Executing Console Applications on Devices

Adding a Console Application to the S60 menu

Having an icon for a console application allows you to start the application on the mobile phone. Only a few steps are necessary to achieve this.

The name of the example application is “ConsoleMenu”, its UID3 is 0xE79A8F85. Whenever this appears in a source code segment, adapt this to your own application name and UID3.

1. Create a new \data\-folder in your project and add a file ConsoleMenu_reg.rss to this directory.

2. Paste the following text into ConsoleMenu_reg.rss:

```
#include <appinfo.rh>

UID2 KUidAppRegistrationResourceFile
UID3 0xE79A8F85              // Replace this with the UID3 of your application

RESOURCE APP_REGISTRATION_INFO
  {
  app_file="ConsoleMenu";    // The filename of your executable, without .exe
  }
```

3. Add the new resource file to the .mmp file, so that it’s included in the compilation-process. Do not change the "10003a3f" ID, this folder is used for all applications.

```
SOURCEPATH ..\data
START RESOURCE ConsoleMenu_reg.rss
    TARGETPATH \private\10003a3f\apps
END
```

4. Add the new resource file to the .pkg-file, in order to include it in the .sis that you will install on the phone:

```
"$(EPOCROOT)Epoc32\data\z\private\10003a3f\apps\ConsoleMenu_reg.rsc"-
"!:\private\10003a3f\import\apps\ConsoleMenu_reg.rsc"
```

That’s it!

Defining your own application icon

Right now, your application will have the default icon. If you want it to have your own icon, follow these additional steps.

1. Create an icon file in any vector graphics application that supports SVG-Tiny (e.g. Adobe Illustrator).

2. Create a \gfx\-folder in your project and save the icon file into this directory (e.g. ConsoleMenu.svg)

3. Add a file called Icons_scalable_dc.mk to the group-directory. Carbide.c++ should automatically suggest adding this new file to bld.inf. Allow this. If you do not use Carbide.c++, add the make file to bld.inf manually:

```
PRJ_MMPFILES
ConsoleMenu.mmp
gnumakefile Icons_scalable_dc.mk    // New line
```

4. Paste the following code into the Icons_scalable_dc.mk-file:

(C) Copyright Nokia 2014. All rights reserved.

```
ifeq (WINS,$(findstring WINS, $(PLATFORM)))
ZDIR=$(EPOCROOT)epoc32\release\$(PLATFORM)\$(CFG)\Z
else
ZDIR=$(EPOCROOT)epoc32\data\z
endif

TARGETDIR=$(ZDIR)\resource\apps
# Adapt the filename of the .mif-file that will be created:
ICONTARGETFILENAME=$(TARGETDIR)\ConsoleMenu.mif

ICONDIR=..\gfx

do_nothing :
 @rem do_nothing

MAKMAKE : do_nothing

BLD : do_nothing

CLEAN : do_nothing

LIB : do_nothing

CLEANLIB : do_nothing

RESOURCE : $(ICONTARGETFILENAME)

# Adapt the filename of the icon in the following lines!
$(ICONTARGETFILENAME) : $(ICONDIR)\ConsoleMenu.svg
 mifconv $(ICONTARGETFILENAME) \
  /c32 $(ICONDIR)\ConsoleMenu.svg

FREEZE : do_nothing

SAVESPACE : do_nothing

RELEASABLES :
 @echo $(ICONTARGETFILENAME)

FINAL : do_nothing
```

5. Create a new file called ConsoleMenu.rss in the /data/-folder and fill it with the following definition. Adapt the name of the short_caption and the caption to your needs – this will be the visible name of your application.

```
#include <appinfo.rh>

RESOURCE LOCALISABLE_APP_INFO r_localisable_app_info
    {
    short_caption = "ConsoleMenu";    // Short name of your application
    caption_and_icon =
    CAPTION_AND_ICON_INFO
        {
        caption = "ConsoleMenu";      // Normal name of your application
        number_of_icons = 1;
  icon_file = "\\resource\\apps\\ConsoleMenu.mif"; //.mif created by make file
  };
```


(C) Copyright Nokia 2014. All rights reserved.

```
    }
```

6. Add the new resource file to the .mmp-file, in order to include it in the compilation process:

```
SOURCEPATH ..\data
START RESOURCE ConsoleMenu.rss
    TARGETPATH \resource\apps
END
```

7. The ConsoleMenu_reg.rss-file has to be extended by one line defining the location of the localisable resource file:

```
#include <appinfo.rh>

UID2 KUidAppRegistrationResourceFile
UID3 0xE79A8F85

RESOURCE APP_REGISTRATION_INFO
  {
  app_file="ConsoleMenu";
  localisable_resource_file = "\\resource\\apps\\ConsoleMenu"; // New line!
  }
```

8. Of course, the new files also have to be added to the .pkg-file to include them into the installation package (.sis):

```
"$(EPOCROOT)Epoc32\data\z\resource\apps\ConsoleMenu.rsc"-"!:\resource\apps\ConsoleMenu.rsc"
"$(EPOCROOT)Epoc32\data\z\resource\apps\ConsoleMenu.mif"-"!:\resource\apps\ConsoleMenu.mif"
```

9. Clean the project and rebuild it.

That's it!

Usually, console applications will be rather small and for testing purposes only. If yours is larger and needs to be localized, you should not define the short_caption and the caption directly in the .rss-file, but instead externalize it to localizable files. For explanations on how to do this, take a look at the following pages in the Wiki: How to create application icon(SVG) in S60 3rd edition, How to define application icon.

(C) Copyright Nokia 2014. All rights reserved.