGetWizardPage API
//
#define SPWPT_SELECTDEVICE      0x00000001

//
// Flags for SetupDiGetWizardPage API
//
#define SPWP_USE_DEVINFO_DATA   0x00000001

WINSETUPAPI
HPROPSHEETPAGE
WINAPI
SetupDiGetWizardPage(
   IN HDEVINFO              DeviceInfoSet,
   IN PSP_DEVINFO_DATA      DeviceInfoData,    OPTIONAL
   IN PSP_INSTALLWIZARD_DATA InstallWizardData,
   IN DWORD                 PageType,
   IN DWORD                 Flags
   );


WINSETUPAPI
BOOL
WINAPI
SetupDiGetSelectedDevice(
   IN  HDEVINFO        DeviceInfoSet,
   OUT PSP_DEVINFO_DATA DeviceInfoData
   );


WINSETUPAPI
BOOL
WINAPI
SetupDiSetSelectedDevice(
   IN HDEVINFO        DeviceInfoSet,
   IN PSP_DEVINFO_DATA DeviceInfoData
   );


WINSETUPAPI
BOOL
WINAPI
SetupDiGetActualSectionToInstallA(
   IN  HINF    InfHandle,
   IN  PCSTR   InfSectionName,
   OUT PSTR    InfSectionWithExt,     OPTIONAL
   IN  DWORD   InfSectionWithExtSize,
   OUT PDWORD  RequiredSize,          OPTIONAL
   OUT PSTR   *Extension              OPTIONAL
   );

WINSETUPAPI
BOOL
WINAPI
```

```
SetupDiGetActualSectionToInstallW(
    IN  HINF    InfHandle,
    IN  PCWSTR  InfSectionName,
    OUT PWSTR   InfSectionWithExt,     OPTIONAL
    IN  DWORD   InfSectionWithExtSize,
    OUT PDWORD  RequiredSize,          OPTIONAL
    OUT PWSTR  *Extension              OPTIONAL
    );

#ifdef UNICODE
#define SetupDiGetActualSectionToInstall SetupDiGetActualSectionToInstallW
#else
#define SetupDiGetActualSectionToInstall SetupDiGetActualSectionToInstallA
#endif


#ifdef __cplusplus
}
#endif

#include <poppack.h>

#endif // _INC_SETUPAPI
```

LaunchINFSection & RunDLL32

LaunchINFSection

The LaunchINFSection function in Advanced INF Installer can be used to launch an advanced INF section using RunDLL32.EXE. This API allows user to use advpack.dll from command line without writing the program.

INT WINAPI LaunchINFSection(HWND hwnd, HINSTANCE hinst, PSTR pszCmdline, INT ishow);

Calling Syntax

rundll32.exe advpack.dll,LaunchINFSection *inf filename[,section name][,flags][,smart reboot]*

INF Filename

INF file pathname you want to launch.

Section Name

INF install section name you want to launch in the INF file.

Flags

Flag	Meaning
1	Quiet Mode
2	No GrpConv

Smart Reboot

N	No Reboot
A	Always Reboot
I	Reboot if Needed (default value)

Example:

rundll32.exe advpack.dll,LaunchINFSection myinf.inf,,3

This means install myinf.inf with DefaultInstall section in Quiet Mode with no GrpConv, reboot if needed.

What Happens?

When called, advpack executes the INF section in the same manner as if it was called using RunSetupCommand, except that the dialog title is taken from the INF (BeginPrompt section).

LaunchINFSectionEx & RunDLL32

LaunchINFSectionEx

The LaunchINFSectionEx function in the Advanced INF Installer can be used to launch an advanced INF section with SAVE/ROLLBACK capabilities. It can be called through rundll32.exe from command line.

HRESULT WINAPI LaunchINFSectionEx(HWND hwnd, HINSTANCE hInstance, PSTR pszParms, INT nShow);

Calling Syntax

rundll32.exe advpack.dll,LaunchINFSectionEx *inf_filename,[section name],[cab name],<flags>[,smart reboot]*

INF Filename

INF filename you want to launch. If the given name is not full pathname, advpack.dll will extract the INF from the given CAB file.

Section Name

INF install section name you want to launch. If it is empty string or NULL, DefaultInstall section name will be called.

Cab Name

Specify the fully qualified CAB file pathname which contains the files or INF you want to install to the user's system.

Flags

Flag	Meaning
4	Quiet Mode, no UI
8	Don't Run GrpConv
16	Force Self-Updating on User's System
32	Backup Data Before Install
64	Rollback to Previous State
128	Validate the Backup Data
256	Complete Rollback to Previous State
512	Force Delay of OCX Registration

Smart Reboot

N	No Reboot
A	Always Reboot
I	Reboot if Needed (default value)

Example:

rundll32.exe advpack.dll,LaunchINFSectionEx myinf.inf,,c:\temp\mydata.cab,36

This means to extract myinf.inf file from c:\temp\mydata.cab file and launch myinf.inf with DefaultInstall section in Quiet|Backup install mode, reboot if needed.

rundll32.exe advpack.dll,LaunchINFSectionEx c:\windows\inf\myinf.inf,,,256

This means to rollback to the state before installing myinf.inf DefaultInstall section.

For more information about LaunchINFSectionEx and Advpack.DLL, see advpub.h.

LaunchINFSectionEX

Advanced INF Installer

Overview

The Advanced INF Installer *(advpack.dll)* includes INF extensions that can perform tasks such as:

- Prompting a user for a destination directory
- Installing files to a directory specified in the registry
- Checking for mandatory applications before performing an upgrade.

This document describes the syntax of Advanced INF Installer and how to perform routine tasks using these features.

Syntax

[Version]
Signature="$Chicago$"
AdvancedINF=2.5, "You need a newer version of advpack.dll." - Error message if setup can't find a proper version of advpack.dll to use.

[InstallSection]
CustomDestination=CustomDestinationSection

[CustomDestinationSection]
CustomLDIDNumber=CustomLDIDSection, <Flag>

[CustomLDIDSection]
'<Root Key>','<Sub Key>','<Value Name>','<Message>','<Default to use>'

Note: Single quotes should be used instead of double quotes in the above syntax so the INF stays compatible with Windows NT 4.0 INF processing.

Parameters

Parameter	Description

<Flag>	An number that determines how the value of the custom LDID is determined. This number can be constructed by summing appropriate values from the table below.
<LDID Value>	The LDID value to which a value is assigned. It must be an integer value of at least 49000 and less than 50000. This is to avoid other reserved LDIDs in NT.
<Root Key>	A root key of a registry entry. Same format as AddReg and DelReg .INF file sections.
<Sub Key>	A sub key of a registry entry. Same format as AddReg and DelReg .INF file sections.
<Value Name>	A sub key of a registry entry. Same format as AddReg and DelReg .INF file sections. The default value for a key may be specified with a null string (two double quotes).
<Message>	A message that may be displayed to the user.
<Default>	A string that is used as a default value if no LDID value can be obtained from registry entries.

Flag Values

Flag Value	Description
1	**Default:** Make a registry sub key the value of custom LDID. Return branch name. **Set:** Make registry value the value of the custom LDID. Return a value in the branch.
2	**Default:** If none of the registry entries in the CustomLDIDSection exist, use the default as the LDID value. **Set:** Fail the installation if none of the registry entries exist. Upon failure, show the user the message specified on the last line of the CustomLDIDSection
4	**Default:** Display a dialog box prompting the user to enter a destination directory. Prompt the user with the message string corresponding to the registry entry found or to the default value. Fill the edit box with either the registry value or the default value. **Set:** Do not prompt the user. Use either the registry value found or the default value as the LDID value.
8	**Default:** Strip a semicolon from the end of the custom LDID value if one is there. **Set:** Do not strip a semicolon from the LDID value.
16	**Default:** Before assigning a registry value or sub key to an LDID, check to see if the registry **value** is a valid directory on the user's system. **Set:** Do not check the registry value to see if it is a valid directory.

32	**Default:** No effect on other flags if not set. **Set:** Has to be used in conjunction with flag2. Fails the install if any of the registry keys searched for exists. Reverses logic of flag2.

Note:

- If flag1 is not set then IExpress will not confirm a registry value it looks for is also a valid directory. (Flag16 will be set)
- Replaceable strings from the [Strings] section may be used in Advanced INF Installer sections. Standard LDIDs can also be used.
- Previous versions of IExpress could not create c:\myapp\samples\LFNDIR. This is no longer a limitation.
- Under Windows NT 4.0, any value assigned to an LDID must be in the form of a directory, otherwise the NT 4.0 setup engine (*setupapi.dll*) will attempt to convert it to a valid directory name. Setup will not fail if a conversion can not be made, but the string represented by the LDID probably will not remain intact.

  The only time that you are safe in using the contents of an LDID under NT 4.0 is when the LDID is assigned to a valid directory name. Attempting to set the LDID to a registry branch (first bit of flag = 0) will rarely produce the expected result.

AdvPack User Destination

User Choice of Destination Directory

This is an example of how to use Advanced INF Installer to prompt the user to specify a destination directory for your application. If the MS Office registry key and value name exist, and the registry key value is a valid directory on the user's system, Advanced Pack displays a dialog box asking the user if he would like to install in the office directory.

Note:
The flag value used is 1. To not prompt the user to accept the default destination, a flag value of 5 can be used instead.

[CustomDestinationSection]
49000 = CustomLDIDSection,1

[CustomLDIDSection]
"HKLM","SOFTWARE\Microsoft\Microsoft Office\95\InstallRoot","","Would you like to install this template in your office directory?","C:\MSOFFFICE"

Begin Prompt

```
//***************************************************************************
//*     Copyright (c) Microsoft Corporation 1995-1998. All rights reserved. *
//***************************************************************************
//*                                                                         *
//* ADVPUB.H - Specify the Interface for ADVPACK.DLL                        *
//*                                                                         *
//***************************************************************************


#ifndef _ADVPUB_H_
#define _ADVPUB_H_

#ifdef __cplusplus
extern "C" {
#endif /* __cplusplus */

////////////////////////////////////////////////////////////////////////////
// ENTRY POINT: RunSetupCommand
//
// SYNOPSIS:    Execute an install section in an INF file, or execute a
//              program.  Advanced INF files are supported.
//
// RETURN CODES:
//
//      S_OK                            Everything OK, no reboot needed.
//                                      No EXE to wait for.
//      S_ASYNCHRONOUS                  Please wait on phEXE.
//      ERROR_SUCCESS_REBOOT_REQUIRED   Reboot required.
//      E_INVALIDARG                    NULL specified in szCmdName or szDir
//      HRESULT_FROM_WIN32(ERROR_OLD_WIN_VERSION) INF's not supported on this OS version
//      E_UNEXPECTED                    Catastrophic failure(should never happen).
//      HRESULT_FROM_WIN32(GetLastError())   Anything else
////////////////////////////////////////////////////////////////////////////

#ifndef S_ASYNCHRONOUS
#define S_ASYNCHRONOUS  _HRESULT_TYPEDEF_(0x401e8L)
#endif

#define achRUNSETUPCOMMANDFUNCTION   "RunSetupCommand"

HRESULT WINAPI RunSetupCommand( HWND hWnd, LPCSTR szCmdName,
                        LPCSTR szInfSection, LPCSTR szDir,
                        LPCSTR lpszTitle, HANDLE *phEXE,
                        DWORD dwFlags, LPVOID pvReserved );

typedef HRESULT (WINAPI *RUNSETUPCOMMAND)(
    HWND    hWnd,                   // Handle to parent window  NULL=Quiet mode
    LPCSTR  szCmdName,              // Inf or EXE filename to "run"
```

```
    LPCSTR  szInfSection,           // Inf section to install.  NULL="DefaultInstall"
    LPCSTR  szDir,                  // Path to extracted files
    LPCSTR  szTitle,                // Title for all dialogs
    HANDLE *phEXE,                   // Handle to EXE to wait for
    DWORD   dwFlags,                // Flags to specify functionality (see above)
    LPVOID  pvReserved               // Reserved for future use
);

// FLAGS:

#define RSC_FLAG_INF              1   // exxcute INF install
#define RSC_FLAG_SKIPDISKSPACECHECK 2   // Currently does nothing
#define RSC_FLAG_QUIET            4   // quiet mode, no UI
#define RSC_FLAG_NGCONV           8   // don't run groupConv
#define RSC_FLAG_UPDHLPDLLS       16  // force to self-updating on user's system
#define RSC_FLAG_DELAYREGISTEROCX  512  // force delay of ocx registration
#define RSC_FLAG_SETUPAPI         1024  // use setupapi.dll

// please not adding flag after this.  See LaunchINFSectionEx() flags.

//////////////////////////////////////////////////////////////////////////
// ENTRY POINT: NeedRebootInit
//
// SYNOPSIS:    Initializes state for reboot checking.  Call this function
//              before calling RunSetupCommand.
// RETURNS:     value required to be passed to NeedReboot()
//////////////////////////////////////////////////////////////////////////

#define achNEEDREBOOTINITFUNCTION   "NeedRebootInit"

DWORD WINAPI NeedRebootInit( VOID );

typedef DWORD (WINAPI *NEEDREBOOTINIT)(VOID);

//////////////////////////////////////////////////////////////////////////
// ENTRY POINT: NeedReboot
//
// SYNOPSIS:    Compares stored state with current state to determine if a
//              reboot is required.
//      dwRebootCheck   the return value from NeedRebootInit
//
// RETURNS:
//      TRUE            if a reboot is required;
//      FALSE           otherwise.
//////////////////////////////////////////////////////////////////////////

#define achNEEDREBOOTFUNCTION   "NeedReboot"

BOOL WINAPI NeedReboot( DWORD dwRebootCheck );
```

```
typedef BOOL (WINAPI *NEEDREBOOT)(
    DWORD dwRebootCheck                         // Value returned from NeedRebootInit
);

//////////////////////////////////////////////////////////////////////////
// ENTRY POINT: DoReboot
//
// SYNOPSIS:    Ask advpack to do reboot.
//     hwnd       if it is INVALID_HANDLE_VALUE, no user prompt.  Otherwise promp.
//     pszTitle   User prompt UI title string.
//     dwReserved  Not used.
// RETURNS:
//     FALSE      User choose NO to reboot prompt.
//////////////////////////////////////////////////////////////////////////

// #define achDOREBOOT "DoReboot"

// BOOL WINAPI DoReboot( HWND hwnd, BOOL bDoUI );
// typedef BOOL (WINAPI* DOREBOOT)( HWND hwnd, BOOL bDoUI );

//////////////////////////////////////////////////////////////////////////
// ENTRY POINT: RebootCheckOnInstall
//
// SYNOPSIS:    Check reboot condition if the given INF section is installed.
//     hwnd    windows handle
//     pszINF  INF filename with fully qualified path
//     pszSec  INF section.  NULL is translated as DefaultInstall or DefaultInstall.NT.
//     dwReserved Not used.
// RETURN:
//     S_OK    Reboot needed if INF section is installed.
//     S_FALSE Reboot is not needed if INF section is installed.
//     HRESULT of Win 32 errors
//
//////////////////////////////////////////////////////////////////////////

#define achPRECHECKREBOOT   "RebootCheckOnInstall"

HRESULT WINAPI RebootCheckOnInstall( HWND hwnd, PCSTR pszINF, PCSTR pszSec, DWORD dwReserved );

typedef HRESULT (WINAPI *REBOOTCHECKONINSTALL)( HWND, PCSTR, PCSTR, DWORD );

//////////////////////////////////////////////////////////////////////
// ENTRY POINT: TranslateInfString
//
// SYNOPSIS:    Translates a key value in an INF file, using advanced INF
//              syntax.
// RETURN CODES:
//     S_OK                        Everything OK.
```

```
//     HRESULT_FROM_WIN32(ERROR_INSUFFICIENT_BUFFER)
//                             The buffer size is too small to hold the
//                             translated string.  Required size is in *pdwRequiredSize.
//     E_INVALIDARG                    NULL specified in pszInfFilename, pszTranslateSection,
//                             pszTranslateKey, pdwRequiredSize.
//     HRESULT_FROM_WIN32(ERROR_OLD_WIN_VERSION)
//                             OS not supported.
//     E_UNEXPECTED                    Catastrophic failure -- should never happen.
//     HRESULT_FROM_WIN32(ERROR_INVALID_PARAMETER)
//                             The section or key specified does not exist.
//     HRESULT_FROM_WIN32(GetLastError())   Anything else
//
////////////////////////////////////////////////////////////////////////////

#define c_szTRANSLATEINFSTRING "TranslateInfString"

HRESULT WINAPI TranslateInfString( PCSTR pszInfFilename, PCSTR pszInstallSection,
                        PCSTR pszTranslateSection, PCSTR pszTranslateKey,
                        PSTR pszBuffer, DWORD dwBufferSize,
                        PDWORD pdwRequiredSize, PVOID pvReserved );

typedef HRESULT (WINAPI *TRANSLATEINFSTRING)(
   PCSTR  pszInfFilename,             // Name of INF file to process
   PCSTR  pszInstallSection,          // Install section name (NULL=DefaultInstall)
   PCSTR  pszTranslateSection,        // Section that contains key to translate
   PCSTR  pszTranslateKey,            // Key to translate
   PSTR   pszBuffer,                  // Buffer to store translated key.  (NULL=return required size only)
   DWORD  dwBufferSize,               // Size of this buffer.  If pszBuffer==NULL, this is ignored.
   PDWORD pdwRequiredSize,            // Required size of buffer
   PVOID  pvReserved                  // Reserved for future use
);

////////////////////////////////////////////////////////////////////////////
// ENTRY POINT: RegInstall
//
// SYNOPSIS:    Loads an INF from a string resource, adds some entries to the
//              INF string substitution table, and executes the INF.
// RETURNS:
//     S_OK    success.
//     E_FAIL  failure,
////////////////////////////////////////////////////////////////////////////

#define achREGINSTALL   "RegInstall"

typedef struct _StrEntry {
   LPSTR   pszName;           // String to substitute
   LPSTR   pszValue;          // Replacement string or string resource
} STRENTRY, *LPSTRENTRY;
```

```
typedef const STRENTRY CSTRENTRY;
typedef CSTRENTRY *LPCSTRENTRY;

typedef struct _StrTable {
   DWORD       cEntries;       // Number of entries in the table
   LPSTRENTRY  pse;            // Array of entries
} STRTABLE, *LPSTRTABLE;

typedef const STRTABLE CSTRTABLE;
typedef CSTRTABLE *LPCSTRTABLE;

HRESULT WINAPI RegInstall( HMODULE hm, LPCSTR pszSection, LPCSTRTABLE pstTable );

typedef HRESULT (WINAPI *REGINSTALL)(
   HMODULE hm,                  // Module that contains REGINST resource
   LPCSTR pszSection,           // Section of INF to execute
   LPCSTRTABLE pstTable         // Additional string substitutions
);


//////////////////////////////////////////////////////////////////////////
// ENTRY POINT: LaunchINFSectionEx
//
// SYNOPSIS:    Install INF section with BACKUP/ROLLBACK capabilities.
//
// RETURNS:     E_FAIL on failure, S_OK on success.
//////////////////////////////////////////////////////////////////////////

#define achLAUNCHINFSECTIONEX   "LaunchINFSectionEx"

HRESULT WINAPI LaunchINFSectionEx( HWND hwnd, HINSTANCE hInstance, PSTR pszParms, INT nShow );

typedef HRESULT (WINAPI *LAUNCHINFSECTIONEX)(
   HWND     hwnd,               // pass in window handle
   HINSTANCE hInst,             // instance handle
   PSTR     pszParms,           // String contains params: INF,section,CAB,flags
   INT      nShow
);

// FLAGS:
// FLAGS value this way is for compatibility. Don't change them.
//
#define ALINF_QUIET             4     // quiet mode, no UI
#define ALINF_NGCONV            8     // don't run groupConv
#define ALINF_UPDHLPDLLS        16    // force to self-updating on user's system
#define ALINF_BKINSTALL         32    // backup data before install
#define ALINF_ROLLBACK          64    // rollback to previous state
#define ALINF_CHECKBKDATA       128   // validate the backup data
#define ALINF_ROLLBKDOALL       256   // bypass building file list
```

```
#define ALINF_DELAYREGISTEROCX   512    // force delay of ocx registration


//////////////////////////////////////////////////////////////////////////
// ENTRY POINT: ExecuteCab
//
// SYNOPSIS:    Extract the an INF from the CAB file, and do INF install on it.
//////////////////////////////////////////////////////////////////////////

// RETURNS: E_FAIL on failure, S_OK on success.

#define achEXECUTECAB   "ExecuteCab"

typedef struct _CabInfo {
    PSTR  pszCab;
    PSTR  pszInf;
    PSTR  pszSection;
    char  szSrcPath[MAX_PATH];
    DWORD dwFlags;
} CABINFO, *PCABINFO;

HRESULT WINAPI ExecuteCab( HWND hwnd, PCABINFO pCab, LPVOID pReserved );

typedef HRESULT (WINAPI *EXECUTECAB)(
    HWND     hwnd,
    PCABINFO pCab,
    LPVOID   pReserved
);

// flag as LaunchINFSectionEx's flag defines

//////////////////////////////////////////////////////////////////////////
// ENTRY POINT: AdvInstallFile
//
// SYNOPSIS:    To copy a file from the source to a destination
//              Basicly a wrapper around the setupapi file copy engine
//////////////////////////////////////////////////////////////////////////

// Flags which can be passed to AdvInstallFile
// Here is a copy of the flags defined in setupapi.h for reference below.
//#define COPYFLG_WARN_IF_SKIP          0x00000001   // warn if user tries to skip file
//#define COPYFLG_NOSKIP                0x00000002   // disallow skipping this file
//#define COPYFLG_NOVERSIONCHECK        0x00000004   // ignore versions and overwrite target
//#define COPYFLG_FORCE_FILE_IN_USE     0x00000008   // force file-in-use behavior
//#define COPYFLG_NO_OVERWRITE          0x00000010   // do not copy if file exists on target
//#define COPYFLG_NO_VERSION_DIALOG     0x00000020   // do not copy if target is newer
//#define COPYFLG_REPLACEONLY           0x00000400   // copy only if file exists on target

#define AIF_WARNIFSKIP          0x00000001              // system critical file: warn if user tries to skip
```

```
#define AIF_NOSKIP              0x00000002              // Skip is disallowed for this file
#define AIF_NOVERSIONCHECK      0x00000004              // don't check the version number of the file overwrite
#define AIF_FORCE_FILE_IN_USE   0x00000008              // force file-in-use behavior
#define AIF_NOOVERWRITE         0x00000010              // copy only if target doesn't exist
                                                        // if AIF_QUIET, the file is not copied and
                                                        // the user is not notified
#define AIF_NO_VERSION_DIALOG   0x00000020              // do not copy if target is newer
#define AIF_REPLACEONLY         0x00000400              // copy only if target file already present

// Flags only known to AdvInstallFile
#define AIF_NOLANGUAGECHECK     0x10000000              // don't check the language of the file
                                                        // if the flags is NOT specified and AIF_QUIET
                                                        // the file is not copied and the user is not notified
#define AIF_QUIET               0x20000000              // No UI to the user


#define achADVINSTALLFILE   "AdvInstallFile"

HRESULT WINAPI AdvInstallFile(HWND hwnd, LPCSTR lpszSourceDir, LPCSTR lpszSourceFile,
                        LPCSTR lpszDestDir, LPCSTR lpszDestFile, DWORD dwFlags, DWORD dwReserved);

typedef HRESULT (WINAPI *ADVINSTALLFILE)(
                                HWND hwnd,              // Parent Window for messages
                                LPCSTR lpszSourceDir,       // Source directory (does not contain filename)
                                LPCSTR lpszSourceFile,      // Filename only
                                LPCSTR lpszDestDir,         // Destination directory (does not contain filename)
                                LPCSTR lpszDestFile,        // optional filename. if NULL lpszSourceFile is used
                                DWORD dwFlags,              // AIF_* FLAGS
                                DWORD dwReserved);

//////////////////////////////////////////////////////////////////
//
//////////////////////////////////////////////////////////////////
// the following flags are for backwards compatiable.  No API user
// should reference them directly now.
//
#define  IE4_RESTORE        0x00000001      // if this bit is off, save the registries.
#define  IE4_BACKNEW        0x00000002      // backup all files which are not backed up before
#define  IE4_NODELETENEW    0x00000004      // don't delete files we don't backed up before
#define  IE4_NOMESSAGES     0x00000008      // No message display in any events.
#define  IE4_NOPROGRESS     0x00000010      // this bit on: No file backup progressbar
#define  IE4_NOENUMKEY      0x00000020      // this bit on: Don't Enum sub key even there is no given valuename
#define  IE4_NO_CRC_MAPPING 0x00000040      // Normally you should not turn on this bit, advpack creates
                                // internal mapping for all the entries backed up.
#define  IE4_REGSECTION     0x00000080      // INF AddReg/DelReg section
#define  IE4_FRDOALL        0x00000100      // FileRestore DoAll
#define  IE4_UPDREFCNT      0x00000200      // Update the ref count in .ini backup file list
#define  IE4_USEREFCNT      0x00000400      // use ref count to determin if the backup file should be put back
#define  IE4_EXTRAINCREFCNT 0x00000800      // if increase the ref cnt if it has been updated before
```

```
//////////////////////////////////////////////////////////////////////////
// ENTRY POINT: RegSaveRestore
//
// SYNOPSIS:    Save or Restore the given register value or given INF reg section.
//
// RETURNS:     E_FAIL on failure, S_OK on success.
//////////////////////////////////////////////////////////////////////////

// Save or Restore the given register value
HRESULT WINAPI RegSaveRestore(HWND hWnd, PCSTR pszTitleString, HKEY hkBckupKey, PCSTR pcszRootKey, PCSTR pcszSubKey, PCSTR pcszValueName, DWORD dwFlags);

typedef HRESULT (WINAPI *REGSAVERESTORE)( HWND hWnd,
                                PCSTR pszTitleString,  // user specified UI title
                                HKEY hkBckupKey,       // opened Key handle to store the backup data
                                PCSTR pcszRootKey,     // RootKey string
                                PCSTR pcszSubKey,      // SubKey string
                                PCSTR pcszValueName,   // Value name string
                                DWORD dwFlags);        // Flags

// Save or Restore the given INF Reg Section. At restore, if INF and Section pointers are NULL,
// Restore all from the given backup key handle.
HRESULT WINAPI RegSaveRestoreOnINF( HWND hWnd, PCSTR pszTitle, PCSTR pszINF,
                            PCSTR pszSection, HKEY hHKLMBackKey, HKEY hHKCUBackKey, DWORD dwFlags );

typedef HRESULT (WINAPI *REGSAVERESTOREONINF)( HWND hWnd,
                                    PCSTR pszTitle,        // user specified UI title
                                    PCSTR pszINF,          // INF filename with fully qualified path
                                    PCSTR pszSection,      // INF section name.  NULL == default
                                    HKEY hHKLMBackKey,     // openned key handle to store the data
                                    HKEY hHKCUBackKey,     // openned key handle to store the data
                                    DWORD dwFlags );       // Flags

// FLAG:
#define ARSR_RESTORE    IE4_RESTORE       // if this bit is off, means Save. Otherwise, restore.
#define ARSR_NOMESSAGES IE4_NOMESSAGES    // Quiet no messages in any event.
#define ARSR_REGSECTION IE4_REGSECTION    // if this bit is off, the given section is GenInstall Section

// Turn on the logging by add these RegVale in HKLM\software\microsoft\IE4
#define  REG_SAVE_LOG_KEY    "RegSaveLogFile"
#define  REG_RESTORE_LOG_KEY "RegRestoreLogFile"

// for backwards compatible add this one back
HRESULT WINAPI RegRestoreAll(HWND hWnd, PSTR pszTitleString, HKEY hkBckupKey);
typedef HRESULT (WINAPI *REGRESTOREALL)(HWND hWnd, PSTR pszTitleString, HKEY hkBckupKey);
//////////////////////////////////////////////////////////////////////////
// ENTRY POINT: FileSaveRestore
//
// SYNOPSIS:    Save or Restore the files on the list lpFileList.
```

```
//            If lpFileList is NULL at restore time, the function will restore
//            all based on INI index file.
//
// RETURNS:     E_FAIL on failure, S_OK on success.
//////////////////////////////////////////////////////////////////////////////

HRESULT WINAPI FileSaveRestore( HWND hDlg, LPSTR lpFileList, LPSTR lpDir, LPSTR lpBaseName, DWORD dwFlags);

typedef HRESULT (WINAPI *FILESAVERESTORE)( HWND hDlg,
                                LPSTR lpFileList,    // File list file1\0file2\0filen\0\0
                                LPSTR lpDir,         // pathname of the backup directory
                                LPSTR lpBaseName,    // backup file basename
                                DWORD dwFlags);      // Flags

HRESULT WINAPI FileSaveRestoreOnINF( HWND hWnd, PCSTR pszTitle, PCSTR pszINF,
                            PCSTR pszSection, PCSTR pszBackupDir, PCSTR pszBaseBackupFile,
                            DWORD dwFlags );

typedef HRESULT (WINAPI *FILESAVERESTOREONINF)( HWND hDlg,
                                    PCSTR pszTitle,       // user specified UI title
                                    PCSTR pszINF,         // INF filename with fully qualified path
                                    PCSTR pszSection,     // GenInstall INF section name.  NULL == default
                                    PCSTR pszBackupDir,   // directory to store the backup file
                                    PCSTR pszBaseBackFile, // Basename of the backup data files
                                    DWORD dwFlags );      // Flags


// FLAGS:
#define  AFSR_RESTORE       IE4_RESTORE      // if this bit is off, save the file.
#define  AFSR_BACKNEW       IE4_BACKNEW      // backup all files which are not backed up before
#define  AFSR_NODELETENEW    IE4_NODELETENEW  // don't delete files we don't backed up before
#define  AFSR_NOMESSAGES     IE4_NOMESSAGES   // No message display in any events.
#define  AFSR_NOPROGRESS     IE4_NOPROGRESS   // this bit on: No file backup progressbar
#define  AFSR_UPDREFCNT      IE4_UPDREFCNT    // update the reference count for the files
#define  AFSR_USEREFCNT      IE4_USEREFCNT    // use the ref count to guide the restore file
#define  AFSR_EXTRAINCREFCNT IE4_EXTRAINCREFCNT

//////////////////////////////////////////////////////////////////////////////
// ENTRY POINT: AddDelBackupEntry
//
// SYNOPSIS:    If AADBE_ADD_ENTRY is specified, mark the file in the File list as not existing
//            during file save in the INI file.  This can be used to mark additional files that
//            they did not exist during backup to avoid having them backup the next time the
//            FileSaveRestore is called to save files.
//            If AADBE_DEL_ENTRY is specified, delete the entry from the INI.  This mechanism can
//            be used to leave files permanently on the system.
//
// RETURNS:
//      S_OK    success
```

```
//      E_FAIL  failure
////////////////////////////////////////////////////////////////////////////
HRESULT WINAPI AddDelBackupEntry(LPCSTR lpcszFileList, LPCSTR lpcszBackupDir, LPCSTR lpcszBaseName, DWORD dwFlags);

typedef HRESULT (WINAPI *ADDDELBACKUPENTRY)(LPCSTR lpcszFileList,   // File list file1\0file2\0filen\0\0
                                LPCSTR lpcszBackupDir,   // pathname of the backup directory
                                LPCSTR lpcszBaseName,    // backup file basename
                                DWORD  dwFlags);

#define  AADBE_ADD_ENTRY    0x01          // add entries to the INI file
#define  AADBE_DEL_ENTRY    0x02          // delete entries from the INI file

////////////////////////////////////////////////////////////////////////////
// ENTRY POINT: FileSaveMarkNotExist
//
// SYNOPSIS:    Mark the file in the File list as not existing during file save in the INI file
//              This can be used to mark additional files that they did not exist during backup
//              to avoid having them backup the next time the FileSaveRestore is called to save
//              files
//
// RETURNS:
//      S_OK    success
//      E_FAIL  failure
////////////////////////////////////////////////////////////////////////////
HRESULT WINAPI FileSaveMarkNotExist( LPSTR lpFileList, LPSTR lpDir, LPSTR lpBaseName);

typedef HRESULT (WINAPI *FILESAVEMARKNOTEXIST)( LPSTR lpFileList,    // File list file1\0file2\0filen\0\0
                                LPSTR lpDir,        // pathname of the backup directory
                                LPSTR lpBaseName);    // backup file basename

////////////////////////////////////////////////////////////////////////////
// ENTRY POINT: GetVersionFromFile
//
// SYNOPSIS:    Get the given file's version and lang information.
//
// RETURNS:     E_FAIL on failure, S_OK on success.
////////////////////////////////////////////////////////////////////////////

HRESULT WINAPI GetVersionFromFile(LPSTR lpszFilename, LPDWORD pdwMSVer, LPDWORD pdwLSVer, BOOL bVersion);

typedef HRESULT (WINAPI *GETVERSIONFROMFILE)(
                                LPSTR lpszFilename,         // filename to get info from
                                LPDWORD pdwMSVer,           // Receive Major version
                                LPDWORD pdwLSVer,           // Receive Minor version
                                BOOL bVersion);             // if FALSE, pdwMSVer receive lang ID
                                                            // pdwLSVer receive Codepage ID

////////////////////////////////////////////////////////////////////////////
// ENTRY POINT: IsNTAdmin
```

```
//
// SYNOPSIS:    On NT, check if user has admin right.
//
// RETURNS:     TURE  has admin right; FLSE  no admin right.
////////////////////////////////////////////////////////////////////////////

#define achISNTADMIN "IsNTAdmin"

BOOL WINAPI IsNTAdmin( DWORD dwReserved, DWORD *lpdwReserved );

typedef BOOL (WINAPI *ISNTADMIN)( DWORD,        // not used
                          DWORD * );    // not used

////////////////////////////////////////////////////////////////////////////
// ENTRY POINT: DelNode
//
// SYNOPSIS:    Deletes a file or directory
//
// RETURNS:
//      S_OK    success
//      E_FAIL  failure
////////////////////////////////////////////////////////////////////////////

// FLAGS:
#define ADN_DEL_IF_EMPTY        0x00000001  // delete the directory only if it's empty
#define ADN_DONT_DEL_SUBDIRS    0x00000002  // don't delete any sub-dirs; delete only the files
#define ADN_DONT_DEL_DIR        0x00000004  // don't delete the dir itself

#define achDELNODE              "DelNode"

HRESULT WINAPI DelNode(LPCSTR pszFileOrDirName, DWORD dwFlags);

typedef HRESULT (WINAPI *DELNODE)(
    LPCSTR pszFileOrDirName,                // Name of file or directory to delete
    DWORD dwFlags                           // 0, ADN_DEL_IF_EMPTY, etc. can be specified
);

////////////////////////////////////////////////////////////////////////////
// ENTRY POINT: DelNodeRunDLL32
//
// SYNOPSIS:    Deletes a file or directory; the parameters to this API are of
//              WinMain type
//
// RETURNS:
//      S_OK    success
//      E_FAIL  failure
////////////////////////////////////////////////////////////////////////////

#define achDELNODERUNDLL32      "DelNodeRunDLL32"
```

```
HRESULT WINAPI DelNodeRunDLL32(HWND hwnd, HINSTANCE hInstance, PSTR pszParms, INT nShow);

typedef HRESULT (WINAPI *DELNODERUNDLL32)(
   HWND     hwnd,                    // pass in window handle
   HINSTANCE hInst,                     // instance handle
   PSTR     pszParams,               // String contains params: FileOrDirName,Flags
   INT      nShow
);

//////////////////////////////////////////////////////////////////////////
// ENTRY POINT: OpenINFEngine, TranslateINFStringEx, CloseINFEngine
//
// SYNOPSIS:    Three APIs give the caller the option to be more efficient when need
//              Advpack to translate INF file in a continue fashion.
//
// RETURNS:
//      S_OK    success
//      E_FAIL  failure
//////////////////////////////////////////////////////////////////////////

//
// Define type for reference to loaded inf file
// (from setupapi.h)
typedef PVOID HINF;


HRESULT WINAPI OpenINFEngine( PCSTR pszInfFilename, PCSTR pszInstallSection,
                        DWORD dwFlags, HINF *phInf, PVOID pvReserved );

HRESULT WINAPI TranslateInfStringEx( HINF hInf, PCSTR pszInfFilename,
                              PCSTR pszTranslateSection, PCSTR pszTranslateKey,
                              PSTR pszBuffer, DWORD dwBufferSize,
                              PDWORD pdwRequiredSize, PVOID pvReserved );

HRESULT WINAPI CloseINFEngine( HINF hInf );



HRESULT WINAPI ExtractFiles( LPCSTR pszCabName, LPCSTR pszExpandDir, DWORD dwFlags,
                         LPCSTR pszFileList, LPVOID lpReserved, DWORD dwReserved);

//////////////////////////////////////////////////////////////////////////
// ENTRY POINT: LaunchINFSection
//
// SYNOPSIS:    Install INF section WITHOUT BACKUP/ROLLBACK capabilities.
//
// RETURNS:     E_FAIL on failure, S_OK on success.
//////////////////////////////////////////////////////////////////////////
```

```
INT     WINAPI LaunchINFSection( HWND, HINSTANCE, PSTR, INT );

// LaunchINFSection flags
#define LIS_QUIET               0x0001      // Bit 0
#define LIS_NOGRPCONV           0x0002      // Bit 1

// Flags in Advanced INF RunPreSetupCommands and RunPostSetupCommands of the Install section
// Those flags can tell advpack how to run those commands, quiet or not quiet, wait or not wait.
// The Default for runing those commands are:  Not Quiet and Wait for finish before return the caller.
// I.E>  RunPostSetupCommands = MyCmdsSecA:1, MyCmdsSecB:2, MyCmdsSecC
//
#define RUNCMDS_QUIET           0x00000001
#define RUNCMDS_NOWAIT          0x00000002
#define RUNCMDS_DELAYPOSTCMD    0x00000004

// Active Setup Installed Components GUID for IE4
#define awchMSIE4GUID L"{89820200-ECBD-11cf-8B85-00AA005B4383}"

//////////////////////////////////////////////////////////////////////////////
//
//////////////////////////////////////////////////////////////////////////////

#ifdef __cplusplus
}
#endif /* __cplusplus */

#endif // _ADVPUB_H_
```

AdvPack Registry Destination

Handling Directories for International Installations

This Advanced INF Installer section checks for a registry key associated with Office 95. If this keys does not exist, installation will fail and the user will be see the message in the last line of the section.

Also when creating directories or prompting users for input you'll want to use a mixture of OEM and ANSI characters. The prompts should also use the Strings section.

[CustomDestinationSection]
49000 = CustomLDIDSection , 22 (Original line)

(When handling International installations.)

49000,49001,49002,49003== CustomLDIDSection , 22 (New line)

<OEM SFN>, <ANSI LFN>, <ANSI SFN>, <OEM LFN>= <Section>,Flag

[CustomLDIDSection]
"HKLM","SOFTWARE\Microsoft\Microsoft Office\95\InstallRoot","dir","%PromptUser%","%DefaultDir%"

[Strings]

DefaultDir="C:\MSOFFICE"

PromptUser="Is this where you want Office to go?"

In the above example if you need to use 49000 when setting the LDID for a copy section. You might want to use 49001 when storing the value in the registry for later use. That way the LFN is what the user would really see in the registry, but file system in Windows 95 can use SFN for INF. (Windows NT 4.0 would not need to use the SFN in any install.)

AdvPack Application Check

Checking For Existing Applications Before Installation

This Advanced INF Installer section checks for two registry keys: one associated with Office 95 and one with IExpress. If none of these keys exist, installation will fail and the user will be see the message in the last line of the section. To make sure more than one registry key is present, use two CustomDestination sections, one for each key.

```
[CustomDestinationSection]
49000 = CustomLDIDSection , 22

[CustomLDIDSection]
"HKLM","SOFTWARE\Microsoft\Microsoft Office\95\InstallRoot","","",""
"HKLM","SOFTWARE\Microsoft\IExpress","InstallDir","",""
"","","","You are missing software required for upgrade!",""
```
