Building Symbian C++ Projects Using Qt SDK

This article explains how to use the Qt SDK to build and package native Symbian C++ projects.

Introduction

Most Symbian projects can be compiled without the actual Symbian SDK, using tools provided by the Qt SDK. Here are instructions on how to build a Symbian C++ project with the Symbian^3 Command Prompt delivered with the Qt SDK.

Note: These instructions were tested using the Qt SDK v1.1.3 Symbian^3 command prompt. The instructions may work on both earlier and later SDKs and targets, although this may require some "tweaking".

Building and Packaging a Symbian Project

Launch the Symbian^3 Command Prompt from the Windows start menu: Qt SDK > Symbian^3 Qt 4.7.3 (later version should work as well) > Qt 4.7.3 for Symbian^3 Command Prompt

Compiling

Go to the `group` directory of the Symbian project with the Symbian^3 Command Prompt. To build the project run the following commands:

```
bldmake bldfiles
abld build gcce urel
```

The build will probably fail with errors indicating that there are missing files. This occurs because the Qt SDK locates some Symbian platform files in different paths than the Symbian SDK.

The solution is to add the missing include paths into the project MMP file. There are two ways to this:

- Add `APP_LAYER_SYSTEMINCLUDE` to the beginning of the `.mmp` file
- Add the specific missing include paths as `SYSTEMINCLUDE`. For example, Sudokumaster game example for Symbian requires the following additional system includes in order to compile with Qt SDK:

```
SYSTEMINCLUDE \epoc32\include\mw
SYSTEMINCLUDE \epoc32\include\platform
SYSTEMINCLUDE \epoc32\include\platform\mw
```

After fixing the MMP paths to add the missing include paths, clean the build, regenerate makefiles, and build the project:

```
bldmake clean
bldmake bldfiles
abld build gcce urel
```

The project should now build the `gcce` release binary without errors. We still need to make a package to try it out.

Creating a SIS package

Go to the `sis` folder of the project using the Symbian^3 Command prompt. You cannot make the SIS package directly (with `makesis package.pkg`) since the paths are incorrect. Instead, you must run:

```
makesis -d%EPOCROOT% package.pkg
```

The EPOCROOT variable should point to the folder that contains the /epoc32 tree of your Qt installation. If this is not set you can do so by first running:

```
set EPOCROOT=\QtSDK\Symbian\SDKs\Symbian3Qt473\
```

(note, no need to specify the drive)

If everything goes well, you should now have an unsigned `.sis` file in your `sis` folder. It still needs to be signed in order to run.

Note: You can also do this by modifying your `.pkg` file; Replace the relative paths in the `.pkg` file from original '`epoc32\release\gcce\urel\`' with absolute paths pointing to the corresponding directory in your Qt SDK installation, for example '`C:\QtSDK\Symbian\SDKs\Symbian3Qt473\epoc32\release\gcce\urel\`'.

1. Replace relative paths pointing to `epoc32\...` with absolute paths.
2. Create the `.sis` file using the modified `.pkg` file by running the command: `makesis pkg_file_with_modified_paths.pkg`

Signing the SIS package

The easiest way to sign the package is to do it again using the command prompt. Full instructions on how to do so are provided in How to sign a .Sis file with Self-Sign Certificate; an abbreviated summary is given below:

1. Create key files for custom signing:

```
makekeys -cert -password World123 -len 1024 -dname "CN=World User OU=Development
OR=WorldCompany? CO=FI  EM=World@test.com" WorldKey.key WorldCert.cer
```

2. Sign your `.sis` file, using the keys you just created:

```
signsis yoursisfile.sis yoursignedsisfile.sis WorldCert.cer WorldKey.key
World123
```

You're done! Your project is now built, packaged into a signed `.sis` file, and ready to be installed on your Symbian device.

Summary

Building and packaging a native Symbian C++ project can be done using the Qt SDK since it contains all the necessary tools and software of the Symbian platform. Due to minor differences between the Symbian platform in the native Symbian SDK and Qt SDK, some tricks may be required. However, as the this article demonstrates, the required effort is not very substantial.