
S60 Platform: FAQ

Version 1.9
April 18, 2007

S60 platform

Legal notice

Copyright © 2005, 2006, 2007 Nokia Corporation. All rights reserved.

Nokia, Forum Nokia, Nokia 7650, Nokia 7710, and Nokia 9200 are trademarks or registered trademarks of Nokia Corporation. Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. Bluetooth is a registered trademark of Bluetooth SIG, 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.	Key concepts	7
1.1	What is the S60 platform?	7
1.2	What are the main features available on S60 devices?	7
1.3	What do the different “editions” of the S60 platform represent?	7
1.4	What are feature packs?	8
1.5	What does the S60 platform mean for developers?	8
1.6	How is the S60 platform deployed?	8
1.7	What are the different editions and feature packs of the S60 platform?	8
1.8	What markets are targeted by S60 devices?	8
1.9	What user interfaces are available on S60 devices?	9
2.	S60 platform architecture	10
2.1	What is the architecture of the S60 platform?	10
2.2	What does <i>Symbian OS</i> refer to?	10
2.3	What does <i>Symbian OS Extensions</i> refer to?	10
2.4	What does <i>S60 Platform Services</i> refer to?	11
2.5	What does <i>S60 Application Services</i> refer to?	11
2.6	What does <i>S60 Java™ Technology Services</i> refer to?	12
2.7	What does <i>S60 Applications</i> represent?	12
2.8	What does the S60 UI style represent?	12
3.	Features of the S60 platform	13
3.1	What is the underlying operating system for each edition and feature pack release?	13
3.2	What is included in the base software for S60 1st Edition?	13
3.3	What main features are available in S60 1st Edition, Feature Pack 1?	13
3.4	What is included in the base software for S60 2nd Edition?	13
3.5	What main features are available in each feature pack for S60 2nd Edition?	14
3.6	What is included in the base software for S60 3rd Edition?	15
3.7	What main features are available in S60 3rd Edition, Feature Pack 1?	16
3.8	What main features are available in S60 3rd Edition, Feature Pack 2?	17
3.8.1	Is Flash Lite from Adobe going to be available on all S60 devices from S60 3rd Edition, Feature Pack 1 onward?	18
3.8.2	What other improvements does S60 3rd Edition, Feature Pack 1 offer developers?	18
4.	S60 3rd Edition	19
4.1	What are the key concepts and issues for S60 3rd Edition?	19
4.2	What are the costs and benefits of these new features?	19
4.3	Does S60 3rd Edition maintain binary compatibility with earlier editions of the S60 platform?	20

4.4	What are the advantages of the Application Binary Interface (ABI) for the ARM® Architecture?	20
4.5	Does the binary-compatibility break mean higher porting costs?	20
4.6	What features are included in platform security?	21
4.7	What are the capability sets?	21
4.8	What are the certification requirements of each set of capabilities?	22
4.9	What are the different signing requirements for an S60 3rd Edition application?	22
4.10	Will all applications need to be signed?	23
4.11	Will all applications need to be Symbian Signed?	23
4.12	If an application does not require Symbian Signed to work, are there any other reasons it should be Symbian Signed?	24
4.13	Can capabilities requiring Symbian Signed certification be accessed during testing?	24
4.14	What is involved in obtaining Symbian Signed status?	24
4.15	Are there any tools to assist with the Symbian Signed process?	25
4.16	Are there any other tools to help with development in the platform security environment?	25
4.17	Obtaining Symbian Signed status has a cost, so how are freeware applications going to get to market?	25
5.	Development technologies supported by the S60 platform	26
5.1	What development technologies are available?	26
5.2	C++ application development	26
5.2.1	Is standard C or C++ used for application development?	26
5.2.2	What types of applications can I produce using the native Symbian and S60 C++ APIs?	26
5.2.3	What opportunities does Open C create for C/C++ developers?	27
5.2.4	On which editions of the S60 platform is Open C available?	27
5.2.5	Can I use Open C to create applications without having to learn Symbian C++?	27
5.2.6	What type of applications can I create with the RGA APIs?	27
5.2.7	Will my native C++ application be compatible with other S60 devices?	27
5.2.8	Is there any information on porting existing applications to the S60 platform?	28
5.2.9	How are applications downloaded and installed on an S60 device?	28
5.3	Java™ development	28
5.3.1	What version of Java™ MIDP does the S60 platform support?	28
5.3.2	What Java™ APIs does S60 1st Edition support?	28
5.3.3	What Java™ APIs does S60 2nd Edition support?	28
5.3.4	What Java™ APIs does S60 3rd Edition support?	29
5.3.5	Does Symbian OS platform security affect Java™ development?	29
5.3.6	Will my Java™ application be compatible with other S60 devices?	29
5.3.7	Will my Java™ application be compatible with devices based on other platforms?	29

5.3.8	Will my Java™ application be compatible with other MIDP devices?	29
5.3.9	How are applications downloaded and installed on an S60 device?	30
5.4	Content development	30
5.4.1	What browsing standards does the S60 platform support?	30
5.4.2	What messaging standards does the S60 platform support?	30
5.4.3	Will browser-based applications for S60 3rd Edition run on devices based on earlier editions?	30
5.4.4	Will MMS content and applications for S60 3rd Edition work with earlier editions of the platform?	30
5.4.5	What is OMA DRM?	31
5.4.6	What is OMA Client Provisioning?	31
5.4.7	What enhancements to DRM are implemented in S60 3rd Edition?	31
6.	Development tools for the S60 platform	32
6.1	What IDE options are available for C++ developers?	32
6.2	What tools are available for C++ development?	32
6.3	What Java™ development tools are available for the S60 platform?	33
6.4	What tools are available for content developers?	33
6.5	Are all of Nokia's tools available to all developers?	33
7.	Developing for the S60 platform	34
7.1	Why target the S60 platform for initial development of mobile applications?	34
7.2	What issues are associated with optimizing applications across the range of S60 devices?	34
7.3	What issues are associated with migrating applications to other Nokia platforms?	35
7.4	What issues are associated with porting applications to other devices based on Symbian OS?	35
7.5	What issues are associated with porting applications to devices based on unrelated operating systems across the mobile market?	35
8.	Known issues	36
8.1	What are the known issues that S60 application developers should be aware of?	36
9.	Business case	37
9.1	What is the business case for using the S60 platform?	37
10.	Getting applications to market.....	38
10.1	How can Nokia help get my application to market?	38
11.	Evaluate this resource	39

Change History

October 29, 2003	Version 1.0	Initial document release
June 28, 2004	Version 1.1	Document updates throughout
April 25, 2005	Version 1.2	Updated for S60 3rd Edition
October 10, 2005	Version 1.3	Updated for S60 3rd Edition, Feature Pack 1
November 9, 2005	Version 1.4	Updated Chapter 4 with additional details on signing requirements Updated Chapter 5 with information about new Carbide.c++ tool
December 16, 2005	Version 1.5	Updated DRM details in Chapter 5
December 23, 2005	Version 1.6	Updated JSR support details in Chapters 3 and 5
July 3, 2006	Version 1.7	Updated details on the platform architecture in Chapter 2 and added details about S60 3rd Edition, Feature Pack 1 to Chapter 3
February 7, 2007	Version 1.8	Added details about S60 3rd Edition, Feature Pack 2 to Chapters 2, 3, 5, and 7 Made minor updates throughout the document
April 18, 2007	Version 1.9	Added details on RGA APIs to Chapters 3 and 5

1. Key concepts

1.1 What is the S60 platform?

The S60 platform, built on Symbian OS, is the market-leading smartphone platform. It incorporates all key mobile technologies expected by today's sophisticated enterprise users and consumers, and it provides revenue opportunities for the full range of stakeholders in the mobile marketplace. As the S60 platform has developed, it has raised the bar on smartphone features by taking the lead in implementing many innovations.

1.2 What are the main features available on S60 devices?

S60 devices feature:

- Advanced telephony.
- At minimum, a 176 x 208-pixel color screen.
- Innovative form design and keypad layout.
- Personal information manager (PIM) applications (including Contacts and Calendar).
- Messaging.
- Internet browsing.

Many S60 devices also feature:

- Flash Lite from Adobe player.
- Nokia Push to Talk over Cellular (PoC).
- Digital camera.
- Music player.
- Media gallery.
- Video recorder.
- Sound recorder.
- FM radio.
- Over-the-air (OTA) synchronization.
- Viewer and editor for Microsoft Office documents.

1.3 What do the different “editions” of the S60 platform represent?

The S60 platform is continually developing and adding new features. Stakeholders must be assured that they are buying into the biggest and best platform. A new edition of the S60 platform represents a collection of significant feature updates and additions, often accompanied by a version change in the underlying Symbian OS.

1.4 What are feature packs?

Feature packs are important releases of additional features that become available between releases of new editions. For example, S60 2nd Edition had three feature packs, all of which brought enhanced capabilities to the edition. The ongoing release of feature packs allows the S60 platform to develop continually, while maintaining a common baseline across the editions.

1.5 What does the S60 platform mean for developers?

Developers are continually looking to maximize their total addressable markets at the lowest possible cost. The development of a new application or new content for a target device always introduces a trade-off between the costs of moving the product to new devices and the benefits of having a bigger target audience. The S60 platform allows developers to have the best of both worlds. It provides for software to be delivered across all devices that are based on a particular edition of the platform. Developers can be certain that applications created using only platform-edition features will be easily transferable to any device based on that edition.

1.6 How is the S60 platform deployed?

The S60 platform is licensed by a number of mobile phone manufacturers that are already shipping devices. These licensees are Lenovo, LG Electronics, Nokia Mobile Phones, and Samsung. The range of S60 devices is constantly growing, and detailed information on the current range of devices can be found in the S60 Phones section of the S60 Web site (<http://www.s60.com/phones>).

1.7 What are the different editions and feature packs of the S60 platform?

The main editions and feature packs of the S60 platform are:

- S60 1st Edition.
 - S60 1st Edition, Feature Pack 1.
- S60 2nd Edition.
 - S60 2nd Edition, Feature Pack 1.
 - S60 2nd Edition, Feature Pack 2.
 - S60 2nd Edition, Feature Pack 3.
- S60 3rd Edition.
 - S60 3rd Edition, Feature Pack 1.
 - S60 3rd Edition, Feature Pack 2.

1.8 What markets are targeted by S60 devices?

The S60 platform is designed to serve the widest consumer and enterprise audience through mass-market devices, while allowing for extremely focused device segmentation in areas such as enterprise, music, imaging, entertainment, personal productivity, and games.

1.9 What user interfaces are available on S60 devices?

The S60 platform does not mandate particular UI characteristics — such as icon styles, screen resolutions, or input methods — as part of its specification. Thus, it allows manufacturers to create devices with individual UI customizations. The S60 platform does have a UI style and APIs as part of its specification; therefore, it is expected that the common UI style will be adhered to in any UI implementation. This is reinforced by the Avkon UI libraries on which all UI implementations are based.

2. S60 platform architecture

This chapter examines the underlying architecture of the S60 platform.

2.1 What is the architecture of the S60 platform?

The architecture of the S60 platform is shown in Figure 1.

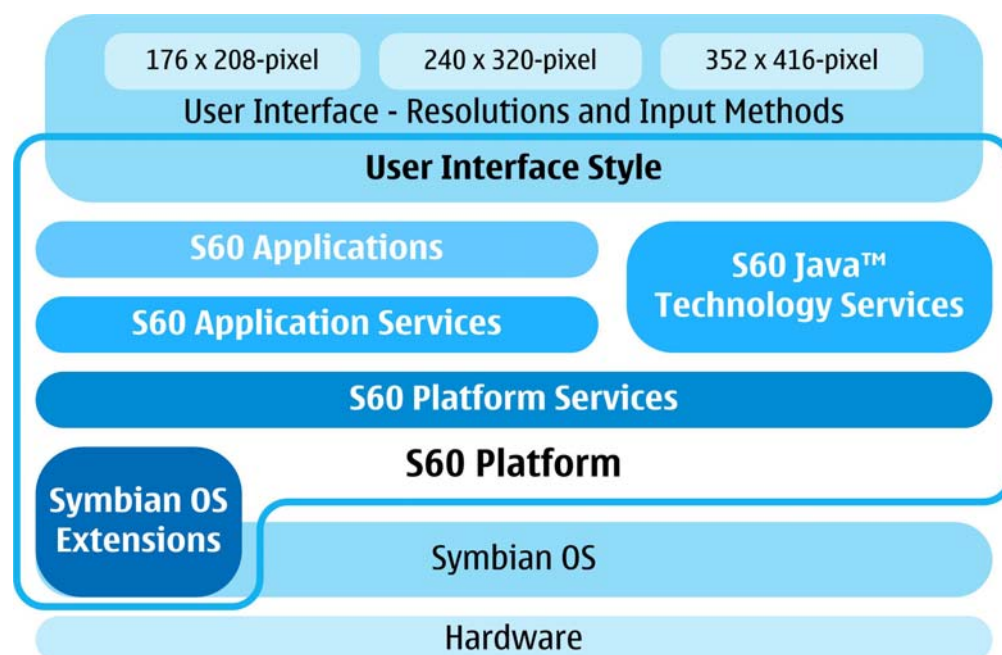


Figure 1: The S60 platform architecture is depicted.

The constituent parts of the S60 platform encompass the S60 UI style, S60 Applications, S60 Application Services, S60 Java™ Technology Services, S60 Platform Services, and Symbian OS Extensions built on top of Symbian OS. These terms are discussed in the following sections.

2.2 What does *Symbian OS* refer to?

Symbian OS is a purpose-built smartphone operating system. It has an impressive range of technology features — even in its earliest versions. Its memory-management and multitasking capabilities allow for safe and efficient operation under conditions of constrained resources that typify mobile devices. Different editions of the S60 platform have used different versions of Symbian OS. Note that not necessarily all features available in a specific version of Symbian OS are implemented in a particular version of the S60 platform: It is always better to follow the S60 guidelines regarding what is available, rather than to assume that a particular Symbian OS feature has been implemented.

2.3 What does *Symbian OS Extensions* refer to?

The *Symbian OS Extensions* are a set of capabilities that allow the S60 platform to interact with device hardware functions such as vibration alert, device lights, and battery charge status.

2.4 What does *S60 Platform Services* refer to?

The *S60 Platform Services* are the fundamental services offered by the S60 platform. These include the:

- Application Framework Services — providing the basic capabilities for launching applications and servers, state-persistence management, and UI components.
- UI Framework Services — providing the concrete look and feel for UI components and handling UI events.
- Graphics Services — providing capabilities for the creation of graphics and their drawing to the screen.
- Location Services — allowing the S60 platform to be aware of a device's location.
- Web-Based Services — providing services to establish connections and interact with Web-based functionality, including browsing, file download, and messaging.
- Multimedia Services — providing the capabilities to play audio and video, as well as support for streaming and speech recognition.
- Communication Services — providing support for local and wide area communications, ranging from Bluetooth technology to voice calls.

2.5 What does *S60 Application Services* refer to?

The *S60 Application Services* provide certain basic functionality for S60 applications. These services are used by the embedded S60 Applications and also are available for use in third-party applications. They include:

- PIM Application Services — providing the basic features of a personal information manager (PIM), including contacts, calendar, and task management, as well as associated functions such as notepad and clock capabilities.
- Messaging Application Services — providing support for various messaging types such as short message service (SMS), multimedia messaging service (MMS), e-mail, BIO messages (smart messaging), and instant messaging (IM).
- Browser Application Services — providing the capabilities to view Web content, including support for Flash Lite from Adobe, video rendering, Scalable Vector Graphics–Tiny (SVG-T) rendering, and audio rendering.

2.6 What does *S60 Java™ Technology Services* refer to?

The *S60 Java™ Technology Services* support the Java™ Platform, Micro Edition (Java™ ME) Java™ Technology for the Wireless Industry (JTWI) (JSR-185) specification. S60 platform support includes the Connected Limited Device Configuration 1.1 (JSR-139) specification and the Mobile Information Device Profile 2.0 (JSR-118) extension to this specification.

In S60 3rd Edition, Feature Pack 2, a subset of the Mobile Service Architecture (JSR-248) is provided. The subset encompasses JSR-118 and JSR-139, as well as PDA Optional Packages for the J2ME™ Platform (JSR-75), Java™ APIs for Bluetooth (JSR-82), Mobile Media API (JSR-135), Mobile 3D Graphics API for J2ME™ (JSR-184), Wireless Messaging API 2.0 (JSR-205), and Scalable 2D Vector Graphics API for J2ME™ (JSR-226).

Further, the S60 Java Technology Services support a range of additional APIs to enable Web services, security and trust services, location information, and Session Initiation Protocol (SIP).

2.7 What does *S60 Applications* represent?

The *S60 Applications* are applications embedded within the platform that are available to a device's user, including personal information manager (PIM), messaging, media applications, and profiles.

2.8 What does the *S60 UI style* represent?

The S60 platform specifies a UI style that defines fundamental UI behavior and component layout, but it does not mandate the screen size or input methods. Licensees are free to implement their own, customized UIs. Developers must program UI applications with scalability in mind, because specific UI dimensions and input methods cannot be assumed.

3. Features of the S60 platform

This chapter examines the underlying operating system, base software, and lead software found in each edition of the S60 platform.

3.1 What is the underlying operating system for each edition and feature pack release?

The releases and their respective operating systems are:

- S60 1st Edition — Symbian OS v6.1.
- S60 2nd Edition — Symbian OS v7.0s.
 - S60 2nd Edition, Feature Pack 1 — Symbian OS v7.0s.
 - S60 2nd Edition, Feature Pack 2 — Symbian OS v8.0a.
 - S60 2nd Edition, Feature Pack 3 — Symbian OS v8.1a.
- S60 3rd Edition — Symbian OS v9.1.
 - S60 3rd Edition, Feature Pack 1 — Symbian OS v9.2.
 - S60 3rd Edition, Feature Pack 2 — Symbian OS v9.3.

3.2 What is included in the base software for S60 1st Edition?

S60 1st Edition includes the following base software:

- Java™ Platform, Micro Edition (Java™ ME) APIs, including:
 - Mobile information device profile (MIDP) 1.0.
 - Connected limited device configuration (CLDC) 1.0.
 - Wireless Messaging API 1.1 (JSR-120).
 - Mobile Media API (JSR-135).
- XHTML/WML browser.
- Multimedia messaging service (MMS) messaging.

3.3 What main features are available in S60 1st Edition, Feature Pack 1?

The introduction of Java™ APIs for Bluetooth (JSR-82) to S60 1st Edition, Feature Pack 1 is a notable addition.

3.4 What is included in the base software for S60 2nd Edition?

S60 2nd Edition includes the following base software:

- Skinning (theme) and digital rights management (DRM) C++ APIs.

- Java™ Platform, Micro Edition (Java™ ME), mobile information device profile (MIDP) 2.0, with the connected limited device configuration (CLDC) HotSpot compiler, providing enhanced application performance.
- Browser supporting XHTML over TCP/IP.
- Messaging software support for multimedia messaging service (MMS) over HTTP transport and advanced presentation capabilities through enhanced synchronized multimedia integration language (SMIL) support.
- Open Mobile Alliance (OMA) Client Provisioning, which allows device settings for services such as browsing, MMS, and over-the-air (OTA) Calendar synchronization. This technology enables easy device configuration and ensures that a user can take full advantage of a device's capabilities.
- DRM via OMA forward-lock.

3.5 What main features are available in each feature pack for S60 2nd Edition?

The following features are the main ones provided by the feature packs for S60 2nd Edition. For more details, please see the Forum Nokia Web site (<http://www.forum.nokia.com/>).

- Feature Pack 1
 - Presence Open and Bluetooth notifier C++ APIs.
 - Java™ technology Wireless Messaging API 1.1 (JSR-120) and Mobile Media API (JSR-135).
 - Support for a megapixel camera with 4x zoom as well as for recording and playback of video clips.
- Feature Pack 2
 - C++ APIs for browser plug-in, connection monitor server, digital rights management (DRM) license manager, simulation file, DRM helper, pictograph, DevASR, speech recognition utility, multimedia framework (MMF) Speech Recognition System (SRS) custom commands, Data Synchronization Profile Listing, content access framework, GIF scaler, Huffman encoding and decoding, message queue, Publish & Subscribe, Location Acquisition, OpenGL ES, EGL, and Symbian XML framework.
 - Mobile 3D Graphics API for J2ME™ (JSR-184) and the FileConnection API and PIM API packages of PDA Optional Packages for the J2ME™ Platform (JSR-75).
 - Support for a 1.3-megapixel camera with 6x zoom, wideband code division multiple access (WCDMA) and EDGE networks, and IPv6.
- Feature Pack 3
 - APIs for feature discovery, scalable icons, browser control, download manager UI library, download manager engine, image transform library, image transform plug-ins, Universal Serial Bus (USB) client driver, and an updated utilities API.
 - J2ME™ Web Services Specification (JSR-172) and Java™ APIs for Bluetooth (JSR-82), with OBEX support.

- Support for scalable UIs (176 x 208-pixel, 240 x 320-pixel, and 352 x 416-pixel screens).

3.6 What is included in the base software for S60 3rd Edition?

In addition to the features delivered in S60 2nd Edition, the following APIs are included in the base software for S60 3rd Edition:

- C++ APIs, including:
 - Location APIs (including Landmark API; Landmark UI Add, Edit, and Select APIs; Landmark Search API; and Basic Location Info Display [BLID] application API).
 - Web services APIs (including WS Connection API, WS Description API, WS Manager API, and XML Extensions API).
 - Session Initiation Protocol (SIP) APIs.
 - Open Mobile Alliance (OMA) digital rights management (DRM) v2 API.
 - OMA Data Synchronization (DS) 1.2 API.
 - Multimedia framework (MMF) DRM API.
 - Light API.
 - Instant messaging (IM) API.
 - IM Application Launch API.
 - Bluetooth 1.2 support in the Bluetooth APIs.
 - Exchangeable Image File (Exif) API.
 - Find Item API.
- Java™ APIs, including:
 - Security and Trust Services API for J2ME™ (JSR-177).
 - Location API for J2ME™ (JSR-179).
 - SIP API for J2ME™ (JSR-180).
 - Wireless Messaging API 2.0 (JSR-205).

3.7 What main features are available in S60 3rd Edition, Feature Pack 1?

The following features are the main ones provided by S60 3rd Edition, Feature Pack 1. For more details, please see the Forum Nokia Web site (<http://www.forum.nokia.com/>).

- Firmware over-the-air (FOTA) updates.
- Nokia Web Browser.
- Improved device management: application delivery and management, settings enforcement, device lock and wipe, and other security features.
- Nokia Push to Talk over Cellular (PoC), as an optional device feature.
- Improved UI customization for licensees and operators via over-the-air (OTA) updates.
- Enhancements to the UI and native applications, including a general settings “control panel.”
- Optional Flash Lite 2.0 from Adobe.
- Configuration options that enable S60 licensees to create lower-cost devices and further boost the volume of S60 devices in the market.
- New C++ APIs, including:
 - Gallery Content Listing API.
 - Optical Character Recognition API.
 - Central Repository Notification Handler API.
 - Profiles Engine API.
 - Screen Mode API.
 - OpenGL V1.1 API (updated).
 - Telnet API.
 - App Framework Animation API.
 - OBEX API (updated).
 - OBEX MTM API (updated).
- New Java™ APIs, including:
 - Advanced Multimedia Supplements (JSR-234).
 - Scalable Vector Graphics 2D API for J2ME™ (JSR-226).
 - Mobile 3D Graphics API for J2ME™ v1.1 (JSR-184) (updated).

3.8 What main features are available in S60 3rd Edition, Feature Pack 2?

The following features are the main additions in S60 3rd Edition, Feature Pack 2. For more details, please see the Forum Nokia Web site (<http://www.forum.nokia.com/>).

- The Nokia Web Browser as the default browser.
- An updated Contacts architecture, enabling the use of multiple data stores and remote address books.
- UI improvements, including a labeled middle softkey and an updated status pane.
- A unified message editor.
- A new media player.
- Optional Flash Lite 2.1 from Adobe.
- New C++ APIs, including:
 - Native support for Open C.
 - Real-Time Graphics and Audio (RGA) APIs.
 - AIW framework APIs, encompassing the AIW Criteria API, AIW Dial Service Consumer API, AIW Generic Parameter API, and AIW Service Handler API.
 - Map framework APIs, encompassing the Map and Navigation API, Map and Navigation AIW API, Geocoding API, and Map and Navigation Provider Discovery API.
 - Connection Settings API.
 - Connection Settings UI API.
- Java™ support for a subset of the Mobile Service Architecture (JSR-248).

3.8.1 Is Flash Lite from Adobe going to be available on all S60 devices from S60 3rd Edition, Feature Pack 1 onward?

No. Flash Lite from Adobe is an optional component of the S60 platform from S60 3rd Edition, Feature Pack 1 onward, and it may not be implemented on all devices. However, end users can install Flash Lite on S60 3rd Edition devices that ship without it.

A Flash Lite Framework API, which allows developers to interact with the Flash Lite 2.0 player and later versions, is present as part of the S60 platform. However, it works only on a device that has the optional Flash Lite 2.0 player or later, either as an original component or installed by the user.

3.8.2 What other improvements does S60 3rd Edition, Feature Pack 1 offer developers?

The S60 documentation delivered in the S60 Platform SDKs for Symbian OS, for C++ has been significantly improved in S60 3rd Edition, Feature Pack 1. The changes, which are in both the structure and content of the documentation, include the following:

- The structure of documentation for each API has been simplified, with a single document providing all the relevant information.
- Each API is documented using the same, common template. Because the placement of material is, therefore, predictable, information is easier to find.
- The documentation for each API is also complete: No sections have been omitted.
- A clear, correct code example is provided for each API.

Overall, these changes should make it easier for developers to find the information they need, and they may be assured that the documentation is complete.

4. S60 3rd Edition

This chapter examines the various characteristics of S60 3rd Edition, such as binary-compatibility breaks, Symbian OS platform security, and the application certification process.

4.1 What are the key concepts and issues for S60 3rd Edition?

There are several important ideas and issues that developers must be aware of with respect to S60 3rd Edition:

- Symbian OS platform security.
- Data caging.
- Trusted computing base.
- Developer self-signing of applications not submitted for Symbian Signed.
- Symbian Signed certification.
- Capabilities.
- Binary-compatibility break.
- Real-time kernel.
- New compiler and build tools.
- Increased device differentiation and segmentation.
- Market volumes.

4.2 What are the costs and benefits of these new features?

Developers will incur various expenses. For instance, new development tools will be required for S60 3rd Edition, although free tools also are available (see question 6.1, “What IDE options are available for C++ developers?”). Furthermore, developers will bear costs of certification that include:

- Obtaining a VeriSign Authenticated Content Signing (ACS) Publisher ID for use in Symbian Signed certification.
- Symbian Signed certification testing. Current testing prices can be found on the Symbian Signed Web site (<https://www.symbiansigned.com/app/page/overview/testhouses>).

The cost of the development tools can be spread across several projects, while Symbian Signed certification incurs per-application charges. Note, however, that developers do not incur certification costs when creating applications that are not Symbian Signed (see question 4.10, “Will all applications need to be signed?”)

The new features of S60 3rd Edition offer the following benefits, among others:

- The new Application Binary Interface (ABI) for the ARM[®] Architecture binary creates applications with smaller memory footprints and more-efficient code, enabling applications to run faster.

- The enhanced platform security reduces the likelihood of corruption or unauthorized disclosure of application data.
- Improved market confidence is resulting from the S60 platform's proactive approach to security issues and the provision of a mechanism for application certification. Cautious consumers will be reassured about applications from previously unknown vendors.
- End users can prevent unsigned applications from being installed on their devices if they are unsure about the sources of those applications.
- More APIs will be available to third-party developers through certification, enabling the creation of a wider range of applications.
- Enhanced digital rights management (DRM) capabilities will make software piracy less likely.
- Single-chip hardware solutions are available for S60 devices, thereby increasing the flexibility of hardware configuration and providing for cheaper devices.

4.3 Does S60 3rd Edition maintain binary compatibility with earlier editions of the S60 platform?

The introduction of the Application Binary Interface (ABI) for the ARM[®] Architecture means that S60 3rd Edition does not maintain binary compatibility with earlier editions of the S60 platform.

4.4 What are the advantages of the Application Binary Interface (ABI) for the ARM[®] Architecture?

The Application Binary Interface (ABI) for the ARM[®] Architecture offers significant performance advantages for devices based on S60 3rd Edition and future editions.

4.5 Does the binary-compatibility break mean higher porting costs?

By itself, the binary-compatibility break is unlikely to result in any significant costs to developers, though applications will need to be rebuilt using different tool sets in order to be available to devices on both sides of the break. The API changes resulting from the implementation of Symbian OS platform security will have more of an impact. Several APIs have been changed or replaced to enable the inclusion of platform security. Coding for these alterations is likely to have the most significant impact on developers building applications that run on S60 3rd Edition and one or more earlier editions.

4.6 What features are included in platform security?

The platform security introduced in S60 3rd Edition has several important aims. It is intended to protect the integrity of the device and to limit access to sensitive data and operations. End users have greater protection from viruses, while operators, licensees, and third-party developers have greater brand and data protection. There are three key concepts involved in platform security.

- *Data caging* — A new directory system allows applications to store data securely with restricted access.
- *Trusted computing base* — The trusted computing base is a collection of software components that enforce 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.
- *Capability model* — A capability grants access to a set of APIs. The ability to use a capability can be obtained through certification, for example, Symbian Signed.

4.7 What are the capability sets?

There are four sets of capabilities:

- *Open to all* — These capabilities cover some 60 percent of Symbian OS APIs. The APIs in this set are sufficient to create stand-alone applications with access to the UI and the ability to store data.
- *Granted by the user at installation time* — These capabilities cover most communications features, personal area networks, Internet access, messaging, and phone calls, as well as the ability to access sensitive user data such as Contacts and Calendar entries. They also cover access to basic location information.
- *Granted through Symbian Signed* — These capabilities include those granted by the user at installation time plus more-advanced device information and control features such as settings, power management, event generation, and extended location information.
- *Granted by the manufacturer* — These capabilities provide fundamental access to the operating system and secured data.

4.8 What are the certification requirements of each set of capabilities?

Each set of capabilities has the following certification requirements for access by applications:

- *Open to all* — These capabilities are open to all applications, whether or not they are certified.
- *Granted by the user at installation time* — These capabilities can be granted by the user to the application on installation, except in the case of Symbian Signed applications, which can access all of these capabilities without requiring user permission.
- *Granted through Symbian Signed* — These capabilities require an application to be Symbian Signed before they can be accessed. In addition, certain capabilities require a declarative statement, explaining why the capabilities need to be accessed, to be provided with the Symbian Signed submission.
- *Granted by the manufacturer* — These capabilities require an agreement with the device manufacturer to allow them to be granted during the Symbian Signed process.

4.9 What are the different signing requirements for an S60 3rd Edition application?

The various signing requirements for S60 3rd Edition applications are shown in Table 1.

Deployment stage	Requirements when seeking Symbian Signed	Requirements when not seeking Symbian Signed
On-device testing	Sign with Symbian Developer Certificate (if capabilities are being accessed)	Sign with self-generated certificate — “self-sign”
Submission for Symbian Signed	Sign with Authenticated Content Signing (ACS) Publisher ID	Not applicable
Granting of Symbian Signed certification	Sign with a single-use ACS Publisher ID that is trusted through the Symbian Root Certificate (by the test house or a self-certifier)	Not applicable
Deployment to users		Sign with self-generated certificate — “self-sign”

Table 1: The various signing requirements for S60 3rd Edition applications are shown.

Details of various signing requirements are as follows:

- *Self-signing* — Content (such as themes) packaged in a *.sis file and applications that are not being submitted for Symbian Signed must be self-signed by the developer. The signature is provided by a key pair (certificate) that the developer creates using tools provided in the S60 3rd Edition SDK. Self-signing does not grant any capabilities to the application. The end user can grant an application access to *granted by the user at installation time* capabilities on installation.
- *Symbian Developer Certificate signing* — An application that requires access to capabilities granted under Symbian Signed or that uses *granted by the user at installation time* capabilities and is to be submitted for Symbian Signed (to bypass user granting of *granted by the user at installation time* capabilities) must be signed with a Symbian Developer Certificate to enable testing on a device during development. These certificates are available for free from the Symbian Signed Web site (<http://www.symbiansigned.com/>).
- *VeriSign Authenticated Content Signing (ACS) Publisher ID signing* — Applications and *.sis packaged content that are to be submitted for Symbian Signed certification must be signed by the developer using its ACS Publisher ID. This signing is to ensure that the test house can verify the sources of the applications or content.
- *Symbian Signed certification signing* — This signature is applied after the application has passed the Symbian Signed tests. The signature is trusted through the Symbian Root Certificate installed on Symbian OS v9 phones. It shows that an application or *.sis content package has passed Symbian Signed and unlocks an application's declared capabilities. Applications and *.sis packaged content can be signed in this manner by a Symbian Signed test house or a developer who has achieved self-certification status. This signature retains the developer's information included in the VeriSign ACS Publisher ID signature.

4.10 Will all applications need to be signed?

Yes. The application installer in S60 3rd Edition requires that all applications and *.sis packaged content be signed in one of two ways. A developer fulfills this requirement either by using a self-signing mechanism (using a certificate created with tools from the S60 3rd Edition SDK or later) or as a result of successfully attaining Symbian Signed certification. Self-signing does not grant any capabilities to the application.

4.11 Will all applications need to be Symbian Signed?

No. Whether an application requires Symbian Signed certification depends on whether or not it accesses capabilities that require certification. Some applications, such as games, may be installed and run without certification. Applications that use *granted by the user at installation time* capabilities — those to which the end user can grant access — will not require certification. They would, however, benefit from certification, since it removes the need for the user to grant access to certain capabilities during installation. In contrast, all applications accessing *granted through Symbian Signed* and *granted by the manufacturer* capabilities need to be Symbian Signed.

4.12 If an application does not require Symbian Signed to work, are there any other reasons it should be Symbian Signed?

There are several advantages to gaining Symbian Signed certification for an application, even when that application can be installed and run on devices without its being Symbian Signed. These include the following:

- *Easier installation* — Obtaining Symbian Signed status for an application means that the user no longer receives a warning message when attempting to install the application and is not prompted to grant (if the application uses them) the *granted by the user at installation time* capabilities. As a result, the installation process is more straightforward, and the user is more likely to install the application because no warning is issued.
- *Greater channel access* — Symbian Signed is a requirement for accessing many sales channels; for example, all native Symbian applications delivered through Nokia sales channels must be Symbian Signed.
- *Improved protection of intellectual property rights* — A Symbian Signed *.sis file incorporates several mechanisms that prevent unauthorized repackaging and distribution of applications and content.

4.13 Can capabilities requiring Symbian Signed certification be accessed during testing?

Yes. The S60 SDK emulator allows security to be turned off to provide unrestricted access to capabilities, so applications can be tested even though they are not Symbian Signed. For device testing, however, a Symbian Developer Certificate is required. These certificates, which are available free of charge, allow an application to be signed for a specific device, thus allowing on-device debugging and acceptance testing to be performed on applications that do not have Symbian Signed status.

4.14 What is involved in obtaining Symbian Signed status?

The Symbian Signed process is as follows:

1. Register on the Symbian Signed Web site (<http://www.symbiansigned.com/>) using the online form and complete the contractual agreement.
2. Using the user name and password supplied, log in to the Symbian Signed account and download the tool for exporting a VeriSign Authenticated Content Signing (ACS) Publisher ID to a standard format for use with Makesis.
3. Sign the application's or content package's *.sis file (with the downloaded tool and Makesis) and submit the *.sis file to the chosen test house, along with the associated *.pkg file and user guide. A receipt will be provided.
4. Once the test house has verified the ACS Publisher ID, it will respond by e-mail with a quote for the testing.
5. Once payment has been made, the tests will take place.

6. Notification will be provided as to whether the application has passed or failed, then:
 - If the application fails, it will need to be modified and submitted again at step 3.
 - If the application passes, it will be re-signed using a single-use ACS Publisher ID that is trusted through the Symbian Root Certificate embedded on S60 devices (Symbian B). The application will then be uploaded to the developer's account, from which it can be collected for distribution.

4.15 Are there any tools to assist with the Symbian Signed process?

Yes. Symbian Ltd., in association with SysOpen Digia Plc, has made available a testing tool, AppTest Lite, that allows applications to be verified against the Symbian Signed test criteria before they are submitted for signing. For more details, see the Symbian Signed Web site (<http://www.symbiansigned.com/>).

In addition to the tools provided, Symbian Ltd. also publishes details of the most common errors made in applications that fail Symbian Signed testing. These details can be accessed at <http://developer.symbian.com/wiki/display/sign/Common+causes+for+failure+during+the+Symbian+Signed+Test+Process>.

4.16 Are there any other tools to help with development in the platform security environment?

Yes. There is a tool to simplify the process of requesting developer certificates. The tool is called the Symbian Developer Certificate Request Wizard Plug-in for Carbide.c++, and it is available as a free download at <http://www.symbiansigned.com/>. Once logged into the Web site, go to **My Symbian Signed** and click **Developer Certificates** for information and download.

4.17 Obtaining Symbian Signed status has a cost, so how are freeware applications going to get to market?

Freeware applications are eligible for the Symbian Signed Freeware Route to Market program. This provides the opportunity for freeware to be Symbian Signed for free. The program covers applications that provide users with full, ongoing access to the applications' features for free. It does not cover demonstration applications, shareware, or other types of licensing that require users to pay to use the full features, nor does it cover applications through which the developer generates revenue by other means, such as adware. Honorware, or donationware — applications for which users are under no obligation to pay for access to the applications' full range of features but may do so if they wish — are covered by the program.

The program is available to free applications and places no restrictions on the developer's business model. Therefore, commercial developers creating freeware have the same eligibility to use the program as open source or hobbyist developers.

5. Development technologies supported by the S60 platform

5.1 What development technologies are available?

C++ is the native programming language of the S60 platform. Developers can use C++ to create applications that access the application engines (including Photos and Phonebook) and that work with a multitude of technologies (such as Bluetooth connectivity, infrared [IR], multimedia, messaging, networking, and telephony).

Applications can also be developed using Java™ MIDP 2.0, Flash Lite from Adobe, and Python, while content can be developed for XHTML browsing and multimedia messaging service (MMS).

5.2 C++ application development

5.2.1 Is standard C or C++ used for application development?

Until the release of S60 3rd Edition, Feature Pack 2, S60 C++ developers used Symbian C++ for application development. The variant of C++ used in the S60 platform has been optimized for small devices with memory constraints. The Standard Template Library (STL) is not supported, and other changes to standard C++ have been made. Native Symbian applications are developed using Symbian Ltd.'s own version of C++, which utilizes most of the C++ idioms but has a unique nomenclature. ANSI C is supported but is rarely required.

From S60 3rd Edition, developers also gained access to a subset of the POSIX libraries through Open C. Open C is an extension of the POSIX libraries for Symbian OS. It provides a subset of functions from nine well-known standard POSIX and middleware C libraries: libc, libdl, libpthread, libm, libz, libcrypto, libcrypto, libglib, and libssl. Open C is a native feature in S60 3rd Edition, Feature Pack 2 and is available as a plug-in for S60 3rd Edition and S60 3rd Edition, Feature Pack 1.

In addition, starting with S60 3rd Edition, Feature Pack 2, the Real-Time Graphics and Audio (RGA) APIs can be used in conjunction with Open C to create applications using standard C/C++ coding methods, without reliance on Symbian OS or S60 native APIs. This feature is targeted primarily at game developers, but it will also be of interest to developers creating applications with rich graphical or audio content.

5.2.2 What types of applications can I produce using the native Symbian and S60 C++ APIs?

Using C++ to develop S60 applications allows developers to realize the full potential of the platform and to produce a wide variety of applications. For example, public APIs allow developers to create applications that manipulate onboard cameras, construct and send multimedia messaging service (MMS) messages, exchange data with other devices using Bluetooth technology, and access the telephony engine.

5.2.3 What opportunities does Open C create for C/C++ developers?

The functions implemented in Open C have been carefully chosen — after extensive analysis of open source projects — to provide a complete set of commonly used functions.

As a result, Open C allows developers to reuse software assets and thereby increase productivity. Such assets may include a developer's own library of functions and open source code. In addition, because C libraries are commonly used to create applications on other platforms, such as Linux or Microsoft Windows, the Open C support significantly simplifies the process of porting existing applications to the S60 platform.

Open C also reduces the knowledge of Symbian C++ required to create S60 applications. Developers now can create much of their applications' business logic using familiar C library functions, while relying on Symbian and S60 APIs to create the application UIs and to provide access to native Symbian and S60 functions.

5.2.4 On which editions of the S60 platform is Open C available?

Open C was introduced as a native feature in S60 3rd Edition, Feature Pack 2, with a plug-in extension for devices and SDKs based on S60 3rd Edition and S60 3rd Edition, Feature Pack 1.

5.2.5 Can I use Open C to create applications without having to learn Symbian C++?

No. Open C needs to be used in conjunction with standard Symbian OS and S60 APIs. These APIs are needed to provide the application with a user interface and to allow the application to access features of the S60 and Symbian OS platform. This approach calls for some knowledge of Symbian C++.

In addition, Open C can be used in conjunction with the Real-Time Graphics and Audio (RGA) APIs on S60 3rd Edition, Feature Pack 2 devices for games and other applications requiring rich graphical or audio content.

5.2.6 What type of applications can I create with the RGA APIs?

The Real-Time Graphics and Audio (RGA) APIs are targeted primarily at game developers, but they will also be of interest to developers creating applications with rich graphical or audio content. The RGA APIs can be used for any application that does not require access to native Symbian OS or S60 APIs and that is primarily graphical in nature.

The RGA APIs will be available in devices based on S60 3rd Edition, Feature Pack 2.

5.2.7 Will my native C++ application be compatible with other S60 devices?

Native C++ applications written for a particular edition of the S60 platform can run on other devices that use the same edition, provided that the applications use no feature pack or lead software APIs. As previously noted, there is a binary-compatibility break between S60 2nd Edition and S60 3rd Edition. Nokia is committed to minimizing compatibility issues between the different editions of the platform, but there are also some known issues between S60 1st Edition and S60 2nd Edition. More information about these issues can be found in Chapter 8, "Known issues."

5.2.8 Is there any information on porting existing applications to the S60 platform?

Yes. The Porting section of the Forum Nokia Web site (<http://www.forum.nokia.com/porting>) contains a series of documents that describe the requirements and considerations involved in porting existing applications from platforms such as UIQ, Palm OS, and Pocket PC, as well as devices in the Nokia 9200 Communicator series.

5.2.9 How are applications downloaded and installed on an S60 device?

There are several ways to download applications. Local PC transfers using Bluetooth connectivity or an infrared (IR) link are possible, and over-the-air (OTA) downloading with a mobile browser is another option. The downloaded applications are installed in the S60 device by an application installer.

5.3 Java™ development

5.3.1 What version of Java™ MIDP does the S60 platform support?

S60 1st Edition supports Java™ Platform, Micro Edition (Java™ ME) technology, connected limited device configuration (CLDC) 1.0, and mobile information device profile (MIDP) 1.0.

S60 2nd Edition supports MIDP 2.0 and CLDC 1.0 with the CLDC HotSpot virtual machine implementation. The S60 implementation of MIDP 2.0 includes support for secure networking.

S60 3rd Edition adds support for CLDC 1.1 as well as for a subset of the Mobile Service Architecture (JSR-248). This support is implemented in S60 3rd Edition, Feature Pack 2.

5.3.2 What Java™ APIs does S60 1st Edition support?

The Java™ APIs supported by S60 1st Edition are:

- The Wireless Messaging API (JSR-120), which supports the sending and receiving of short message service (SMS) messages via GSM and code division multiple access (CDMA).
- The Mobile Media API (JSR-135), which enables applications to use sound, video, and animation playback, and video and audio recording.
- The Nokia UI API, a proprietary Nokia Java API, which provides an interface for sound generation, low-level graphics, and access to the entire screen of a device.

5.3.3 What Java™ APIs does S60 2nd Edition support?

The Java™ APIs supported by S60 2nd Edition include:

- All of the Java APIs supported by S60 1st Edition.
- The Wireless Messaging API (JSR-120), which has been enhanced to support short message service (SMS) Push.
- The Java™ APIs for Bluetooth (JSR-82), for exploiting Bluetooth connectivity.

5.3.4 What Java™ APIs does S60 3rd Edition support?

The Java™ APIs supported by S60 3rd Edition include:

- All of the Java APIs supported by S60 2nd Edition.
- The Security and Trust Services API for J2ME™ (JSR-177).
- The Location API for J2ME™ (JSR-179).
- The Session Initiation Protocol API for J2ME™ (JSR-180).
- The Wireless Messaging API 2.0 (JSR-205).
- A subset of the Mobile Service Architecture (JSR-248) (in S60 3rd Edition, Feature Pack 2).

5.3.5 Does Symbian OS platform security affect Java™ development?

No. Java™ development is not affected by platform security. Only native Symbian C++ applications are affected, because of their inherent low-level capabilities.

5.3.6 Will my Java™ application be compatible with other S60 devices?

Java™ applications should be compatible with devices that use the same edition of the S60 platform, provided that the applications do not make use of any other Java™ Specification Requests (JSRs) that may have been implemented as lead software. There are some variations among the editions of the S60 platform (as detailed in Section 5.3.1, “What version of Java™ MIDP does the S60 platform support?”) that may give rise to compatibility issues. Also, developers must always be aware of issues related to UI scaling and input methods when migrating applications among S60 devices.

5.3.7 Will my Java™ application be compatible with devices based on other platforms?

The Series 40 platform, the Series 80 platform, and the Nokia 7710 widescreen smartphone implement the same mobile information device profile (MIDP) APIs and Java™ Specification Requests (JSRs) that are available on the S60 platform, so a high level of compatibility should be achievable. There will be the usual issues with UI variations and differing memory conditions. For example, Series 40 devices can have smaller screen sizes and more-limited memory than S60 devices.

5.3.8 Will my Java™ application be compatible with other MIDP devices?

The extent of compatibility with other devices that run applications that conform to the mobile information device profile (MIDP) will depend on the supported MIDP version and Java™ Specification Requests (JSRs) on the target device. Developers who want to take full advantage of the Java support provided by the S60 platform may need to change their applications to achieve compatibility with other devices. Developers who want to create applications that can run on the widest range of devices should consider limiting their use of APIs to those that are most commonly implemented.

Obviously, any use of the Nokia UI API will preclude application compatibility with devices based on other platforms.

5.3.9 How are applications downloaded and installed on an S60 device?

There are a number of ways to deploy a MIDlet to a device. The choice of method will depend mainly on the hardware available. The main deployment methods are: Bluetooth connectivity, infrared (IR) link, over the air (OTA) (downloading from a WAP or Web site), e-mail, multimedia messaging service (MMS), or a serial cable.

5.4 Content development

5.4.1 What browsing standards does the S60 platform support?

S60 1st Edition supports:

- WML and WMLScript.
- XHTML Mobile Profile (XHTML MP).
- WAP and related services, such as WAP Push.
- WAP Cascading Style Sheets (WAP CSS).

S60 2nd Edition and S60 3rd Edition support:

- HTTP/1.1 over TCP/IP.
- XHTML over TCP/IP.
- Secure Sockets Layer (SSL) v3.0.
- Transport Layer Security (TLS) 1.0.

5.4.2 What messaging standards does the S60 platform support?

The S60 platform supports multimedia messaging service (MMS). MMS is based on the specifications originally produced by the WAP Forum and 3GPP (<http://www.3gpp.org/>). MMS has a store-and-forward model similar to short message service (SMS), but MMS greatly expands the earlier technology by handling content such as audio clips and pictures. S60 2nd Edition (and later) offers enhanced synchronized multimedia integration language (SMIL) support.

5.4.3 Will browser-based applications for S60 3rd Edition run on devices based on earlier editions?

Yes. The browser available with each edition supports both XHTML Mobile Profile (XHTML MP) and WML. Nokia recommends use of XHTML MP for creating services.

5.4.4 Will MMS content and applications for S60 3rd Edition work with earlier editions of the platform?

Multimedia messaging service (MMS) content should work on any S60 device. S60 3rd Edition uses MMS digital rights management (DRM), which is not available in earlier editions. Also, because S60 1st Edition does not support synchronized multimedia integration language (SMIL), developers need to ensure that content is ordered correctly when it is sent to S60 1st Edition devices.

5.4.5 What is OMA DRM?

S60 2nd Edition supports the Open Mobile Alliance (OMA) forward-lock method of content protection. This means that certain types of files can be protected to prevent them from being forwarded (via the full range of messaging and communications technologies) from a device. This protection applies to a variety of media file types and Java™ MIDlets. Content that has been protected using OMA forward-lock can be received only via HTTP download or multimedia messaging service (MMS). The forward-lock feature is automatically activated when protected content is received.

Applications that handle protected objects must use the encryption services provided by the digital rights management (DRM) engine to store or temporarily save such objects. Most preinstalled terminal applications, such as the Voice Recorder and the Media Gallery, are designed to recognize objects protected by forward-lock.

5.4.6 What is OMA Client Provisioning?

Device settings can be configured via Open Mobile Alliance (OMA) Client Provisioning with a minimum of user interaction. For example, a device can receive a message containing the required setup for a WAP client and configure itself accordingly. Other settings that can be communicated this way include multimedia messaging service (MMS), e-mail, instant messaging (IM), and access points.

The configuration information can be delivered by a variety of mechanisms, including over the air (OTA) or by means of media such as subscriber identity module (SIM) cards.

5.4.7 What enhancements to DRM are implemented in S60 3rd Edition?

The following APIs and features have been provided in S60 3rd Edition with respect to digital rights management (DRM):

- S60 Open Mobile Alliance (OMA) DRM v2 API migration.
- Multimedia framework (MMF) DRM API.
- OMA DRM 2.0 (implementation is device-dependent).
- E-mail synchronization (local and remote over the air [OTA]) based on OMA Data Synchronization (OMA DS) V1.2, e-mail filtering, and time zone support for Calendar.
- Rich Push e-mail (IMAP/POP, OMA e-mail notification, OMA DS, and polling e-mail).

6. Development tools for the S60 platform

6.1 What IDE options are available for C++ developers?

Symbian C++ applications can be created using a variety of integrated development environments (IDEs): Carbide.c++, the CodeWarrior® Development Studio for Symbian OS, Borland C++ Mobile Edition, Microsoft Visual C++ 6.0, and Microsoft Visual Studio .NET 2003.

The recommended tools for C++ development are those from the Carbide.c++ range, a set of tools based on Eclipse. There are four versions, three of which are already available:

- Carbide.c++ Express. This version contains all the tools necessary to create C++ applications with an S60 SDK, a Series 80 SDK, or a UIQ SDK. Carbide.c++ Express is available as a free download from the Forum Nokia Web site's Carbide Development Tools section (<http://www.forum.nokia.com/carbide>).
- Carbide.c++ Developer Edition. This version extends Express with a user interface design tool and the ability to perform on-device debugging.
- Carbide.c++ Professional Edition. This version extends Developer Edition with a tool for application performance investigation and a crash debugger.
- Carbide.c++ OEM. This version will extend Professional Edition with the features and tools required by device manufacturers to create devices based on Symbian OS. It will be made available during 2007.

6.2 What tools are available for C++ development?

SDKs supporting Carbide.c++, the CodeWarrior® Development Studio for Symbian OS, Borland C++ Mobile Edition, Microsoft Visual C++ 6.0, and Microsoft Visual Studio .NET 2003 development environments for C++ development are available for download from the Tools and SDKs section of the Forum Nokia Web site (<http://www.forum.nokia.com/tools>). Each SDK includes:

- S60 APIs.
- An S60 emulator for testing.
- Sample applications.
- Documentation.

SDKs are currently available for editions through S60 3rd Edition, Feature Pack 1. An SDK for S60 3rd Edition, Feature Pack 2, is expected to be made available during the first half of 2007.

Note that use of Microsoft Visual Studio .NET 2003 requires Carbide.vs.

6.3 What Java™ development tools are available for the S60 platform?

The S60 Mobile Information Device Profile (MIDP) SDK is available for download from the Tools and SDKs section of the Forum Nokia Web site (<http://www.forum.nokia.com/tools>). It includes:

- An S60 emulator.
- Java™ libraries.
- APIs (including the Nokia UI API).
- Documentation.

The kit is compatible with professional Java development environments from IBM Corporation, Sun Microsystems, Inc., and Borland Software Corporation. More recent SDKs also support NetBeans and Eclipse integrated development environments (IDEs).

IDEs from Borland and Sun can be used in conjunction with Carbide.j. Carbide.j is also a prerequisite for using Eclipse. This tool accepts SDK plug-ins for each of the supported devices and includes the S60 MIDP SDK.

6.4 What tools are available for content developers?

Carbide.ui S60 Theme Edition for Symbian OS provides graphic designers and artists with a convenient way to create, edit, and package the content of themes for S60 devices. For more details, go to the Tools and SDKs section of the Forum Nokia Web site (<http://www.forum.nokia.com/tools>).

The Nokia Mobile Internet Toolkit (NMIT) provides the tools that developers need to create and test browsing and messaging applications. The NMIT contains, among other tools, a reference implementation browser. For more details, go to the Tools and SDKs section of the Forum Nokia Web site (<http://www.forum.nokia.com/tools>).

Developers of messaging applications will benefit from the Nokia Developer's Suite for MMS (multimedia messaging service), while those working on WAP/XHTML and Push applications should get NMIT. Another resource is the Nokia Mobile Server Services API and Library for developers working with the Nokia multimedia messaging service center (MMSC) and server deployment projects. All of these tools are available free of charge from the Tools and SDKs section of the Forum Nokia Web site (<http://www.forum.nokia.com/tools>).

For multimedia projects, there is the Nokia Audio Suite, a tool set for developers of SP-MIDI (Scalable Polyphony) ring tones.

6.5 Are all of Nokia's tools available to all developers?

Nokia offers developers the opportunity to join Forum Nokia PRO. One of the benefits of the Forum Nokia PRO program is that members get early access to new and updated tools. Once its early-access period on Forum Nokia PRO has expired, each tool is made available through the Tools and SDKs section of the Forum Nokia Web site (<http://www.forum.nokia.com/tools>).

7. Developing for the S60 platform

7.1 Why target the S60 platform for initial development of mobile applications?

The mobile device market has expanded dramatically in terms of design variation and technology adoption since the introduction of the first smartphones. The S60 platform has occupied a leading position within the marketplace since the introduction of the Nokia 7650 imaging phone in 2002, and a profusion of new-device releases and announcements from a range of S60 licensees has followed.

For two important reasons, the S60 platform is the ideal starting point for developing mobile applications. First, it is the leading platform within the smartphone market, with numerous devices available. There is, therefore, an established market for new applications, with millions of end users who regularly access sales channels.

Second, software developed for the S60 platform is highly portable. For example, it is relatively easy to migrate applications designed for the S60 platform to other Nokia platforms. There are also other platforms based on Symbian OS, such as UIQ, to which S60 applications can be ported with changes being required primarily for the UI, and this is true for both native applications (written in C++) and MIDlets. There is literature available that explains how to port from the S60 platform to those platforms — as well as to the S60 platform from BREW[®], Microsoft Pocket PC, and Palm OS — in the Porting section of the Forum Nokia Web site (<http://www.forum.nokia.com/porting>).

7.2 What issues are associated with optimizing applications across the range of S60 devices?

The major sources of variation for devices based on the S60 platform are UI, platform editions, feature packs, and lead software.

The S60 platform does not mandate a particular UI resolution, display orientation, or input method as part of its specification (although the UI style is part of the platform). Developers should therefore create applications with flexibility and scalability in mind and use features such as the feature discovery API wherever possible.

Developers may choose to use features provided by an edition, a feature pack, or lead software that will not be available across the entire S60 platform. Before using the features provided by an edition, a feature pack, or lead software, developers should find out which devices the resulting applications will run on. Such choices will, of course, limit the extent of those applications' addressable markets. However, by becoming acquainted with the technologies available in each edition and designing to allow for feature differences, developers can minimize the effort involved in migrating applications within the S60 platform's editions.

The introduction of S60 3rd Edition, with its binary-compatibility break from the earlier editions and API changes for Symbian OS platform security, will require more effort from developers to create applications for new and existing devices. Careful application design will minimize the code changes between editions, though all applications will need to be compiled using the tool chain appropriate to the edition.

There are some known issues concerning binary compatibility between the first two editions of the S60 platform. (See Chapter 8, "Known issues," for details about relevant literature.)

7.3 What issues are associated with migrating applications to other Nokia platforms?

Nokia produces two other platforms, the Series 40 platform and the Series 80 platform, and the company has a device, the Nokia 7710 widescreen smartphone, that is based on Symbian OS. All of these are capable of running applications based on Java™ MIDP. All of them, except for the Series 40 platform, are based on Symbian OS and are capable of running applications written in native Symbian C++.

The major issues that will be encountered by developers who want to migrate applications to these platforms will be the differences in UI styles and the underlying APIs, differences in memory constraints (especially with Series 40 devices), and differences in implemented technologies. For more information, see the Forum Nokia Web site's Platforms section (<http://www.forum.nokia.com/platforms>).

It should be noted that there are differences in screen orientation between the landscape-style UI on Series 80 devices and the Nokia 7710 smartphone and the portrait-style UI on Series 40 devices. All editions of the S60 platform support a portrait-style UI, but from S60 3rd Edition onward, landscape support for screens with certain resolutions is also available. This may mean that a UI designed to use the width available on Series 80 devices and the Nokia 7710 smartphone may require significant redesign to successfully operate on S60 devices and Series 40 devices. If a high degree of UI consistency is required between different device versions of the same application, then the initial UI design stage will have to take these factors into account.

7.4 What issues are associated with porting applications to other devices based on Symbian OS?

All devices based on Symbian OS are capable of running applications written in native C++ or Java™ MIDP, so a certain amount of compatibility can be expected. The most common source of differentiation can be found in the UI, but developers should investigate the range of differences in the capabilities and technologies as well.

7.5 What issues are associated with porting applications to devices based on unrelated operating systems across the mobile market?

The natural choice to ensure the highest level of cross-platform compatibility is to develop applications in Java™ MIDP, but the choice of development language will depend on the capabilities required by the application.

Applications written in Symbian C++ will probably need to be completely rewritten for platforms that use other operating systems. However, because Symbian OS supports development in ANSI C, it is possible to write an application engine for the S60 platform that is independent of the operating system and then use that engine as the basis for applications developed for other operating systems. This development approach has been considerably enhanced with the Open C support available for S60 3rd Edition. Open C, which makes a range of functions from POSIX and middleware C libraries available to S60 developers, significantly simplifies the design of portable application engines. Building a platform-independent application engine requires developers to understand the limitations of all their target platforms and to create engines that accommodate all these limitations.

8. Known issues

8.1 What are the known issues that S60 application developers should be aware of?

Nokia is committed to maintaining compatibility among the different editions of the S60 platform and among the various platforms wherever possible.

Where compatibility is not achieved, Nokia will maintain and publish a record of any issues, such as minor differences in API implementations or problems with technology implementations. This record, along with details of technical solutions (comprising mainly developers' questions that have been answered by Forum Nokia Professional Support) are available in the Forum Nokia Technical Library on the Library page of the Forum Nokia Web site (<http://www.forum.nokia.com/library>). The Forum Nokia Technical Library is available online and as a download in Microsoft HTML Help and WebHelp formats.

9. Business case

9.1 What is the business case for using the S60 platform?

The S60 platform is the established leader in the smartphone market, making it an important source of revenue for third-party application and content developers. S60 3rd Edition significantly extends the business opportunities and introduces major enhancements designed to provide long-term opportunities for developers.

The S60 platform gives the developer community access to industry-standard technologies and a market that can be measured in tens of millions of consumers. The range of S60 devices offers opportunities both for horizontal applications and for market segmentation in enterprise, games, music, video, personal productivity, and other sectors.

The opportunities extend beyond S60 devices, because the platform provides standard technologies that allow developers to build applications and content for Series 40 devices, Series 80 devices, and the Nokia 7710 widescreen smartphone, as well as devices provided by other manufacturers.

The S60 platform is well-supported by development tools and documentation provided on the Forum Nokia Web site (<http://www.forum.nokia.com/>) and by a number of tools companies. Established sales channels help developers realize their investments as quickly and easily as possible.

There are three compelling reasons for developers to use the S60 platform as the basis for mobile applications and content.

First, users of S60 devices are significant consumers of applications and content.

Second, Nokia alone has shipped more than 70 million S60 devices cumulatively (as of October 2006). More than a dozen S60 3rd Edition device models have been launched, and S60 devices are sold by the majority of retail operators worldwide. The S60 platform is licensed to four device manufacturers. All these factors mean that the S60 platform offers a significant and growing market of potential customers.

Third, the platform approach allows developers to create content for one device or a range of devices that can, with minimal additional development work, be optimized to work with other devices that incorporate platforms. This allows developers to multiply the potential market for their applications or content, with only a small incremental investment in optimizing or migrating their offerings.

10. Getting applications to market

10.1 How can Nokia help get my application to market?

Nokia connects mobile application developers to the market in multiple ways.

The Nokia Content Discoverer client is an on-device content portal that makes it easy for mobile consumers to discover, download, and purchase great content and applications. With the ability to integrate with multiple content-delivery systems, the client helps operators maximize mobile application and content sales.

Nokia also offers developers lucrative opportunities to sell Symbian Signed applications and content, and Java Verified™ applications through Nokia sales channels: Nokia Software Market and Nokia Catalogs. In addition, some applications may be chosen to be embedded in new Nokia devices.

For full details, please follow the Marketplace link in the Developers section of the Nokia Web site (<http://www.nokia.com/developers>).

11. 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](#).