

SignHelper for Symbian OS

Developer's Guide

**This is the preview version.
You can purchase the full version of this document at
<http://www.maccent.com/solutions.shtml>**

Legal notice

Copyright© mAccent 2007. All rights reserved

SignHelper is a trademark of mAccent.

Symbian and Symbian OS are trademarks of Symbian Software Ltd

Nokia and S60 are registered trademarks of Nokia Corporation.

UIQ is a trademark of UIQ Technology AB.

Other product and company names mentioned herein may be trademarks or trade names of their respective owners.

Disclaimer

This document is designed to provide information about the subject matter covered. The document and the source code are intended for learning purposes only. Therefore, the texts should be used only as general guides and not as the ultimate sources of the subject matters covered.

The authors shall have neither liability nor responsibility to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the information covered in the document and the source code.

Table of Contents

This is the preview version.
You can purchase the full version of this document at
<http://www.maccent.com/solutions.shtml>

1. Introduction

1.1 Purpose and scope

This document is intended primarily for those developing and designing Symbian C++ applications, but it also contains information that might interest a much wider audience: engineering managers, product managers, software architects, technology strategists etc.

The document shows how applications for Symbian OS can be designed in the way which helps make the process of going through the Symbian Signed procedure more manageable, pay less for the signing, and reduce risk of having Symbian Signed on the critical path of your project.

In addition to this document you will find example applications with the source code. These applications can be installed on S60 3rd Edition¹ handsets as a proof-of-concept.

The document and example applications offer a completely new approach to Symbian Signed. You will not find information about it neither at <http://forum.nokia.com> nor at <http://www.symbiansigned.com>

It's not a hack of any sort though. You don't need to do anything special with your handset in order to install our example applications.

1.2 Prerequisites

To use the example applications provided with this document you will need the following:

1. S60 3rd Edition Platform SDK for Symbian OS (Maintenance Release) – for building example applications. The SDK can be found [here](#). Consult the SDK documentation for hardware and software requirements.
2. Nokia PC Suite – in order to be able to install the example applications on the device (you can find it, e.g., [here](#)).
3. S60 3rd Edition or S60 3rd Edition FP1 device ([look at this list](#)) with the SIM which can register in the network. Also you'll need some means of connecting the handset with Nokia PC Suite: USB cable or ability to connect over Bluetooth or Infrared.

You should be familiar with how to build applications for Symbian OS using the S60 3rd Edition Platform SDK.

¹ The approach described in this document is valid for any UI platform based on Symbian OS v9 onwards (which means S60 3rd Edition and UIQ3 onwards). The examples are developed for S60 3rd Edition Platform but can be easily modified to address another UI platform.

2. Briefly about Symbian Signed

We assume that you are already familiar with the security requirements of Symbian OS v9 and the Symbian Signed process. If you are not, you can find more information at <http://www.symbiansigned.com>. Briefly, from Symbian OS v9 onwards all applications must be signed with a certain certificate, otherwise they cannot be installed on a device. Type of certificate depends on “Capabilities” which are required by an application. Basically the Capabilities are required when an application uses some of so called protected (or sensitive) APIs. In order to find if the application requires any Capabilities you should consult the SDK (Symbian OS v9 and S60 3rd Edition or UIQ3 SDK) regarding APIs the application is using. If your application doesn’t need any Capabilities (Capability: None or no Capability statement at all) this means you can sign it with a self-signed certificate and you don’t need to go through the Symbian Signed process. You can start distributing your application even if it’s self-signed.

If your application does require some Capabilities you must enlist them in the project file (.mmp) and sign the application with the Developer certificate to be able to install it on S60 3rd Edition handset.

There are so-called Basic and Extended sets of Capabilities. The Basic set includes the following: LocalServices, UserEnvironment, NetworkServices, Location, ReadUserData, WriteUserData, the Extended set includes ReadDeviceData, WriteDeviceData, SwEvent, ProtServ, Power Mgmt, SurroundingDD, DRM, NetworkControl, MultimediaDD, TCB, AllFiles, CommDD, DiskAdmin, TrustedUI.¹

DRM, NetworkControl, MultimediaDD, TCB, AllFiles, CommDD, and DiskAdmin Capabilities also require the Phone Manufacturer Approval for signing and therefore applications requesting these must be submitted through the approved Phone Manufacturer channel.

The Developer certificate can be requested at <http://www.symbiansigned.com>. You must specify IMEIs (equipment identifiers) of the handsets you are going to install your application on and a list of Capabilities for your application. The validity period of this certificate is limited – half a year at this time. This basically means that you can test your application signed with the Developer certificate on a limited number of handsets but cannot sell it to end-users. You have to go through the Symbian Signed process in order to get a real certificate.

As a prerequisite for that you’ll also need to sign your application with the ACS Publisher ID certificate which identifies your company. It can be used for all applications issued by your company and is valid for 1 year.

¹ Some sources claim that applications which require the Basic Set of capabilities can be signed with a self-signed certificate. We tried that on S60 3rd Edition devices and that didn’t work for us.

When the development and testing of your application is finished and you are about to release it as a commercial product you need to submit the application to the Symbian Signed web-site for certification.

Until recently the only way was to submit your application to a test house (there are currently 3 of them) and they would test it against certain criteria. Test houses charge in the range of 185-560 Euros for the first submission (check <https://www.symbiansigned.com/app/page/overview/testhouses> for current prices).¹ If the application passes all tests you'll get it back signed with a final certificate. If there are some failures, you have to fix them and re-submit the application again for 50-100 % of the cost of the first submission.

Eventually, you get your application signed. Congratulations! Now you can sell it and there are no restrictions anymore. But, let's say, later you find a bug in that application or you just want to make the UI a little nicer basing on users' feedback. Is there a way to easily patch or partially upgrade your application? Well, no. Even a minor upgrade will require another certification. You have to go through the same Symbian Signed process again for the whole application and pay the same money. Do you still want to fix that bug?

Recently Symbian Signed announced some new simplified and cheaper ways of getting certification. This move was inevitable. But even with cheaper certification the process remains the same: you'll need to pay for certification of every new version of your application.

We would like to help developers by showing how they can develop more applications and, at the same time, keep their expenses on a manageable level. We hope this can result in more applications for Symbian OS.

Actually by expenses we mean not only the money. The Symbian Signed process is at the end of the development cycle and any delay in it translates into a real-time delay in shipping your product. You depend on a 3rd party test house or in the case of manufacturer-approved certificates you also depend on the handset manufacturer. Do you want to minimize the risk of having a 3rd party in the critical path of your project? We will show how to make that risk more manageable.

On the other hand, we don't encourage developers to pay less attention to the quality of code even if we show some back door for Symbian Signed. You'll still need to make sure that your application complies with all Symbian Signed requirements (you don't want your application to be crashing on the end-user handsets after all).

¹ There is a case when you don't pay anything for the signing. That's only if your application is going to be a freeware. You cannot sell that application, though.

3. Getting started with SignHelper

Please install S60 3rd Edition Platform SDK [1] and Nokia PC Suite [2] on your PC if you haven't done yet. Unpack our example source code into any directory. Run the Windows command line tool.

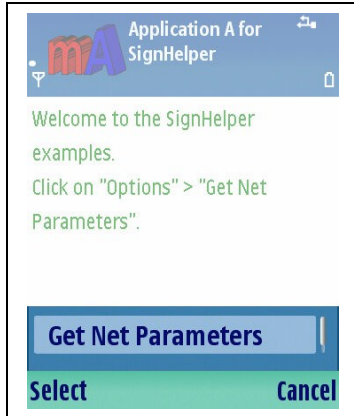
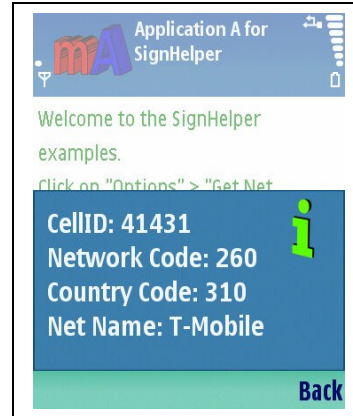
This is the preview version.
You can purchase the full version of this document at
<http://www.maccent.com/solutions.shtml>

What you'll learn from this document and code examples is how to create applications that make use of protected APIs but don't require Capabilities and, as a result, don't need to be certified by Symbian Signed and can be self-signed with your own certificate. For example, our Application A has no announced Capabilities in its project file (App_A\group\App_A.mmp),

```
TARGET          App_A.exe
UID              0x100039CE 0xA1000000
VENDORID        0
TARGETTYPE exe
EPOCSTACKSIZE   0x5000

CAPABILITY      NONE
...
```

but is able to display information about a network the handset is currently registered in (see Fig.1, Fig.2).

**Fig.1****Fig.2**

It will be no secret to say that we used `GetCurrentNetworkInfo()` from `CTelephony` class for this particular application. But if you'll look into the S60 3rd Edition SDK you'll find that using that method requires the `ReadDeviceData` Capability. And, as we said, there are no Capabilities required by the Application A. How could we achieve that? It's just a special way of developing applications.

Our example applications show how to make it working for the `ReadDeviceData` Capability but the same methodology can be used for any API that requires Capabilities.

4. How the SignHelper actually helps

**This is the preview version.
You can purchase the full version of this document at
<http://www.maccent.com/solutions.shtml>**