
Testing and Signing with Symbian Platform Security

Version 1.5
January 12, 2007

S
Y
M
B
I
A
N
O
S

Legal notice

Copyright © 2005–2007 Nokia Corporation. All rights reserved.

Nokia and Forum Nokia are registered trademarks of Nokia Corporation. Other product and company names mentioned herein may be trademarks or trade names of their respective owners.

Disclaimer

The information in this document is provided “as is,” with no warranties whatsoever, including any warranty of merchantability, fitness for any particular purpose, or any warranty otherwise arising out of any proposal, specification, or sample. Furthermore, information provided in this document is preliminary, and may be changed substantially prior to final release. This document is provided for informational purposes only.

Nokia Corporation disclaims all liability, including liability for infringement of any proprietary rights, relating to implementation of information presented in this document. Nokia Corporation does not warrant or represent that such use will not infringe such rights.

Nokia Corporation retains the right to make changes to this specification at any time, without notice.

License

A license is hereby granted to download and print a copy of this specification for personal use only. No other license to any other intellectual property rights is granted herein.

Contents

1.	Introduction	5
2.	Symbian Security Model.....	6
2.1	Reasoning behind Symbian Platform Security Model.....	6
2.2	Symbian Platform Security Model in practice	6
2.2.1	Trusted computing base	6
2.2.2	Data caging	6
2.2.3	Capabilities	7
2.3	Application signing	8
2.4	Assigning capabilities.....	8
2.5	The need for platform or manufacturer capabilities	10
3.	IDs.....	12
3.1	UID	12
3.2	Product ID	12
3.3	Manufacturer ID	12
3.4	Example of manufacturer ID and product ID usage.....	12
4.	Embedding SIS files.....	13
5.	Symbian Security Model in application development	14
5.1	Defining the application.....	14
5.2	Testing the application	14
5.2.1	Symbian Developer Certificate.....	14
5.3	Symbian Signed.....	15
5.4	Granting TCB/DRM.....	16
5.5	Symbian Self Certifiers.....	16
6.	Terms and abbreviations.....	18
7.	Evaluate this resource	19

Change history

September 5, 2005	Version 1.0	Initial document release
October 17, 2005	Version 1.1	Information on the Developer Certificate pricing updated
March 1, 2006	Version 1.2	Information on capabilities updated. Chapter about IDs added.
March 20, 2006	Version 1.3	Chapter about embedding SIS files added. Minor update to the ID chapter.
September 15, 2006	Version 1.4	Some clarifications to the capability granting section added. Section 2.5 added describing cases where certain capabilities are needed. Information about Symbian Self Certifiers added.
January 12, 2007	Version 1.5	Section 5.2.1, "Symbian Developer Certificate," updated to reflect the current changes. Clarifications made to Section 2.5, "The need for platform or manufacturer capabilities."

1. Introduction

This document describes first the basics of Symbian Platform Security, and then widens the scope to how the developer can obtain capabilities.

2. Symbian Security Model

2.1 Reasoning behind Symbian Platform Security Model

Mobile devices are gaining more and more features and thus they are becoming more valuable to their users with the amount of important information they possess. However, as mobile devices they are not the same as computers, and the idea is to make sure users still find their phones easy to use, reliable, secure, and trustworthy. For this purpose, the Symbian Platform Security Model is introduced. The purpose of the model is not to close the phones but to enable users to continue using their mobile phones as before in an easy and trusted manner. This way it is possible to provide more intelligent devices to a larger user base and this will offer a larger audience for applications.

2.2 Symbian Platform Security Model in practice

There are three main modules in the Symbian Security Model:

1. Trusted computing base.
2. Data caging.
3. Capabilities.

2.2.1 Trusted computing base

The trusted computing base is a collection of software that enforces capabilities and data caging. It contains the kernel, the file system, and the software installer. This is the controlling part of the operating system for the platform security model.

2.2.2 Data caging

Data caging means that the applications and the users have access only to certain areas of the file system. In practice the applications can access their own private directories and directories that are marked as open. It means, for example, that one application cannot access another application's private directory and data.

The access is restricted as follows:

`\resource`

Location for application icons, bitmaps, etc. Writing allowed only at application installation. Everyone can read the folder.

`\sys`

Location for binaries, including application installation registry and root certificates. Writing allowed only at application installation. Reading allowed to backup the application.

`\private`

This is a private playground for each application. Reading and writing is only allowed to the application's own directory. Backup software has read and write access to this directory.

all the rest

Access to all the other folders is free for all, for example, user's own photos, music, and documents.

2.2.3 Capabilities

A capability grants access to a set of APIs and can be obtained through certification, for example Symbian Signed. The capabilities can be divided into four:

1. Open to all
 - APIs in this category enable development of all the basic applications, for example, most of the single-player games.
 - Generally speaking, about 60 percent of the APIs are freely available without any capability requirement.
2. Granted by the user at installation time
 - Some capabilities can be granted by the user at the application installation phase.
 - The application will have the capability until the application is removed from the device.
 - This option may not be active in some devices by default. Thus the user has to activate the installation time capability granting separately.
3. Granted through Symbian Signed
 - Some capabilities are available after passing Symbian Signed testing.
 - More sensitive capabilities require declarative justification why the application needs access to such a capability. Passing the testing is required as well.
 - The most sensitive capabilities require the developer to fill in the Capability Request Form and acceptance from the platform manufacturer. Passing testing including an additional platform manufacturer defined criteria is required as well. The S60 criteria can be found in the document [Nokia Test Criteria for Symbian C++ Applications](#) available at www.forum.nokia.com/testing.
4. Granted by the manufacturer
 - The Capability Request Form also includes TCB and DRM capabilities. These are granted by the device manufacturer and are subject to a legal agreement.
 - Strong business reasoning is usually required to access these capabilities.

2.3 Application signing

S60 3rd Edition introduces mandatory signing of applications. This means that the application will not install if it has not been signed. Generally speaking there are two types of signing:

1. Signing with any private key to achieve uniqueness and to confirm the integrity of the SIS file. The makekeys application can be used to create the needed key and signsis to sign the application. Both applications are delivered with the SDK.
2. Signing with a specific private key to achieve certification — signing the application so that a trusted root certificate on a device trusts the application.

The first case has to be done by the developer during the development process in order for the application to install to the device. The latter can be done through Symbian Signed and the capabilities can be granted through it.

2.4 Assigning capabilities

Capabilities required by the application are defined at the design phase of the application. The application definition files include an MMP file, which has the information of the capabilities that the application uses. However, Carbide users can place the capability information to a specific field in Carbide UI.

At the installation phase the installer application in the device checks whether the application has been certified or signed. Then it checks the capabilities required by the application. If the application has been certified, it is checked that the root certificate is allowed to grant the required capabilities. The installation can continue if no problems are encountered.

Table 1 shows how the capabilities are divided.

Unrestricted	User-grantable (at installation time)	Symbian Signed	Manufacturer approval
60% of APIs	ReadUserData WriteUserData NetworkServices LocalServices	The user-grantable capabilities + • Location	Symbian Signed capabilities +
	UserEnvironment Note: Implementation may vary between devices.	<u>Declarative:</u> • ReadDeviceData • WriteDeviceData • PowerMgmt • SurroundingsDD • ProtServ • TrustedUI • SwEvent	<u>Capability Request Form & Manufacturer approval:</u> • DRM • TCB

Unrestricted	User-grantable (at installation time)	Symbian Signed	Manufacturer approval
		<u>Capability Request Form & Platform approval:</u> <ul style="list-style-type: none"> • DiskAdmin • AllFiles • CommDD • MultiMediaDD • NetworkControl 	

Table 1: Comparison of capabilities

Table 2 provides more information on what the separate capabilities mean in practice.

	Capability	Description
1	NetworkServices	This capability is for, e.g., dialing a number or sending a text message.
2	LocalServices	This capability is for sending or receiving information through USB, IR, and point-to-point Bluetooth profiles.
3	ReadUserData	Grants read access to user data. System servers and application engines are free to grant this restriction level to their data.
4	WriteUserData	Grants write access to user data. Again, system servers and application engines are free to grant this restriction level to their data.
5	Location	Grants access to the location of the phone.
6	UserEnvironment	Grants access to live confidential information about the user and his/her immediate environment.
7	PowerMgmt	Grants the right to kill any process in the system or to switch machine state (turn the phone off).
8	MultimediaDD	Controls access to all multimedia device drivers (sound, camera, etc.).
9	ReadDeviceData	Grants read access to sensitive system data.
10	WriteDeviceData	Grants write access to sensitive system data.
11	DRM	Grants access to protected content.
12	TrustedUI	This capability differentiates "normal" applications from "trusted" applications. If a trusted application is displaying something on the screen, a normal application cannot fake it.

	Capability	Description
13	ProtServ	Grants the right to a server to register with a protected name. Protected names start with an "!" (exclamation point). The kernel will prevent servers without ProtServ capability from using such a name, and therefore will prevent protected servers from being impersonated.
14	NetworkControl	Grants the right to modify or access network protocol controls.
15	SwEvent	Grants the right to generate and capture software key and pen events.
16	SurroundingsDD	Grants access to logical device drivers that provide input information about the surroundings of the phone.
17	TCB	Grants access to /sys and /resource directories in the phone.
18	CommDD	Grants access to communication device drivers.
19	DiskAdmin	Grants the right to disk administration functions, such as formatting a drive.
20	AllFiles	Grants visibility to all files in the system and extra write access to files under /private.

Table 2: Description of capabilities

2.5 The need for platform or manufacturer capabilities

In some cases it is evident that a capability is needed, whereas in other cases the need is not so straightforward. For example:

1. All – TCB (All capabilities except TCB) is needed from the following applications in S60 3rd Edition:
 - Client-side Message Type Module
 - Front End Process
 - Browser Plug-ins

The need for these capabilities arises from the DLL loading rules of Symbian Platform Security.

2. Device encryption applications using File Server process require the following capabilities: TCB, ProtServ, DiskAdmin, AllFiles, PowerMgmt, and CommDD.
3. Antivirus applications need at least TCB.
 - Antivirus applications also need to have access to the Symbian file server hooks to do a proper implementation of the application. This requires that the developer becomes a Symbian Platinum Partner.

There are also certain cases where the available documentation states that certain capability is needed, but closer examination can reveal that it is not the case. Forum

Nokia will place these cases in the platform security pages FAQ section at www.forum.nokia.com/platformsecurity.

3. IDs

Several different types of IDs are used in Symbian OS. It is important to know some of them in more detail.

3.1 UID

A Unique Identifier (UID) is used to uniquely identify the application. UIDs can be obtained from Symbian at www.symbiansigned.com. When applying for a UID, it is recommended to use exactly the same company name as what is stated in the company's VeriSign ACS Publisher ID.

The UIDs are divided into two ranges:

- Protected UID range: 0x00000000 ... 0x7FFFFFFF
- Unprotected UID range: 0x80000000 ... 0xFFFFFFFF

The protected range is intended for applications that will be certified — the unprotected for those that are signed. If a signed application is to be certified, the UID needs to be changed.

3.2 Product ID

A Product ID is used to identify which product the application is intended to run on. If you use a device-specific product ID, the application will install in that specific device. If you use a product ID of a specific platform version, the application will install to all of the products in that specific platform version. If the product ID is incorrect, the user will receive a warning message but is able to continue the installation.

3.3 Manufacturer ID

If you use a manufacturer ID in an IF ELSE statement with, for example, a platform ID, the application will only install to the specific manufacturer's devices in that platform.

3.4 Example of manufacturer ID and product ID usage

The following is an example of a part of a PKG file:

```
;Supports S60 3rd Edition
[0x101F7961], 0, 0, 0, {"Series603rdEditionProductID"}

IF manufacturer = 2 ; (2 is Nokia)
;This part will then contain the installation information about the
;files of the application

ELSE
"badmanufacturer.txt"-", FILETEXT, TEXTEXIT
ENDIF
```

In the example, the application can be installed to all S60 3rd Edition devices from Nokia.

4. Embedding SIS files

When granting sensitive capabilities it is important to limit the sensitive capability exposure to the minimum. This can be done by embedding a SIS file into the main application distribution SIS file. This way the embedded SIS file will include only the part of the application which will require the more sensitive capabilities. CommDD, MultimediaDD, NetworkControl, DiskAdmin, AllFiles, DRM, and TCB are regarded as sensitive capabilities.

An SA type of a SIS file can be easily embedded inside an SA-type SIS file. This can be done by adding the following line to the PKG file of the main SIS file:

```
@ " The_Embedded_SIS_name.sis", (The_Embedded_SIS_UID)
```

The benefit of using SA-type SIS files is that the implementation is relatively easy. The downside is that the embedded SIS file is visible in the Application Manager. Thus, the user may accidentally remove the file.

5. Symbian Security Model in application development

As can be understood from the security model description, it is important to know which capabilities are needed by the application and how to obtain the needed capabilities.

5.1 Defining the application

At the phase where the application is on the drawing board, that is, being planned and defined, there are two important issues that need to be taken into consideration:

1. What do the Symbian Signed criteria require from the application?
2. Which capabilities, if any, does the application require?

5.2 Testing the application

The application can be tested with the emulator of an SDK. It is advisable to also test the application on a real device in a live network — the emulators seldom get any incoming calls. If the application requires capabilities that require digital signature, testing must be done first with the emulator and secondly a Developer Certificate can be acquired through Symbian Signed for the application to obtain the desired capabilities for testing the application on a device.

5.2.1 Symbian Developer Certificate

The Symbian Developer Certificate can be used by the developer to sign their applications in order to obtain the restricted capabilities for device testing. The certificate is restricted to a certain set of IMEIs and the set cannot be changed. There are a few requirements to get a Symbian Developer Certificate. These are shown in Table 3.

Number of IMEIs	Authentication	Capabilities
1	Symbian Signed account	LocalServices, UserEnvironment, NetworkServices, Location, ReadUserData, WriteUserData, SWEvent, SurroundingsDD, ProtSrv, PowerMgmt, TrustedUI, ReadDeviceData, WriteDeviceData,
Up to 100	VeriSign ACS Publisher ID and Symbian Signed account	As above
Custom	VeriSign ACS Publisher ID, Symbian Signed account and manufacturer support	As above + DRM, NetworkControl, MultimediaDD, TCB, AllFiles, CommDD, DiskAdmin

Table 3: Requirements for the Symbian Developer Certificate

In general, the process goes as follows:

1. The developer enters the Symbian Signed portal and registers.
2. The developer uses a request tool to send a request for the developer certificate.
 - The tool can be downloaded from the Symbian Signed Web site.
 - The VeriSign ACS Publisher ID may be needed at this phase. Note that the VeriSign ACS Publisher ID incorporates a yearly fee.
3. The certificate is created and sent back to the developer.
 - The certificate is valid for 6 months from the date the Developer Certificate was received.
 - The Symbian Developer Certificate is free of charge; however, the developer may need the VeriSign ACS Publisher ID, which is subject to a fee.

The process of obtaining the custom developer certificate is done through the same portal, but using a different link. However, there the manufacturer will need to approve the request before the developer can get a hold of the developer certificate. The approval is done with the basic assumption that if the application can get a developer certificate it can also receive the final certification.

5.3 Symbian Signed

To receive the final certification, the application must go through Symbian Signed. This section lists some points in the process that are important to understand.

The VeriSign ACS Publisher ID is required for any application submitted to Symbian Signed. It is a prerequisite for entering Symbian Signed. The same VeriSign ACS Publisher ID can be used as the one that was used when applying for the Developer Certificate.

As already mentioned, certain capabilities can be obtained through standard Symbian Signed by passing the testing and declaring why certain capability is used. In case the application requires any of the following capabilities:

- CommDD
- MultimediaDD
- NetworkControl
- DiskAdmin
- AllFiles

the developer must fill in a Capability Request Form while submitting the application. The Capability Request Form will be sent to the platform manufacturer for approval.

The following requirements apply to the above-mentioned five capabilities:

- The application must pass the Symbian Signed and additional test criteria. The Nokia test criteria for S60 applications can be found in the document [Nokia Test Criteria for Symbian C++ Applications](http://www.forum.nokia.com/testing) available at www.forum.nokia.com/testing.

- If feasible to the application architecture, the spread of the sensitive capabilities should be limited by using the embedded SIS method described in Chapter 4, "Embedding SIS files."
- The part of the application which requires the above-mentioned capabilities needs to be packaged into a separate SIS file.
 - This SIS file will have the required capabilities.
 - The developer can use this SIS file as embedded in the main application SIS file in application distribution.

5.4 Granting TCB/DRM

With TCB and DRM capabilities the process is a bit different. For obtaining the capabilities the developer must fill in the Capability Request Form. The case is evaluated and the developer is contacted for more details.

The requirements for acceptance are as follows:

- A legal agreement with Nokia about the possible liabilities and the use of the capability.
- The application must pass the Symbian Signed tests and potentially the additional defined criteria. The Nokia test criteria for S60 applications can be found in the document [Nokia Test Criteria for Symbian C++ Applications](#) available at www.forum.nokia.com/testing.
- The application is defined so that it can be installed in all the devices of the same platform version from Nokia. See Section 3.4, "Example of manufacturer ID and product ID usage."
- The part of the application which requires the above-mentioned capabilities needs to be packaged into a separate SIS file.
 - This SIS file will have the required capabilities.
 - The developer can use this SIS file as embedded in the main application SIS file in application distribution.

5.5 Symbian Self Certifiers

A developer who creates vast amounts of applications annually can become a Symbian Self Certifier. Such developers must have a solid quality assurance process and be of good repute. They should have had at least one application Symbian Signed prior to application. They must also reach a legal agreement with Symbian. The scheme will require the developer to pay Symbian an annual fee.

Such developers can certify their applications with the capabilities any developer can obtain through Symbian Signed without platform or device manufacturer approval. If the developer creates applications that require platform or device manufacturer approval, the Symbian Self Certifier has two options:

1. Reach an agreement with the platform or device manufacturer that the developer can use the needed capabilities. Such arrangement requires solid business reasoning for why the developer has to have access to the platform or device manufacturer capabilities in Self Certification.
2. Package the part of the application requiring the platform or device manufacturer capabilities into a separate SIS file which is certified

separately. The developer embeds the certified SIS into the main application delivery and uses the Self Certification scheme to certify applications.

6. Terms and abbreviations

Term or abbreviation	Meaning
API	Application programming interface
CA	Certificate Authority
PKI	Public Key Infrastructure
SIS	Installer file format used in Symbian OS
UI	User interface
PU type SIS file	Partial Upgrade type of SIS file.
VeriSign ACS Publisher ID	VeriSign's product, an identity certificate. "Authenticated Content Signing Publisher ID"

7. Evaluate this resource

Please spare a moment to help us improve documentation quality and recognize the resources you find most valuable, by [rating this resource](#).