Archived:Backup of add-on DLLs on Symbian

Archived: This article is **archived** because it is not considered relevant for third-party developers creating commercial solutions today. If you think this article is still relevant, let us know by adding the template {{ReviewForRemovalFromArchive|user=~~~~|*write your reason here*}}.

Remove from Archive?: This article has been marked for removal from archive, for the following reasons:
This looks to still be true in current versions. Confirm?

Description

3rd party application developers need to enable backup and restore for their applications by including a `backup_registration.xml` file in the application's private directory. If the main application is designed so that its functionality can be later extended by installing add-on components (DLLs), they should be registered for backup as well.

Solution

If the UID of the DLL to be installed is known in beforehand, the backup registration of the DLL can be done by the main application package by using the following statement in the pkg file:

```
"DLL1BinaryBackup.xml"
     - "!:\private\10202D56\import\packages\<DLL1_UID>\backup_registration.xml"
```

In this case, the add-on DLL can be installed as a normal SISX package. Typically, a minimal system backup definition (taking backups of the binaries) is sufficient for the DLL.

The DLL installation can also be carried out with a partial upgrade (PU) SIS package over the existing application in case the UID of the DLL is not known. This upgrade package then defines the backup registration file, and either the add-on DLL directly or an embedded SISX file containing the binaries.

See File:DLLBackupExample.zip for sample .pkg definitions.

Backup of files generated by plug-ins

Apart from just backing up the binaries of the plug-in binaries, it might also be necessary for the user to backup the files generated by the plug-ins (for example, plug-in-specific settings).

Any application backup_registration*.xml present in the application's private directory indicates the files that need to be backed up. The backup_registration.xml is regarded as the main backup file of the application whereas other files only provide additional files not mentioned in the main backup file.

See File:BackupOfPluginComponents.zip for sample .pkg definitions.

MainApp.pkg: Package file of the main application with its own *backup_registration.xml* file.

MyPrivateFilesDLL1_Upgrade.pkg: Upgrade package for the main application. Includes the plug-in DLL1 as an embedded SIS.

The important line in the *MyPrivateFilesDLL1_Upgrade.pkg* file :-

```
   "<DLL1 Path>\DLL1PrivateFileBackup.xml"
       - "!:\private\<APP_UID>\backup_registration1.xml"
```

copies an additional file *backup_registration1.xml* to the private directory of the main application.

The *backup_registration1.xml* can define all additional files that need to be backed up. For example:

```
<include_file name="c:\private\<APP_UID>\backup\<DLL_UID>\DLL1File1.txt" />
<include_file name="c:\private\<APP_UID>\backup\<DLL_UID>\DLL1File2.dat" />
...
```

Any additional files to be backed up must either be in a public folder or under the main application's private folder structure.

Notes

The file name `"DLL1BinaryBackup.xml"` is only used to distinguish between the main application and the add-on DLL. When defining backup registration files in a .pkg file, the target name must always be in the following format: `backup_registration*.xml`.

Trying to update the main application's *backup_registration.xml* to contain additional files/folders with a PU-type SIS package has no effect.

See also

Archived:Enabling backup and restore for installed C++ applications