Advanced Package File Options

This article needs to be updated: If you found this article useful, please fix the problems below then delete the {{ArticleNeedsUpdate}} template from the article to remove this warning.

Reasons: hamishwillee
(30 Jan 2012)
I don't see any value of this article over the content in the Developer library. Propose this simply links to the developer library, or is deleted.

Text Notices

You can specify a text file to be displayed to the user during the installation process. The file itself is not copied to the target. This is useful for displaying basic readme information or license agreements. To do this, add the line below to the pkg file:

```
"license.txt" - "", FILETEXT, TEXTCONTINUE
```

FILETEXT indicates to display the file during install. TEXTCONTINUE will provide a continue button that will dismiss the text file and continue the installation.

Instead of **TEXTCONTINUE**, you can specify one of the following:

- **TEXTSKIP** displays a Yes/No option. If Yes is selected, installation continues. If No is selected, the next statement is skipped, but installation continues normally afterwards.
- **TEXTEXIT** displays a Yes/No option. If Yes is selected, installation continues. If No is selected, the installation stops and any files that have already been installed are removed.
- **TEXTABORT** displays a Yes/No option, but when No is selected, the installation just stops, without removing any installed files. Removing Runtime-Generated Files.

Multi Language Text Notices

The language of text notices can be customized depending on the device's default language. For example:

```
IF  (LANGUAGE=2) ; If the language is French, then display mytext.t02
   "mytext.t02"-"", FILETEXT, TEXTABORT
ELSEIF (LANGUAGE=3) ; If the language is German, then display mytext.t03
   "mytext.t03"-"", FILETEXT, TEXTABORT
ELSE ; Otherwise display mytext.t01
   "mytext.t01"-"", FILETEXT, TEXTABORT
ENDIF
```

The example above will display `mytext.t02` when the default language on the device is French; `mytext.t03` when the language is German; and `mytext.t01` for any other languages.

BTW, you must have also the languages defined in PKG

```
;Language - standard language definitions
&EN,FR,GE
```

Removing Runtime-Generated Files

When you uninstall a program via its sis file, the uninstaller will remove all the files that were copied to the phone by the installer as specified in the file specification lines of the pkg file. But what if a file is generated at runtime?

If the names of the files to be created by the application are known at installation time, they must be added to the .pkg file as 'null files'. The format of the filename (language-independent files) in the .pkg file is as follows:

```
"" - "C\system\data\my_runtime_generated_file", FILENULL
```

where my_runtime_generated_file is a data file created by the application. FN (FILENULL) parameter means that a file does not yet exist, so it is not included in the sis file. It is created by the running application and will be deleted when the application is removed. Note that the name assigned to the source file is not important and should be empty (""). Also note that such files will not be deleted when upgrading to a later version. This ensures that files, such as .ini files, which store application preferences, are not lost in an upgrade.

The FILENULL option is deprecated in Symbian OS v9.x. Developers are advised to create the files in their application's data cage which will be removed automatically (and completely) on uninstall.

If the number of application-created files (or their exact names) is not known, the case is a bit more complicated. As a solution, the .sis package could include a specific uninstall application that takes care of finding and removing the right files at uninstallation. Such an application can be run automatically during uninstallation as follows:

```
"\epoc32\release\armi\urel\uninst.app"  - "!:\system\apps\myapp\uninst.app", RR, RW
```

RR (RUNREMOVE) parameter sets the file to be run at remove (uninstall) time. RW (RUNWAITEND) indicates that the (system) remover should start the program and then wait until it has completed before resuming the installation.

Embedding SIS Files

You can include another sis file within your sis file with the following line:

```
@"sis file name",(UID)
```

For example: @"prog1.sis",(0x12341234) installs prog1.sis, with UID 0x12341234, at the point where this line is encountered. Note that, on uninstallation, this embedded sis file will not be uninstalled until the system determines that no other currently installed components use it (i.e. there is no other installed component that also includes that sis file in its pkg file).

Running Executables on Install or Uninstall

Note: FILERUN will not work with Self-signed applications.

You can specify that an executable be run during an installation by adding **FILERUN (FR)**and **RUNINSTALL (RI)** keywords at the end of the executable's file specification line.

For example:

```
"\Symbian\9.1\S60_3rd_MR\Epoc32\release\gcce\urel\myprogram.exe"-"!:\sys\bin\myprogram.exe",FR,RI
```

will install myprogram.exe and execute it during the installation. The RUNINSTALL keyword can be replaced by either of the following alternatives:

- **RUNREMOVE (RR)** causes execution to occur only during uninstallation.
- **RUNBOTH (RB)**causes the executable to be run on both installation and uninstallation.

Any of these three options may be further qualified by use of the RUNWAITEND (RW) keyword, which causes the installation to wait for the executable to complete before continuing. If not specified, then installation continues immediately after the executable is launched.

Note: FILERUN (FR) RUNINSTALL (RI) options does not work with Self-Signed applications. It should be signed with a trusted certificate (Open Signed Online or Open Signed Offline during R&D stage and Symbian Signed when released) even if otherwise the capabilities required for the project do not justify it.

Requisite Lines

You can use a requisite line to specify that a particular component must already be installed in order for the current installation to continue. It has the following format:

```
{UID}, Major_Version_#,Minor_Version_#,Build_#,{“Product Name”}
```

This means that the component with the specified UID and Product Name, with a version number not earlier than the one specified, must exist for the installation to continue.

For example:

```
{0x10000123},1,0,0,{"MyDll"}
```

indicates that a component named MyDll, with UID 0x10000123 and a version number of at least 1.0.0 must exist already before installation can proceed. The requisite line should look familiar – it is how the target platform line is implemented. The example target platform line:

```
(0x101F6F88), 0, 0, 0, {"Series60ProductID"}
```

is a requisite statement that the 'component' named Series60ProductID, with a UID of 0x101F6F88, and version number 0.0.0 or higher, must exist in order for the installation to continue.

Starting from S60 3rd Edition, the S60 product identification has to be written in square bracket. For example:

```
[0x101F7961], 0, 0, 0, {"S60ProductID"}
```

Language Support

Multiple translated versions of an application can exist within a single sis file. When a user installs a sis file, they are prompted to select which language

they would like installed. To specify the language variants that you want to be included, add a language line at the top of your pkg file. The language line begins with '&' and contains a list of comma-separated language codes from the following (incomplete) list.

- AM– US English
- AS – Austrian German
- AU – Australian English
- BF – Belgian French
- BL – Belgian Flemish
- CS – Czech
- DA – Danish
- DU – Dutch
- EN – UK English
- FI – Finnish
- FR – French
- GE – German
- HK – Hong Kong Chinese
- HU – Hungarian
- IC – Icelandic
- IF – International French
- IT – Italian
- JA – Japanese
- NO – Norwegian
- NZ – New Zealand
- PL – Polish
- PO – Portuguese
- RO - Romanian
- RU – Russian
- SF – Swiss French
- SG – Swiss German
- SK – Slovak
- SL – Slovenian
- SP – Spanish
- SW – Swedish
- TC – Taiwan Chinese
- TH – Thai
- TU – Turkish
- ZH – Prc Chinese

An example language line is:

```
&EN, FR, FI
```

which specifies that the sis file contains English, French and Finnish language variants. If a language line is not included, &EN is assumed.

Limiting Device Support

It is possible to limit the installation of package file to certain devices only. For example, an application that uses WiFi should be installable on devices that have WiFi only. If the package file is installed on non-WiFi devices, the installer will display a warning saying that the application is not compatible.

The following examples show an example of package file that can only be installed on Nokia N80, N93, N93 and N95.

```
[0x200005F9], 0, 0, 0, {"Nokia N80 ID"}
[0x20000600], 0, 0, 0, {"Nokia N93 ID"}
[0x20000605], 0, 0, 0, {"Nokia N93i ID"}
[0x2000060B], 0, 0, 0, {"Nokia N95 ID"}
```

The list of device identification can be found on S60 Platform and device identification codes.

Installing Device-specific Files

It is possible to install files depending on the user's device. For example, a game developer may create a special DLL for devices that support 3D-accelerator. For devices with no 3D-accelerator, the package file should install standard DLL. This can be achieved with the following code:

```
IF (MachineUID=0x20000600) OR (MachineUID=0x2000060B)

  ; If the device is Nokia N93 or N95, then install files in this block.
  "\epoc32\release\armv5\urel\mydll_3d" -"!:\sys\bin\mydll.dll"

ELSE
```

```
   ; Otherwise install any files in this block.
   "\epoc32\release\armv5\urel\mydll" -"!:\sys\bin\mydll.dll"

ENDIF
```

The list of device identification can be found on S60 Platform and device identification codes.

Installing Platform-specific Files

It is possible to install files depending on the operating system. For example 3rd edition FP2 has dropped some Bluetooth libarires, so different files must be installed. Phone's platform can be obtained by checking files in z:\system\install folder:

```
if exists("z:\system\install\Series60v3.2.sis")
"\Epoc32\release\gcce\UREL\Common_0x123123123_fp2.dll"-"C:\sys\bin\common_0x123123123.dll"
else
"\Epoc32\release\gcce\UREL\Common_0x123123123.dll"-"C:\sys\bin\common_0x123123123.dll"
endif
```

Conditional Installation

The following functions can be used in Software Install PKG files:

- `package()` tests for the existence of a given installed package.
- `exists()` tests for the existence of a given file and
- `appprop()` retrieves the properties from another registry entry.
- `DevProp()` function may be used to query device capability values that are not provided directly by the named attributes.

`package()`: this returns true if a package with the specified UID is installed, false otherwise, so the following code installs foo.txt if a package exists with package UID 0x11223344.

```
IF package(0x11223344)
"foo.txt"-"c:\foo.txt"
ENDIF
```

`exists()`: this returns true if test.txt exists at specified location, the file exists.txt is installed.

```
if exists("c:\test.txt")
    "text\exists.txt"-"!:\private\0x01234567\exists.txt"
```

`appprop()`: the first parameter to appprop() is the package UID of the other package file in which the property is defined. The second parameter is the propid(property id) to retrieve. The following code tests the capabilities of another PKG file that it is dependent upon:

```
;Dependencies
(0x10000003), 1, 2, 3, {"Depend-EN"}

if (appprop(0x10000003,0) = 1)
   "text\appprop1.txt"-"!:\Documents\appprop1.txt"
elseif (appprop(0x10000003,0) = 2)
   "text\appprop2.txt"-"!:\Documents\appprop2.txt"
endif
```

`DevProp()`: One of the functions available for use within Software Install PKG files is the device capabilities function, devcap(). This function provides direct access to the HAL API, i.e., the HAL::Get() function. The DevProp() function may be used, within a condition block, to query device capability values that are not provided directly by the named attributes given above.

```
+(0=1,1=2,3=-1)

    if DevProp(1)
        "text\appprop1.txt"-"", FT, TC
    endif
```

Internal links

- S60 Platform and device identification codes.
- How to sign a .Sis file with Self-Sign Certificate.
- How to guide for creating/signing sis files.

External links

- Package file format (Developer Library)
- Full list of supported languages (Developer Library)