
S60 3rd Edition: Tool Chain, IDEs, And Development Process

Version 1.2
November 21, 2005

S60 p l a t f o r m

Legal Notice

Copyright © 2005 Nokia Corporation. All rights reserved.

S60, Nokia and Nokia Connecting People are registered trademarks of Nokia Corporation. Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. ActiveState, ActivePerl, and PerlScript are trademarks of ActiveState Tool Corp. Metrowerks and CodeWarrior are registered trademarks of Metrowerks Inc. 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.	Changes in the Development Process	6
2.1	How Does Platform Security Affect the Development Process?	6
2.1.1	Update the .mmp file	6
2.1.2	Build for the emulator and test	6
2.1.3	Request ACS Publisher ID	6
2.1.4	Acquire Symbian Developer Certificate.....	7
2.1.5	Build for the device and test:	7
2.1.6	Acquire Symbian Signed certification for releasing the application.....	7
2.2	Contents of an .mmp File	8
2.3	Mandatory Signing of Applications.....	9
2.4	An Example of the Renewed Development Process	10
2.4.1	Building an application from the command line.....	10
2.4.2	Building an application through the CodeWarrior IDE.....	10
2.4.3	Testing the application in the emulator.....	11
2.4.4	Acquiring Symbian Developer Certificate.....	12
2.4.5	Building the application for a target device.....	14
2.4.6	Acquiring Symbian Signed for releasing the application	15
2.5	Symbian Signed Process	15
3.	Changes in Tool Chain	16
3.1	New Compilers.....	16
3.1.1	GNU C Compiler EABI	16
3.1.2	Nokia CodeWarrior EABI.....	16
3.1.3	ARM Real View Compilation Tools	16
4.	Supported C++ IDEs.....	17
5.	Terms and Abbreviations	18
6.	References	19
7.	Evaluate This Resource.....	20

Change History

June 3, 2005	V1.0	Initial document release
September 1, 2005	V1.1	Updated for S60 3 rd Edition SDK Help file Beta release
November 21, 2005	V1.2	Updated for S60 3 rd Edition SDK Final release

1. Introduction

S60 3rd Edition is based on Symbian OS v9.1. This entails the following new features, which have an impact on the development process:

- Platform security

Platform security protects the integrity of phones, provides confidentiality for sensitive data, and controls access to sensitive operations. In application development, platform security entails acquiring certificates and determining capabilities for applications. See Section 2.1, “How Does Platform Security Affect the Development Process?,” for more information on this topic.

For more information about platform security, please refer to the *Platform Security – a Technical Overview* document (at http://www.symbian.com/developer/techlib/papers/cpp_sysarch.asp).

- Total binary break

The new ARM compiler standard Application Binary Interface (ABI) causes a total binary break, which means that S60 2nd and 3rd Edition will not be binary compatible. The new standard enables more code in ROM and improved application performance. In application development, the total binary break entails the need for new compilers.

- Real-time kernel

A real-time Symbian kernel EKA2 suitable for telephony stacks, multimedia, Games, VoIP.

This document aims to outline the effect these features have on the development process. The document also provides information on the new compilers needed for development as well as IDEs supported in S60 3rd Edition SDKs.

2. Changes in the Development Process

This chapter outlines how the new features introduced in S60 3rd Edition affect the development process.

2.1 How Does Platform Security Affect the Development Process?

The targets of platform security are:

- To protect the integrity of the phone.
- To provide confidentiality for sensitive data.
- To control access to sensitive operations.



Note: Platform security only affects native Symbian C++ applications. Thus the development process for Java/MIDP applications is not affected by platform security.

To migrate an application to S60 3rd Edition and platform security, you can follow various useful migration guides found in the Symbian Developer Library:

- **Symbian OS SDK v9.1 > Symbian OS Guide > Platform security**

The new development process consists of the following steps:

2.1.1 Update the .mmp file

- With the capabilities that the application requires.
- Add secure ID and vendor ID (see **Symbian OS SDK v9.1 > Symbian OS Guide > Platform security**).

See also Section 2.2, “Contents of an .mmp File.”

2.1.2 Build for the emulator and test

- Build the application for WINSW (command line or IDE).
- Test the application in the emulator using different settings to control platform security behavior.

2.1.3 Request ACS Publisher ID



Note: This step is optional and needed only if:

- You intend to request a larger set of capabilities for your Developer Certificate, or
- You need a Developer Certificate for up to 20 devices, or
- You plan to submit your application for the Symbian signed program.

See also Section 2.4.3, “Testing the application in the emulator.”


2.1.4 Acquire Symbian Developer Certificate

The Symbian Developer Certificate can be used by developers 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 1.

Number of IMEIs	Authentication	Capabilities
1	Symbian Signed account	Local Services, User Environment, Network Services, Location, Read User Data, Write User Data, SWEvent, SurroundingsDD, ProtSrv, Power Mgmt
Up to 20	VeriSign ACS Publisher ID and Symbian Signed account	As above + ReadDeviceData & Write Device Data, Trusted UI
Custom	VeriSign ACS Publisher ID, Symbian Signed account and manufacturer support	As above + DRM, Network Control, MultimediaDD, TCB, All Files, CommDD, Disk Admin

Table 1: Requirements for the Symbian Developer Certificate

See also Section 2.4.4, “Acquiring Symbian Developer Certificate.”

	Note: It may take some time to get the VeriSign ACS Publisher ID. Thus it is good to apply for one in time.
-------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------

2.1.5 Build for the device and test:

- Build the application for the target device.
- Sign the `.sis` file with the developer access certificate.
- Test the application in the target device.

See also Section 2.4.5, “Building the application for a target device.”

2.1.6 Acquire Symbian Signed certification for releasing the application

- Once certified, the application can be shipped.

See also Section 2.5, “Symbian Signed Process.”

2.2 Contents of an .mmp File

The contents of an .mmp file in S60 3rd Edition are the following:

```

/* Copyright (c) 2005, Nokia. All rights reserved */

TARGET                HelloWorldBasic.exe
TARGETTYPE            exe
UID                   0x0 0xA000017F
SECUREID              0x00000000
EPOCSTACKSIZE        0x5000
SOURCEPATH            ..\src
SOURCE                HelloWorldBasic.cpp
SOURCE                HelloWorldBasicApplication.cpp
SOURCE                HelloWorldBasicAppView.cpp
SOURCE                HelloWorldBasicAppUi.cpp
SOURCE                HelloWorldBasicDocument.cpp
SOURCEPATH            ..\data
START_RESOURCE        HelloWorldBasic.rss
HEADER
TARGETPATH APP_RESOURCE_DIR
END //RESOURCE
START_RESOURCE        HelloWorldBasic_reg.rss
#ifdef WINS32
TARGETPATH            \private\10003a3f\apps
#elseif
TARGETPATH            \private\10003a3f\import\apps
#endif
END //RESOURCE
USERINCLUDE           ..\inc
SYSTEMINCLUDE         \epoc32\include
LIBRARY               euser.lib
LIBRARY               apparc.lib
LIBRARY               cone.lib
LIBRARY               eikcore.lib
LIBRARY               avkon.lib
LIBRARY               commonengine.lib
LANG SC
CAPABILITY            NONE

// End of File

```

Notice that in S60 3rd Edition the `TARGET` and `TARGETTYPE` values in the above are `.exe`, and not `.app`, as in S60 2nd Edition. See Symbian Developer Library platform security migration guide for more information (**Symbian OS SDK v9.1 > Symbian OS Guide > Platform security**).

New values in S60 3rd Edition .mmp files are:

`SECUREID` – This keyword is used to define the Secure Identifier (SID), which is a locally unique identifier (that is, in the phone where it is installed). Each executable contains a secure identifier. The SID is used to

- determine which private directory a process can access
- identify the caller applications.

The SID can be specified by a `SECUREID` statement in the project's .mmp file. If a `SECUREID` statement is not given, the `UID3` specified in the .mmp file is used. If this is not defined, `KNu11UID` is used.

`CAPABILITY` – This keyword is used to define the capabilities for the application. See *S60 3rd Edition: Capabilities* in the S60 3rd Edition SDK for Symbian OS Help for more details about possible capabilities.

If the `CAPABILITY` keyword is not present in the `.mmp` file, the default of `CAPABILITY NONE` is used.



Note: Capabilities are defined only for signed applications; unsigned applications have no capability defined. The signing procedure depends on the application's needed capabilities. For details, please, refer to Section 2.3, "Mandatory Signing of Applications."

VENDORID – This keyword is used to define a Vendor Identifier (VID). The VID uniquely identifies the source of the application.

- Signed applications from the same vendor will have the same VID.
- A VID cannot be modified after building the application.

2.3 Mandatory Signing of Applications

S60 includes mandatory `.sis` file signing, which means that all `.sis` files must be signed before installing them to the device. The table below summarizes the different signing options and when to use them.

Signing With	Used When
Developer Certificate	<p>When testing the application in developer's own device(s). This certificate is received from Symbian.</p> <p>The Symbian Developer Certificate can be used by developers 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 1, "Requirements for the Symbian Developer Certificate."</p> <p>See Sections 2.1.4, "Acquire Symbian Developer Certificate," and 2.4.4, "Acquiring Symbian Developer Certificate."</p>
Self Signed Certificate	<p>When an application requires no capabilities or utilizes APIs, which are open to all (see <i>S60 3rd Edition: Capabilities</i> for more information). This can be used when testing an application in one's own device or when shipping the application.</p> <p>Note: The application is regarded as Untrusted if signed with the Self Signed Certificate.</p> <p>See the <i>How to create a private key and self signed certificate</i> section in <i>How to Sign .sis Files</i> (in the S60 3rd Edition SDK for Symbian OS Help) for more details.</p> <p>This process is recommended for freeware applications using unrestricted capabilities.</p>

Signing With	Used When
Symbian Signed	<p>Mandatory if the application is using capabilities beyond the scope of APIs open to all and the application is shipped.</p> <p>See Section 2.5, “Symbian Signed Process,” for more details.</p>

2.4 An Example of the Renewed Development Process

The following example illustrates the renewed application development process in S60 3rd Edition.

2.4.1 Building an application from the command line

To build a test application (`winscw`) from the command line, do the following:

1. Create `abld` and the project `makefiles` by entering the following command in the SDK installation directory:

```
bldmake bldfiles
```

2. Build `winscw` by entering the following command in the SDK installation directory:

```
abld build winscw
```

2.4.2 Building an application through the CodeWarrior IDE

To build a test application (`winscw`) through the CodeWarrior IDE, do the following:

1. Start the CodeWarrior IDE.
2. Select **File > Import Project From .mmp File**.
3. Select an SDK to use in this project and click **Next**.
4. Select the `.mmp` file to import (for example, `helloworldbasic.mmp`).
5. Select `WINSCW` as the platform selection.
6. Select check box **Create Files in Default Root Directory**.
7. Click **Finish**.

The `.mmp` project file has now been imported to the CodeWarrior IDE.

To compile the application in the CodeWarrior IDE, do the following:

1. Select **Project > Make** (or press **F7**) to build the project.
2. Select **Project > Remove Object Code** (or press **Ctrl + -**) to clean the application.
3. Select **Project > Debug** (or press **F5**) to debug your application.

2.4.3 Testing the application in the emulator

You can test your application in the emulator provided with the SDK. The following important settings in the `epoc.ini` file are very useful when testing an application with platform security:

- **PlatSecEnforcement**: This setting determines what action is taken when a capability or other PlatSec policy check fails. The `PlatSecEnforcement` key word is followed either by an `On` or `Off` token in `epoc.ini`. For example, `PlatSecEnforcement On` enforces platform security in the emulator.
- **PlatSecDiagnostics**: If set, platform security diagnostics messages are sent to the trace/debug output file and/or an IDE's system log view. The `PlatSecDiagnostics` key is followed either by an `On` or `Off` token.
- **LogToDebugger**: Enables logging of trace messages to the debugger's **System Log** window. To enable logging, enter `LogToDebugger = 1` in `epoc.ini`. To disable debugging, enter `LogToDebugger = 0`.
- **LogToFile**: Logs are sent to the trace/debug output file, `epocwind.out` in the Windows temporary directory. This setting can be on at the same time as the `LogToDebugger` setting. To enable log file writing, enter `LogToFile = 1` in `epoc.ini`. To disable log file writing, enter `LogToFile = 0` in `epoc.ini`.

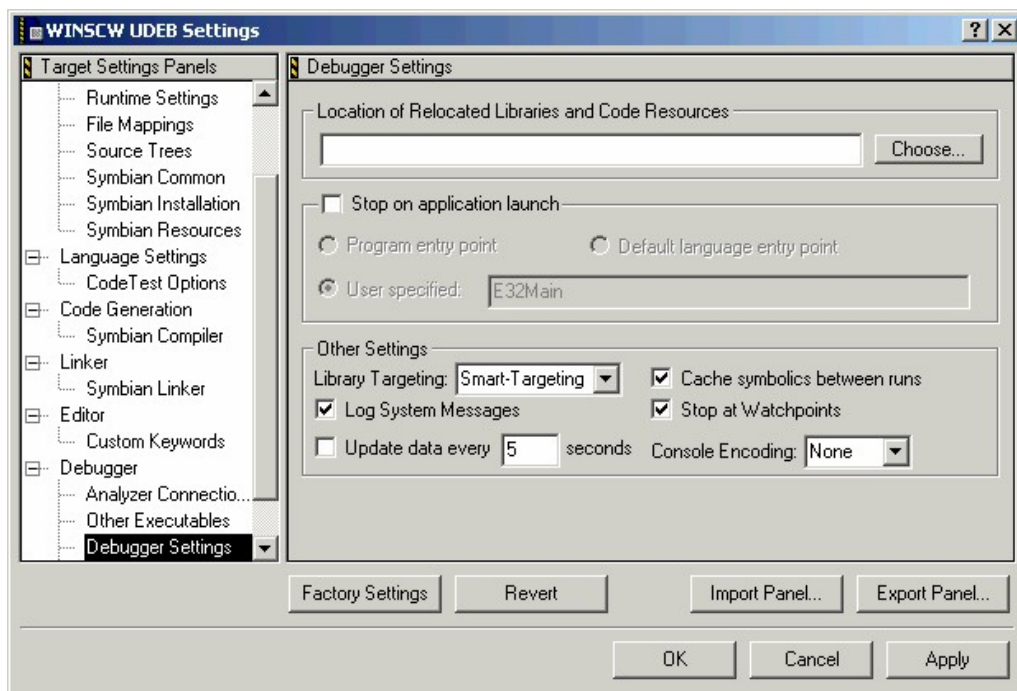


Note: The S60 3rd Edition C++ SDK has a graphical user interface, which can be used to configure the above settings. The GUI is available from the emulator menu bar: Select **Tools > Preferences** and then select the **CPP Debug** tab.

The normal development cycle in the emulator would be to first disable platform security enforcement, but in this case activate platform security diagnostics and one or more of the desired log outputs (debugger or file). Once all platform security diagnostics messages have been resolved (by adding the needed capabilities to the `.mmp` file, etc.), the next step would be to turn enforcement on and test the application again. To run the application in the emulator, select **Project > Run** from the Main menu of the CodeWarrior IDE. The application opens in the emulator.

To enable the **System Log** window in the CodeWarrior IDE, proceed as follows:

- Select **Edit > WINSCW UDEB Settings** from the Main menu
- In the **Target Settings Panel** select **Debugger > Debugger Settings** and check the **Log System Messages** check box. See the picture below how to activate the **System Log** window in CodeWarrior.



2.4.4 Acquiring Symbian Developer Certificate

Before installing the application's `.sis` file to the target device and testing it there, you need to acquire a Developer Certificate for the device/devices in question. The Developer Certificate mechanism provides a certificate, which enables the requested capabilities for the application.



Note: If your application does not require capabilities beyond the group of APIs open to all (see *S60 3rd Edition: Capabilities* for more information), you don't necessarily need a Developer Certificate.

The Symbian Developer Certificate can be used by developers 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 2, "Requirements for the Symbian Developer Certificate."

Number of IMEIs	Authentication	Capabilities
1	Symbian Signed account	Local Services, User Environment, Network Services, Location, Read User Data, Write User Data, SWEvent, SurroundingsDD, ProtSrv, Power Mgmt
Up to 20	VeriSign ACS Publisher ID and Symbian Signed account	As above + ReadDeviceData & Write Device Data, Trusted UI

Number of IMEIs	Authentication	Capabilities
Custom	VeriSign ACS Publisher ID, Symbian Signed account and manufacturer support	As above + DRM, Network Control, MultimediaDD, TCB, All Files, CommDD, Disk Admin

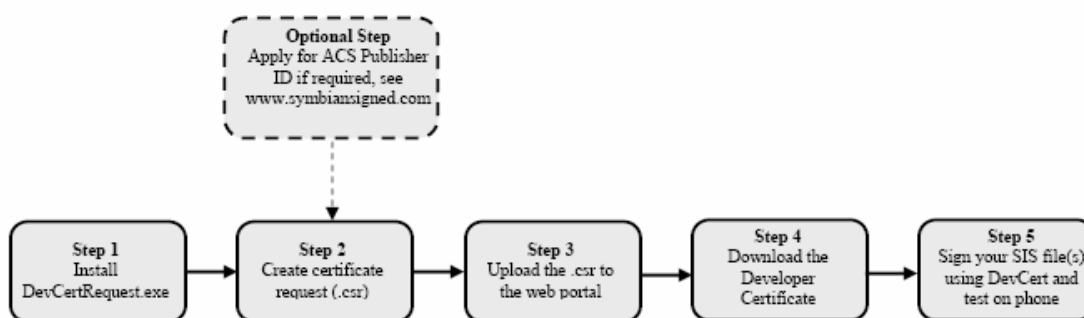
Table 2: Requirements for the Symbian Developer Certificate

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 VeriSign ACS Publisher ID may be needed at this phase. Note that the VeriSign ACS Publisher ID incorporates a yearly fee. See Section 2.4.4.1, “Apply for VeriSign ACS Publisher ID” for details.
3. The certificate is created and sent back to the developer.
 - The certificate is valid for 6 months from the date the Developer Access 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. However, there the manufacturer will need to approve the request before the developer can get a hold of the developer certificate.

See below an overview of Symbian Developer Certificate Request process:




See <https://www.symbiansigned.com/app/page/devcertgeneral> for more details about the Developer Certificate and more detailed description of the process.

2.4.4.1 Apply for VeriSign ACS Publisher ID


You may need a VeriSign ACS Publisher ID before applying for the Developer Certificate if you want a certificate, which:

- Enables you to sign the application for 1-20 devices, or
- Enables a larger set of capabilities.

See <https://www.symbiansigned.com/app/page/devcertgeneral> for more exact details when the ACS Publisher ID is needed.

	<p>Note: The VeriSign ACS Publisher ID is optional and needed only if:</p> <ul style="list-style-type: none"> • You intend to request a larger set of capabilities for your Developer Certificate, or • You need a Developer Certificate for up to 20 devices, or • You plan to submit your application for the Symbian signed program.
-----------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Next, you need to apply for and purchase an ACS Publisher ID. To do this, please go to Verisign's Web site at <http://www.verisign.com/products-services/security-services/code-signing/symbian-content-signing/index.html>.

	<p>Note: Getting the VeriSign ACS Publisher ID may take some time. Thus it is recommended that the process is started early in application development.</p>
-------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------

Security issues of the procedure are managed by ACS Publisher ID and PKI certification.


2.4.5 Building the application for a target device

After installing the Developer Certificate in your development environment, you can build the needed `.pkg` file by entering the following command in the `\sis\` directory (see *How to Sign .sis File*):

```
abld build armv5
```

The `.pkg` file created looks like this:

```
;Header
#{ "HelloWorldBasic" }, (0x11111111), 1, 2, 3, TYPE=SA
%{ "HelloWorldBasic" }
: "Unique Vendor Name"
"\epoc32\release\armv5\urel\helloworldbasic.exe"
- "!:\sys\bin\helloworldbasic.exe";
```

	<p>Note: All executable binaries are located in the <code>\sys\bin</code> directory.</p>
-------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------

To build the needed `.sis` file, enter the following command in the `\sis\` directory:

```
makesis helloworldbasic.pkg
```

The `helloworldbasic.sis` file is created in the `\sis\` directory.

To sign the `.sis` package, run the `signsis` command and enter certificate information as a parameter (see *How to Sign .sis Files* for more information). Use either:

- Developer Certificate
- Self Signed certificate

2.4.6 Acquiring Symbian Signed for releasing the application

In addition to the Developer Certificate, Symbian Signed is also needed before applications can be published. This is an industry-wide application testing and signing program for native Symbian OS applications. For more detailed information about Symbian Signed, please refer to Section 2.5, “Symbian Signed Process.”

2.5 Symbian Signed Process

Symbian Signed is an industry-wide application signing and verification program for native Symbian OS applications.

Symbian Signed includes:

- A means of authenticating the developer (the same VeriSign ACS Publisher ID which is used for the Developer Access Certificate is used here, too).
- A process for testing applications against a series of industry-agreed criteria.
- A means of signing applications that have passed the tests.

More information on Symbian Signed is available from the Symbian Signed Web site (<http://www.symbiansigned.com/>). The Symbian Signed Web site provides useful information about the verification process, test criteria, testing tools, qualified Test Houses, and the usage of the Symbian OS logo.

3. Changes in Tool Chain

The ARM compiler standard Embedded Application Binary Interface (EABI) causes a total binary break between S60 2nd Edition and 3rd Edition.



Note: ARM EABI is specific to C++ development; it will not have an effect on Java/MIDP development.

Benefits of the ARM EABI are:

- Decreased memory usage
- More code in ROM
- Improved application performance

From the developer's point of view, the total binary break means that applications need to be compiled with a new compiler (see Section 3.1, "New Compilers") before installing them to a target device. If you have an application that has been created for devices compliant with S60 2nd Edition (2.x), you need to recompile it in order for it to run on devices compliant with S60 3rd Edition.



Note: Recompiled applications are not backward compatible. That is, if you recompile, for example, a S60 2nd Edition, Feature Pack 3, application with one of the compilers listed in Section 3.1, "New Compilers," it will no longer run on S60 2nd Edition, Feature Pack 3. Thus, you may want to maintain two versions of the application; one for S60 2nd Edition and one for S60 3rd Edition.

3.1 New Compilers

The following compilers can be used for compiling S60 3rd Edition applications for devices. Notice that you will need to use one of these compilers also if you want to recompile a S60 2nd Edition application for S60 3rd Edition.

3.1.1 GNU C Compiler EABI

A free GNU C Compiler (GCC) Embedded Application Binary Interface (EABI). The compiler is primarily aimed at developers and will be available in the first half of 2005. This compiler is delivered in the S60 SDK.

3.1.2 Nokia CodeWarrior EABI

The Nokia CodeWarrior EABI compiler is included with Nokia CodeWarrior for Symbian OS 3.1 Pro and OEM Editions. It is targeted for application development and will give better performance than the GCC Compiler. Nokia CW EABI has been planned to be available in the second half of 2005.

3.1.3 ARM Real View Compilation Tools

ARM Real View Compilation Tools (RVCT, version 2.2 or higher) is primarily aimed at S60 licensees. It gives the best performance and smallest code compared to other alternatives; it will be used primarily for ROM builds. ARM RVCT has been planned to be available in the second quarter of 2005.

4. Supported C++ IDEs

The following C++ IDEs are supported in S60 3rd Edition Software Development Kits:

- Metrowerks CodeWarrior Development Studio for Symbian OS 3.1 (OEM, Professional and Personal)
- Microsoft .NET 2003.



Note: The Nokia Developer's Suite (NDS) for Symbian OS package must also be installed in order to fully support the Microsoft .NET 2003 IDE. For more information on the NDS, please visit Forum Nokia at <http://www.forum.nokia.com>.

5. Terms and Abbreviations

Term or abbreviation	Meaning
ACS Publisher ID	<p>The ACS Publisher ID identifies the developer and that the company he or she works for does really exist. The ID is used to verify the source of the content.</p> <p>The ACS Publisher ID can be purchased from VeriSign at http://www.verisign.com/products-services/security-services/code-signing/symbian-content-signing/index.html</p>
Capability	<p>A capability grants access to a set of APIs, directories, and files. Most capabilities are given at installation. An application signature determines its capabilities.</p>
Data caging	<p>Applications and the user have restricted access to the file system.</p> <p>An application can access its own private directory and open directories, but it can't access private directories of other applications.</p> <p>The new file structure:</p> <p><code>/sys/bin</code> directory holds all executables (restricted access).</p> <p><code>/private/</code> contains application-specific private data (restricted access).</p> <p><code>/private/<SID></code>, where SID is an application's ID.</p> <p><code>/resource/</code> contains publicly shared read-only files.</p> <p>Other directories are open for read and write.</p>
EABI	Embedded Application Binary Interface
PKI public key infrastructure	<p>System of digital certificates, certification authorities, and registration authorities that verify and authenticate the validity of each party involved in an electronic transaction through the use of public key cryptography.</p>
Total binary break	<p>Between S60 2nd and 3rd Edition there is a total binary break; the 3rd Edition will not be binary compatible with the 2nd Edition. The total binary break is caused by the ARM compiler standard EABI.</p>
Trusted Computing Base, TCB	<p>Trusted Computing Base is the collection of software that enforces capabilities, identifiers, and data caging. TCB contains the kernel, the file system, and the software installer.</p>

6. References

Symbian Ltd., *Platform Security - a Technical Overview*
(http://www.symbian.com/developer/techlib/papers/cpp_sysarch.asp)

S60 3rd Edition: Capabilities (S60 3rd Edition SDK for Symbian OS **Help > Introduction to S60 3rd Edition**)

How to Sign .sis Files (S60 3rd Edition SDK for Symbian OS **Help > Introduction to S60 3rd Edition**)

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](#).