
Series 60 Developer Platform 1.0/2.0: Basics

Version 1.0
April 2, 2004

DEVELOPER PLATFORM
60
SERIES

Legal Notice

Copyright © 2004 Nokia Corporation. All rights reserved.

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. 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	6
1.1	Purpose and Scope.....	6
1.2	The Series 60 Developer Platform.....	6
1.3	Series 60 Developer Platform Versions	6
1.3.1	Series 60 Developer Platform 1.0	6
1.3.2	Series 60 Developer Platform 2.0	7
2.	Market Opportunities with the Series 60 Developer Platform	8
2.1	What is the Target Audience for the Series 60 Developer Platform	8
2.2	Optimizing Applications Designed for the Series 60 Developer Platform.....	8
2.2.1	Series 60 device range.....	9
2.2.2	Other Developer Platforms	9
2.2.3	Other devices based on Symbian OS	9
2.2.4	The broader market.....	9
3.	Series 60 Platform Features	10
3.1	Target Platform Requirements.....	11
3.2	Application Installation and Management	12
3.3	Download of Content.....	12
3.3.1	Summary of supported content formats	12
3.4	WAP Browsing	13
3.5	Messaging.....	13
3.5.1	Multimedia Messaging Service (MMS).....	13
3.5.2	Short Message Service (SMS).....	13
3.5.3	POP3/IMAP4/SMTP e-mail	13
3.5.4	Nokia smart messaging.....	13
3.6	SyncML	13
3.6.1	Series 60 SyncML elements.....	14
3.6.2	SyncML requirements.....	14
4.	User Interface Styles for the Series 60 Developer Platform.....	16
5.	Symbian Ltd.....	18
5.1	Symbian OS	18
6.	Series 60 Developer Platform Technical Overview	21
6.1	Avkon UI Library.....	22
6.2	Applications.....	23
7.	Developing Applications	24
7.1	MIDP Java Development	25
7.2	C++ Development	26

7.3	Comparison of Java and C++ Development.....	26
7.3.1	Symbian OS C++.....	26
7.3.2	Java MIDP.....	27
7.4	WAP Content Development.....	27
7.4.1	Interactive data applications.....	28
7.4.2	Other graphical user interface elements.....	29
7.5	Messaging.....	29
7.5.1	MMS messaging.....	29
7.5.2	E-mail.....	30
7.6	Bluetooth.....	30
7.7	SyncML.....	30
8.	Platform Compatibility.....	31
9.	Developer Resources.....	32
9.1	Forum Nokia.....	32
9.2	Symbian Developer Network.....	32
9.3	Books.....	33
9.4	Other Sources.....	33
10.	Terms and Abbreviations.....	34

Change History

Date	Version	Comment
April 2, 2004	1.0	Replaces the <i>Series 60 Platform Basics</i> document

1. Introduction

This document is an introduction to the Series 60 Developer Platform for potential developers and content providers who are new to wireless development.

As there are actually two main versions of 'Series 60' – Series 60 Developer Platform 1.0 and 2.0, for brevity, this document will utilize a generic naming approach and make reference to specific differences between the variants where necessary.

1.1 Purpose and Scope

Series 60 devices are based on Symbian OS, an open, robust, 32-bit multitasking operating system designed for data-enabled mobile phones. Introductions are provided to the Series 60 Developer Platform, Symbian OS, the technologies available, and the associated development options. In addition, information about further sources of information is provided.

This document provides foundation information for readers. It is assumed the reader has no prior specific knowledge of the Series 60 Developer Platform or Symbian OS, but is familiar with some or all of the related wireless technologies.

1.2 The Series 60 Developer Platform

The Series 60 Developer Platform is a complete smartphone software package that provides a mandatory base of technology implementations. The Series 60 Developer Platform enables application and service developers to produce smartphone products for the Series 60 Platform. While the term "Series 60 Platform" encompasses the technologies of the relevant version of the Series 60 Developer Platform on which it is based, it also provides an optional range of lead software that licensees may wish to support. Devices based on the Platform therefore provide the developer with a guaranteed set of technologies while allowing the manufacturer to differentiate between devices by choosing the lead software options best tailored to the target consumer segment. By providing a standardized platform while maintaining a high degree of flexibility, the Series 60 Developer Platform is well positioned to maintain its market leader status in the smartphone sector.

1.3 Series 60 Developer Platform Versions

As previously stated, there are two versions of the Series 60 Developer Platform. These are discussed below with a brief outline of the technologies available on each.

1.3.1 Series 60 Developer Platform 1.0

Series 60 Developer Platform 1.0 contains the following technologies:

- J2ME™ Java™ APIs
 - Mobile Information Device Profile (MIDP) 1.0
 - Connected Limited Device Configuration (CLDC) 1.0
 - Wireless Messaging API (JSR 120)
- Mobile Media API (JSR 135)

- XHTML browsing
- MMS messaging
- Symbian OS 6.1 native APIs

1.3.2 Series 60 Developer Platform 2.0

Series 60 Developer Platform 2.0 contains the following technologies:

- J2ME Java APIs
 - MIDP 2.0
 - CLDC 1.0
 - Wireless Messaging API (JSR 120)
 - Mobile Media API (JSR 135)
 - Bluetooth API (JSR 82)
- XHTML browsing over TCP/IP
- MMS messaging with Synchronized Multimedia Integration Language (SMIL)
- OMA Digital Rights Management (DRM) (forward-lock)
- OMA Client Provisioning
- Symbian OS v7.0s native APIs

2. Market Opportunities with the Series 60 Developer Platform

The success of smartphone category devices is highly dependent on the availability of innovative applications and content from third parties – that is, the growth of the mobile services and applications businesses. End users are becoming accustomed to downloading content and applications onto their devices. The perceived value of terminals will therefore be influenced by the availability of high-quality services, applications, and content.

Diversification between handset designs and capabilities has greatly increased. This has resulted in minimal similarity amongst competitive devices in terms of screen size, keypad, browser, and other elements of the user interface. Applications, services, and other content have to be adapted to these different devices.

Nokia has made the Series 60 Platform available for licensing by other handset manufacturers enabling them to bring phones to market with equivalent and compatible functionality. Standardizing the application environment helps service creation and application interoperability. Common input methods, APIs, and supported technologies allow services and applications to interoperate seamlessly, but still give licensees the freedom to innovate and design excellent smartphones.

Benefits for Series 60 developers include:

- A larger market – increasing potential revenues
- A market that is accessible through one common platform – lowering development costs
- Wider availability of applications – fueling demand for terminals, creating a virtuous circle

2.1 What is the Target Audience for the Series 60 Developer Platform

The Series 60 Developer Platform allows manufacturers to target the smartphone market segment. The underlying approach to the design of the platform is that of a fully connected mobile device enhanced with PDA capability. Cutting-edge technology is available on this platform and it is continually being expanded as new standards are established. It therefore targets all sub-segments of the smartphone market sector including enterprise, gaming, and music and media.

2.2 Optimizing Applications Designed for the Series 60 Developer Platform

As the mobile sector has evolved, it has produced a greater diversity of devices in terms of capabilities, input methods, and physical design than most developers accustomed to creating applications for PCs have seen before. Some devices are consciously targeted at specific audience sectors, yet most of them retain some standardized capabilities to allow the installation of new applications. It is obviously in the interests of the manufacturers that their devices are capable of running popular and successful applications, while developers will naturally wish to maximize the range of devices featuring their applications. The Series 60 Developer Platform is the ideal starting point for the development and subsequent optimization of new applications as it has an extremely broad combination of features, capabilities, development environments, and existing devices within the smartphone market

segment. It has been licensed to a number of manufacturers and has an exciting range of differentiated devices that feature common standards.

2.2.1 Series 60 device range

While some Series 60 functionality is standard across the whole Series 60 device range, there are different platform versions and software options within those versions that can cause substantial differentiation between devices. It is always best to consult different device specifications and other documentation before beginning development. However, it is possible to write applications that will be able to run on several Series 60 devices with no (or only minor) adaptations. One important point to note is that the user interface is NOT part of the Series 60 Developer Platform and developers must, therefore, expect variations in the UI implementation across the Series 60 device range.

2.2.2 Other Developer Platforms

Nokia has produced three other Developer Platforms that share some similarities with the Series 60 Developer Platform; these are the Series 40 Developer Platform, the Series 80 Developer Platform, and the Series 90 Developer Platform. All of these are capable of running applications based on Java MIDP. All of them, except for the Series 40 Developer Platform, are based on Symbian OS and capable of running applications written in native C++. In addition to the expected differences in user interfaces and keypad configurations, there are substantial differences in the capabilities of devices of different developer platforms.

2.2.3 Other devices based on Symbian OS

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

2.2.4 The broader market

The natural choice to ensure the ease of cross-platform compatibility is to develop applications in Java MIDP, but the choice will depend on the capabilities required by your application. It may be that you will need to write applications in C++ for devices based on Symbian and then completely rewrite them in the appropriate native language (or language variant) for other platforms.

3. Series 60 Platform Features

The Series 60 Platform is a complete smartphone reference design that includes a host of wireless applications. The platform builds on the Symbian OS, complementing it with a configurable graphical user interface library.

The term "Series 60 Developer Platform" represents a base set of technologies that are a mandatory part of any Series 60 device based on the particular version of the platform; the term "Series 60 Platform" encompasses the technologies of the relevant version of the Series 60 Developer Platform, it also provides an optional range of technologies that licensees may wish to support. Due to the optional nature of the lead software available with Series 60 Platform, versions of the Series 60 Developer Platform have greater longevity than versions of Series 60 Platform. Series 60 Developer Platform is specifically aimed at application and service developers as it provides a reliable base of software that will typically last through several versions of the Series 60 Platform..

This chapter describes the features available in the Series 60 Platform. All features are common to both versions unless otherwise indicated.

A comprehensive suite of reference applications is provided including:

Phone application	Phone Features – Call creation, incoming call, active call, call timers, advice of charge, voice call barring, hold, answer, reject, transfer calls, three-way calling, swap between calls, reject calls, mute/un-mute, send predefined DTMF tone, call forwarding (divert), emergency calling, and other features are supported.
Telephony services and features	Voice Recorder, Telephone Settings, Voice Dialing, and Voice Tags. Speed Dialing – Assign keypad buttons 2 through 9 for user-defined phone numbers. Call Logs and Message Indications – Display name, number, call indication, and picture; show list of received calls, dialed calls, missed calls, call costs, and call summary after the call. User Profiles – User can set name, ringing tone, sound volume, sound type, VIP group, personal tone, vibrate, received message, key click, and notification and warning tones. Profiles can be changed, personalized, and re-named.
Phonebook	Contact database integrated with messaging and other applications, supports vCard.
Calendar	Scheduling application, supports vCalendar data format.
To-do list	vCalendar items can be downloaded into this to-do list.
WAP browser	WAP 1.2.1 browser with features such as Push, WTAI, and enhanced security
Notepad	For creating text notes
Photo album	Store for multimedia messages and images.
Pinboard	Application management and organization tool for the multitasking environment.

Clock, calculator, unit converter	World clock, business calculator etc.
Composer	Lets users compose new tunes for applications.
Messaging, i.e., e-mail, SMS, and MMS	Client software for messaging applications.
Java J2ME	Platform 1.0 implements Java MIDP 1.0 and Platform 2.0 implements MIDP 2.0, both over CLDC 1.0
Camera Support	Series 60 Platform 2.0 implements support for an onboard camera API via a stub. This is due to the fact that the camera hardware implemented by the manufacturer will require its own set of unique drivers.
Multimedia	Handling of many formats implemented via the Media Server in Platform 1.0 and by the Multimedia Framework in Platform 2.0
Application installation	Installing new software via the PC Connectivity suite or OTA. Platform 1.0 has separate .sis and Java installation utilities, Platform 2.0 provides a new, unified Application Manager for Symbian Apps.
Synchronization	Platform 1.0 implements a SyncML 1.0.1 synchronization engine and supports data transfer over mobile networks (WAP), Bluetooth, and infrared (IrDA). vCalendar 1.0 and vCard 2.1 data formats. Platform 2.0 supports SyncML 2.0 standards with supported elements at revision 1.1.1.
Security	Security settings and software certificate management.

Other platform services include a short-range connectivity OBEX engine, PC-connectivity, Bluetooth, IR stacks, and a SyncML synchronization engine.

A set of robust components and many varied APIs are provided in the platform. The suite of “standard” applications uses the APIs supplied extensively, but they are designed to be re-used by application and service developers as well.

3.1 Target Platform Requirements

The following are specific *minimum* requirements for a device being targeted for a port of the Series 60 Platform:

Display	176 x 208 pixel, 256 color display.
Input	Two soft-keys, five-way navigation, an application launching and swapping key, as well as Send and End keys. To improve and facilitate text input, it includes a Clear key and an alpha toggle key. It uses a standard 12-key number keypad with alpha printings.
Processor	It is recommended that the target device use a 32-bit ARM processor.
Code Size (ROM)	16 MB
RAM Usage	8 MB

3.2 Application Installation and Management

An installation engine is provided as part of the Series 60 Platform that supports adding and removing applications to the platform via PC connectivity or OTA downloads. A tested interface linking the browser to a standard J2ME/MIDP application execution environment is also included, ensuring the ability to download additional applications and content. In Platform 1.0, these installation utilities were separate. In Platform 2.0, both .sis and Java MIDlets are handled via a new, unified application installation utility.

Series 60 is an open platform and as such presents a potential security risk to malicious applications and content. Java applications are run with security restrictions and Symbian native applications can be packaged in certified installation files. Symbian's secure software installation system allows users, prior to installing software, to identify the software vendor and verify that the installation file has not been tampered with since it was created. This functionality is particularly important within an environment in which there is easy access to a wide range of freely downloadable software (that is, that might be infected with viruses). The Certificate Generator can create a public/private key pair. These are used by the Installation File Generator to create a digitally signed installation file.

Applications can be added, removed, selected, and launched from the Pinboard, which is a new application management tool. The Pinboard lets users organize 25 icons representing different applications from the application menu. It is also possible to immediately determine which applications are currently in use through a pop-up menu that displays them in a vertical list. The Pinboard can be used to store URLs and documents.

3.3 Download of Content

Messaging-based download for relatively small and inexpensive content items is supported by the Series 60 Developer Platform. As such, Series 60 will inherently support the handling of multiple MIME content types. All types supported will be accessible through a URL on a WML page, as well as through WAP Push messages, MMS messages, e-mail, IrDA, Bluetooth, and PC-based provisioning.

Digital Rights Management enables the protection of downloaded content from misuse, thus allowing operators to gain as much value as possible from their services (early releases may not include full DRM support). Platform 2.0 implements only forward-lock functionality.

3.3.1 Summary of supported content formats

- Pictures: a wide range of formats including WBMP, JPEG, PNG, and GIF image formats
- Java MIDlets: JAD and JAR files
- Series 60 Native Symbian Applications: Symbian OS installation file (.sis) format
- Audio files: .wav, .amr
- vCard and vCalendar
- Ring Tones, Operator Logo Icons, Calling Line Identification Logos: as defined in Nokia Smart Messaging 3.0.0

3.4 WAP Browsing

The Series 60 Developer Platform contains a WAP 1.2.1 browser with features such as Push, WTAI, and enhanced security. Some of the key features of the Series 60 Mobile Browser are:

- Push functionality with Service Indication capability that enables application and content providers to push information and services to wireless terminals
- Telephony integration supporting the public WTAI functions for making a call, adding an entry into the device's phonebook, and the sending of dialing tones during a call

3.5 Messaging

All messaging activities in the Series 60 Platform are concentrated into a single application entitled "Messaging". It offers common functionality to all supported message types. This includes the composition and viewing of messages, as well as the editing of messaging settings.

3.5.1 Multimedia Messaging Service (MMS)

Multimedia Messaging Service (MMS) is based on WAP Forum and 3GPP specifications. MMS is used for user-to-user messaging, storing and forwarding. The MMS client complies with the following specifications:

- WAP Forum specifications, WAP205-WAP209
- R99 3GPP specifications, TS 22.140 (Multimedia Messaging Service, Stage1) and TS 23.140 (Multimedia Messaging Service, Stage 2)
- MMS Conformance Document, from Nokia and Sony Ericsson

3.5.2 Short Message Service (SMS)

The Series 60 Developer Platform includes an SMS client capable of sending long SMS messages (up to 450 characters) in a single user-interface operation. The client will be able to receive long SMS messages (up to 450 characters).

3.5.3 POP3/IMAP4/SMTP e-mail

The Series 60 Developer Platform contains a standard POP3, IMAP4, and SMTP e-mail client provided as part of the Symbian operating system.

3.5.4 Nokia smart messaging

The Series 60 Platform includes a Nokia Smart Messaging Specification 3.0.0 client.

3.6 SyncML

SyncML is an XML-based technology for universal data synchronization of networked devices. Ericsson, IBM, Lotus, Motorola, Matsushita Corporation, Nokia, Openwave, Starfish Software, and Symbian are the sponsoring members of the SyncML initiative, and drive the development of the technology together with Promoter

members. The objective of the SyncML technology is to enable synchronization of any networked data with any mobile device and to ensure seamless interoperability between devices.

SyncML is designed for use between mobile devices that are intermittently connected to the network and network services that are continuously available on the network. However, SyncML can also be used for peer-to-peer data synchronization. SyncML is specifically designed to handle cases where network services and mobile devices store the data in different formats or use different software systems.

To ensure interoperability, SyncML describes how common data formats are represented over the network. SyncML permits the definition of new data formats as needs arise, ensuring extensibility. Operators will be able to offer a common interface to their customers, regardless of the type of mobile device. First implementations of SyncML enable users with a SyncML-enabled device to always have an up-to-date calendar and contacts database.

3.6.1 Series 60 SyncML elements

The Series 60 Platform implements a SyncML client agent and a synchronization engine. In Platform 1.0 the engine is based on the SyncML 1.0.1 specification and in Platform 2.0 on the SyncML 1.1.1 specification. The engine can be used for Over-the-Air (OTA) synchronization. OTA synchronization works over the HTTP and WAP protocol stacks: initially on WSP, and subsequently on HTTP.

The features for OTA synchronization on the Series 60 Platform are:

- Client-initiated two-way synchronization of contacts (object type vCard 2.1)
- Client-initiated two-way synchronization of calendar events and to-do lists (object type vCalendar 1.0)

Contacts format is vCard 2.1; vCalendar 1.0 for calendar. Contact cards can be exchanged via Bluetooth, IrDA, and e-mail.

Initially OTA synch is supported (over WAP 1.2); later versions of the Series 60 Developer Platform will also support local sync (Obex 1.2 over Bluetooth and IrDA). Second-phase enhancements to synchronization may include back up, restore and server-initiated synchronization.

3.6.2 SyncML requirements

A SyncML-compliant server with a SyncML server agent and a synchronization engine is required to complete the end-to-end system. The typical solution comprises a database server and an application server. The SyncML server is integrated with other network elements, such as authentication, billing and profiling, in an operator 's network.

A user normally initiates the synchronization session from the terminal. A data call connection (CSD or GPRS) is then established between the SyncML client and the SyncML server. The data interchange begins when the client has been authenticated. The SyncML server manages the synchronization process, during which the following takes place:

- New data is uploaded to the terminal or the application server
- Deleted data is removed from the terminal or the application server
- Modified data is updated in the terminal or the application server

Unmodified data is not exchanged, thereby saving time and precious bandwidth. When the operation is completed, both the server and the client update their log files to keep them up-to-date for the next synchronization session.

4. User Interface Styles for the Series 60 Developer Platform

A user interface is not part of the Series 60 Developer Platform specification. Manufacturers are therefore free to create their own customized user interfaces. Developers should not make assumptions that all Series 60 devices have identical user interfaces and should therefore program with scalability and flexibility in mind. It should be emphasized that the range of available user interfaces is not static and will grow and develop as new devices from a number of manufacturers come to market. Having made these important points, this chapter will present a brief overview of an existing UI style used by Series 60 devices based on the AVKON libraries.

The user interface style of an application for devices using the AVKON libraries typically starts with dividing the structure into browsing elements and detailed views. For example, the browsing view allows the user to select one entry from a list of elements, whereas the grid view (see Figure 1) allows selection of elements arranged in a two-dimensional array. When the user activates an entry, the respective detailed view displays the data. A wide selection of user interface elements is generally available for a developer, ranging from list boxes, standard dialogs, pop-up menus, check boxes, and radio buttons to rich text and color graphics.

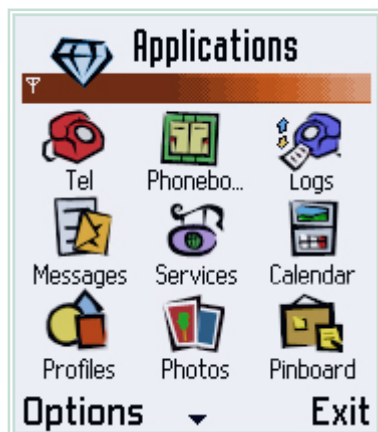


Figure 1: Application grid view

Typically for devices using this user interface, many reference applications are supplied – some have ready-made views available to application developers. For example, the Contacts (Phonebook) application (Figure 2) can be called on to display a list of contacts for selection.

The views shown in Figure 2 show typical approaches to application layout and design with respect to displaying options and information. Note the use of tabbed dialogs and scrollable option lists with only a few options available at the same time.

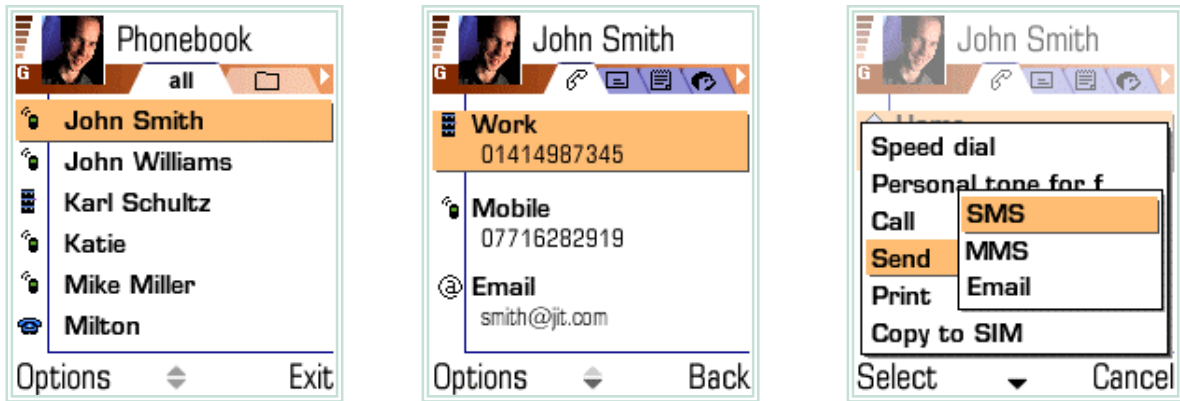


Figure 2: Application Views

5. Symbian Ltd

Symbian focuses on a single objective: developing operating systems for advanced mobile terminals. With its commitment to open standards and the support of major mobile players, Symbian is acknowledged as the industry's first choice for smartphones and communicators. Formed in 1998 and owned by the leading players in the mobile device industry (Psion, Nokia, Sony Ericsson, Panasonic (Matsushita), and Siemens), the shareholders are also licensees of Symbian OS. In addition, because Symbian OS is an open platform, any manufacturer can license it.

The once-new technologies of fax, e-mail, and Internet each saw little growth until the number of users reached critical mass. By fully embracing open standards wherever possible, Symbian is ensuring that the full potential of the mobile services market is achieved as rapidly as possible.

5.1 Symbian OS

Central to the success of the Series 60 Developer Platform is the Symbian OS, indeed, it is the foundation of the product. Symbian OS is a 32-bit multitasking operating system, where events often happen asynchronously and applications are designed to interact with one another. For example, a phone call may interrupt a user composing an e-mail message, a user may switch from e-mail to a calendar application in the middle of a telephone conversation, or an incoming SMS may trigger the user to access the contact database and forward the SMS. By complying with the platform architecture and software design guidelines, application designers can routinely manage such occurrences in the daily lives of smartphone users.

From the start, Symbian OS was designed for small, resource-constrained devices with wireless communications. Key design features are:

- Performance – Symbian OS is designed to make minimal demands on batteries and to have low memory footprint
- Multitasking – telephony and universal messaging are fundamental components. All applications are designed to work seamlessly in parallel
- Standards – the use of technologies based on agreed-upon standards is a basic principle of Symbian OS, ensuring that applications are robust, portable, and interoperable
- Object-oriented software architecture
- Memory management optimized for embedded software environment
- Runtime memory requirements are minimized – very small executable sizes and ROM-based code that executes in place
- Security mechanisms for enabling secure communications and safe data storage
- Application support for international environment with built-in Unicode character sets
- A rich and varied API allowing access to reusable components in developer applications.

As with any software development, Symbian OS is evolving. Symbian has placed a framework around these OS developments and termed them 'Generic Technology.'

Thus, Series 60 Developer Platform 1.0 is based on Symbian Generic Technology Version 6.1, and Platform 2.0 is based on Generic Technology Version 7.0s. A basic representation of the Symbian OS Generic Technology layering is shown in Figure 3.

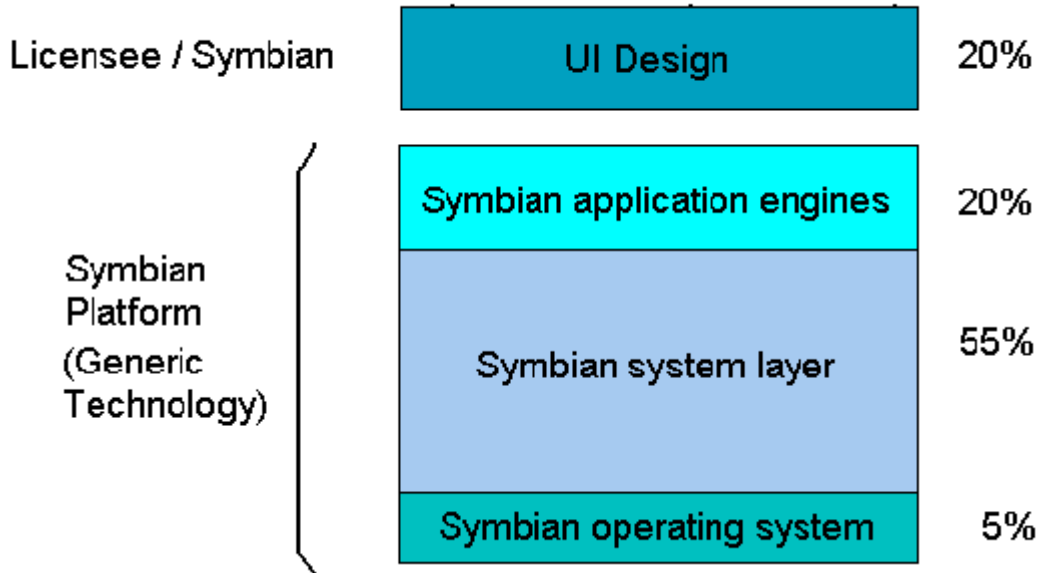


Figure 3: Symbian OS Generic Technology - Basic Layering

Figure 4 shows a more detailed component breakdown of these layers.

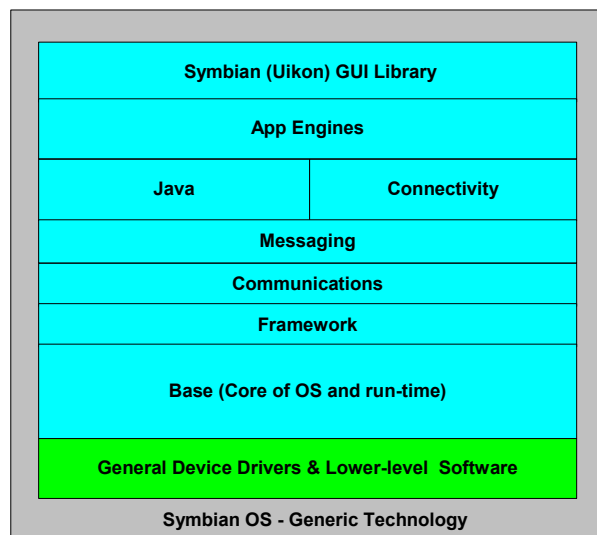


Figure 4: Symbian OS - Detailed Layering

The system kernel, file server, memory management, and device drivers are located in the “Base” operating-system layer.

The upper layers of the system provide communication and computing services, such as TCP/IP, IMAP4, SMS, and database management. Symbian OS components provide data management, communications, graphics, multimedia, security, application engines, messaging engine, Bluetooth, browser engines, and support for data synchronization and internationalization.

Symbian C++ APIs enable extremely efficient multitasking and memory management. Memory-intensive operations such as context switching are minimized. Symbian OS is primarily event-driven rather than multithreaded. Multithreading is possible but is avoided because it potentially creates several kilobytes of overhead per thread. Conversely, a primarily event-driven approach doesn't need any context switching and can have an overhead as low as a few tens of bytes. Special attention has been given to designing the Symbian OS to be robust and reliable.

Among the user requirements that were kept in mind when the Symbian OS was developed were that user data should never be lost and that the device should never have to be rebooted (there isn't a boot sequence when the device is turned on). This has been achieved by:

- Effective memory management that prevents memory leaks
- Releasing resources as soon as they are no longer needed
- Effective error-handling framework that handles out-of-memory errors properly
- Secure data storage
- Careful and device-specific power management

A smartphone application may be kept running for weeks, months, or even years, making memory management critical for the operating system. Symbian provides programming methods, such as heap checking, (debug) asserts, coding conventions and leave-trap mechanisms, which enforce good memory management.

The Symbian design is a client/server architecture, where applications are clients that use the resources of a wide variety of system servers. The client-server framework is widely acknowledged in the software community as a powerful mechanism. In the Symbian platform, clients are programs that have user interfaces, and servers are programs that can only be accessed via a well-defined interface from other programs. The role of a client is to serve the user, while servers ensure a timely response to all the clients while controlling access to the resources of the actual system.

The application engines enable software developers to create their own user interfaces to the application data and databases. Data synchronization is provided through a SyncML engine and external connectivity such as Bluetooth and a PC Connectivity suite.

In summary, the Symbian platform is a very effective way of improving the quality and performance of software on handheld devices.

6. Series 60 Developer Platform Technical Overview

The Series 60 Developer Platform runs on Symbian OS Versions 6.1 or 7.0s, which are common cores of Symbian APIs and operating system technologies. The Platform contains all of the interfaces to the UI applications, dynamic link libraries, executables and device drivers for controlling the keyboard, display, RTC, Bluetooth, IrDA, and Flash file devices. Symbian OS communicates with the device's core cellular software through a well-defined and documented messaging architecture.

Symbian OS v7.0s is an evolution of 6.1 and implements revised and new features while maintaining backward compatibility with its predecessor as far as possible. Among others, the new OS implements features such as:

- OMA Client Provisioning
- ECOM Plug-in architecture
- Version 6 IP addressing
- 3GPP R99/R4 support
 - Multiple PDP context
 - Quality of Service Framework
- Java MIDP 2.0
 - Including JSRs for wireless messaging and Bluetooth
- MultiMedia Framework (MMF – replacing the Media Server of OS v6.1)
- Language support
 - Arabic, Hebrew and Thai text rendering

The core of the Series 60 Developer Platform is Symbian OS GT (Generic Technology) layers (see Figure 3 and Figure 4). Series 60 adds the Avkon UI layer, a suite of applications based on the Avkon and Uikon libraries, plus the application engines. Porting to a new target platform will involve production of some low-level hardware-specific code such as device drivers.

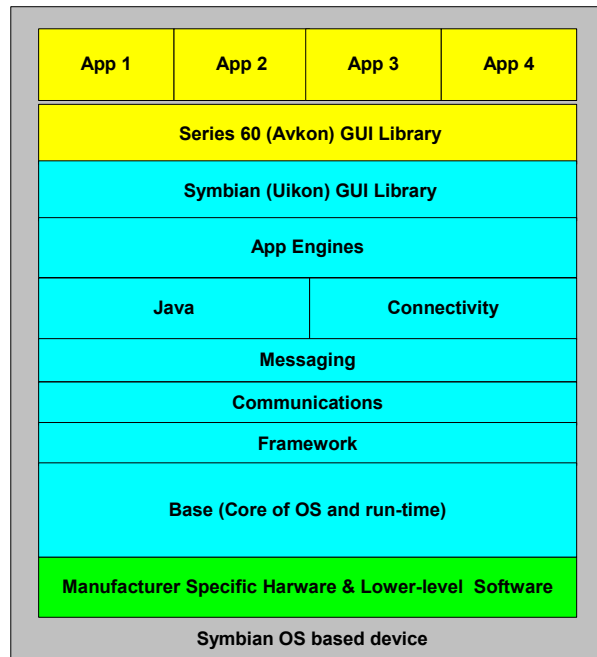


Figure 5: Series 60 layers

All the main logic and processing for core applications is contained in the Application Engines. Examples of Application Engines include:

- Contacts Database
- Multimedia Support
 - Media Server Engine (Platform 1.0)
 - Multimedia Framework architecture (Platform 2.0)
- WAP Loader
- WAP Stack
- Access Point (database containing connection settings)
- MLayDoc (base GUI objects used by layout engine)
- File System (access to stored data)

These engines use system services from lower layers. For example, the WAP Loader uses the WAP Stack to fetch data using the Wireless Application Protocol. The WAP Stack, in turn, uses the Socket engine in the Communication layer for network access, which in turn uses the ETel engine for hardware-specific telephony data access. In general, engines may be layered upon each other.

6.1 Avkon UI Library

The Avkon library defines many user interface components. Some key examples are:

- Status pane – framework and contents
- Main pane – Listbox, Form, Options menu, Grid, Query, Note, Soft notification and Setting page

- Control Pane – Soft keys and Scrolling indicator

Avkon builds on and extends the framework and controls provided in the generic Symbian Uikon library and a version of the Standard Eikon library modified for Series 60.

6.2 Applications

Each GUI application is then based on framework classes provided as part of the Avkon library and on the UI layers below that, for example, in Uikon. The majority of the Series 60 Platform user interface elements are based on standardized controls provided by the UI libraries or as specific custom controls provided by the developer. See Figure 4 and Figure 5 for a structural overview.

The Series 60 Platform GUI determines the rendering of all GUI elements, so the individual applications share a common look and feel (much like how Windows applications all share the same look and feel for menu and other UI components as implemented in Win32 libraries).

About 80% of the code for each application is contained in the application engines, which have no UI. The UI generally accounts for about 20% of an application, and can share services from all Symbian generic and Avkon elements.

7. Developing Applications

To enable creation of first-class applications and content, the terminal platform itself must provide rich APIs and other resources for use by producers. Additionally, the tools available to developers must allow efficient development and testing cycles to occur, preferably well before the availability of actual hardware devices.

Developers need tools that speed up and simplify the development process and thus time-to-market and hence time-to-revenue. Good-quality Series 60 Software Development Kits (SDKs) are key enablers for application and service developers who want to create compatible, high-quality applications and content for a variety of devices based on the Series 60 Platform. They need to feel confident that the applications they create will run on Series 60 terminals as they become available from licensees.

Application developers who want to reach the widest possible market should adhere to standards and use popular programming languages, such as Java and C++. The crucial standards and development tools, amongst many other protocols and utilities, have been implemented in the Series 60 Developer Platform, providing smartphones with an open-architecture-based operating environment. In mobile devices, the screen and keyboard hardware dictate how the user interface portion of the software is designed. In order to promote interoperability from the start, the Series 60 Developer Platform defines a common user interface for smartphones that are one-hand operated. This creates a uniform user interface and screen format for developers enabling simplification and savings on development cost.

Application designers can use a Series 60 SDK to create both stand-alone and server-connected applications for smartphones. Additionally, separate Java and WAP / XHTML toolkits can be used to design applications for Series 60. Applications developed once for Series 60 will then run on other Series 60 smartphones.

The Series 60 Developer Platform relies on standard protocols that have been defined for communication, messaging, browsing, and data synchronization. For example, a Java application in a mobile phone that extracts data from a corporate customer database may use HTTP for communication. A game can be downloaded into a phone, transferred over the WAP protocol through a mobile network. Synchronization of a contact list may take place at the user's PC using SyncML over Bluetooth. In addition to stand-alone applications designed for Series 60 devices, server applications, server utilities and middleware software represent a market for software companies.

Server applications can be hosted by a mobile network operator, provided as a service to subscribers. Mobile portals, payment solutions, location-based solutions, advertising and virtual communities are examples of these services.

For enterprises, GPRS means IP connectivity to mobile devices through mobile networks. Server software for mobile applications can reside in corporate networks, allowing Intranet applications to reach employees out of the office and facilitating mobile commerce applications to reach customers.

Data transfer between applications in different devices is enabled by standards that define syntax and format for data. The Series 60 Platform features a SyncML engine, which can be accessed through a public API. PC Connectivity over wireless or cable connection can be used for data synchronization and application installation:

- PC Connectivity Cable, Bluetooth, Infrared, SyncML
- Mobile network GSM, GPRS, MMS, SMS, WAP

- Enterprise POP3, IMAP4, HTTP, TCP/IP

In addition to designing for mobile devices, developers can find opportunities in designing software products for servers. For instance, mobile network operators and service providers provide content and application services for their subscribers. Enterprises need software that connects their mobile employees to the Intranet and applications that can be used for customer service and mobile commerce.

7.1 MIDP Java Development

For application developers, Java provides hardware independence, the support of a large development community, and built-in security. Java frees developers from dealing with the specifics of different devices, allowing them to concentrate on their applications.

Series 60 SDK for Java MIDP includes:

- Series 60 Emulator
- Integration with leading IDEs, for example, Borland JBuilder, JBuilder MobileSet, Sun Forte for Java
- Automated code generation wizards

Developer Platform 1.0 supports the MIDP 1.0 standard whereas Platform 2.0 supports all mandatory elements of the MIDP 2.0 specification with extensions to ensure backward compatibility with existing supported Series 60 audio formats and also features such as vibrate (where the device hardware supports it) and flash backlight.

For application developers, the J2ME platform provides a unique opportunity to write applications for devices that are very personal, yet are used by a vast number of people. Since the Java programming language is familiar to many application developers, it is easy for them to start developing applications for wireless devices. Moreover, the J2ME platform hides the complexity of the device from the applications themselves as well as from the application developers. By specifying a standard set of APIs and a standard application execution model, the J2ME platform gives application developers the opportunity to concentrate on deploying attractive mass market applications, while freeing them from dealing with the specific characteristics of different devices.

There are numerous benefits for operators as well. Operators can attract and retain subscribers by providing easy convenient access to applications, while at the same time ensuring that subscribers get top-quality applications and services.

In a smartphone environment, Java is best used for applications where time-to-market is critical, the client application acts as an extension to a server application, application requirements update frequently, and deployment of the application to different platforms is important.

Series 60 smartphones can be connected to the Internet, for instance using packet-based GPRS mobile networks. Network and PC connectivity allow users to download the applications and data of their choice into their smartphone. Since all users may not be security aware, or verifying the origins of the downloadable software may be difficult for users, Java has a built-in sandbox security model to protect the system from malicious software. This security is achieved by setting clear boundaries about what applications can do in the device.

Since Java has been designed to run on any device with a Java virtual machine, there are some compromises in its generic functionality. A Java application can't

access all features of a device, such as the contact database or the calendar – the full functionality is available in the C++ development environment.

7.2 C++ Development

The Series 60 Developer Platform ships with its own C++ SDK based on the Symbian SDK. The APIs that are documented enable third parties to develop Series 60 applications for inclusion in new Series 60 terminals or to be distributed as value-added, after-market applications.

Symbian OS is written largely in C++; the language therefore represents a strong development choice for third parties. Most popular development IDEs are supported by appropriate Series 60 SDKs, (available free from www.forum.nokia.com if not bundled with the product itself) these include Microsoft Visual C++/Studio and .Net, Borland C++Builder and Metrowerks CodeWarrior products. The Series 60 SDK provides documentation, tools, and sample code to assist developers, along with a Microsoft Windows hosted emulator. The SDK is essential for developing, testing, and debugging C++ applications.

Although C++ development is more complex than MIDP Java development, the advantage is that all of the device's capabilities are directly accessible by the application. Applications running natively provide superior performance and can take full advantage of a multitude of functions provided by Symbian OS, including access to functionalities such as Bluetooth, Infrared, networking functions, device native UI libraries, all messaging interfaces, graphics libraries, multimedia, and telephony.

The Series 60 Developer Platform includes ready-to-run applications that hardware manufacturers can embed in their Series 60 devices. These applications illustrate the possibilities of the platform, but they also guide developers in designing software that complies with the user interface style. Some of the reference applications provide public APIs for accessing their services from other applications. For example, the Phonebook application has a service for displaying a list of contacts (view switching), the Photo Album application has a service for finding images, and the Messages application has a service for sending e-mails.

The Series 60 Developer Platform fully supports installing and running applications designed natively for Symbian OS. The supported install file format is the Symbian Installation System (.sis) format. Applications can be downloaded via the browser, e-mail, file transfer by IrDA or Bluetooth, or added via a PC connection through an IrDA or a Bluetooth connection.

7.3 Comparison of Java and C++ Development

7.3.1 Symbian OS C++

Suitable when high performance and comprehensive functionality is required. A very extensive set of APIs provides access to all smartphone features, for example,

- Bluetooth and infrared
- Networking and communications
- Native UI classes
- All messaging interfaces
- Telephony

Typical application sizes are 50–500 KB with installation via a .sis installation file.

7.3.2 Java MIDP

Suitable when portability between different terminals and device categories is essential, that is, Volume category Java phones, smartphones, PDAs, and Communicators. The limitation is that not all phone features will be accessible/available. Available Java MIDP APIs include:

- Standard Java libraries
- Persistent data storage methods
- MIDP UI classes
- Basic networking

Typical application sizes are 20-40 KB, installed using a clearly defined JAD-JAR downloading method.

7.4 WAP Content Development

It is well understood that browsing on a mobile device is very different from browsing on a personal computer. A mobile device must show essential, personally relevant information, in a compact and visually appealing manner. The Wireless Application Protocol (WAP) is a global standard for mobile Internet applications and browsing. Functionally similar to the World Wide Web, it is designed to accommodate the limited memory and small screens of mobile devices, which are connected to servers over low bandwidth connections.

Packet-based GPRS networks make accessing WAP pages and applications more convenient over Circuit Switched Data (CSD) connections by shortening connection times and increasing the data transfer speed.

Examples of applications that involve real-time interaction or long-lasting sessions are: interactive games, on-line auctions, chat, and newsgroups.

A banking transaction on a smartphone is an example of an application that requires secure online connection between the client and the server, all the way through the transaction. A secure session over a browser connection is a proven way to implement such an application.

The Series 60 Developer Platform supports WAP 1.2.1 WML browsing. Features include color browsing, push functionality, OTA configuration of WAP settings, content download, WTAI, and WTLS security.

A number of tools are available for application designers and content publishers to create both XHTML and WML content. The Nokia Mobile Internet Toolkit is one option available for download at www.forum.nokia.com. Mobile Internet Toolkit, which runs on a PC, provides a complete scripting, testing, and simulation environment for developers. The included emulator is adjustable for different screen sizes, including the 176 x 208 pixel format normally used in Series 60 devices. Developers can create content and applications without an XHTML device, or without access to a mobile network operator infrastructure. Mobile Internet Toolkit 3.1 offers a PC-based development, testing and simulation environment for WAP and XHTML/CSS.

7.4.1 Interactive data applications

7.4.1.1 XHTML and WAP browsing

There is universal agreement that it is essential to drive new consumer-oriented functionality into early WAP services. The advent of GPRS coupled with WAP 1.2.1 features such as Push, WTAI, and enhanced security will certainly move the industry in the right direction. Early or prereleases of the Series 60 Developer Platform will contain a WAP browser that enables this functionality.

Later releases of the Series 60 Developer Platform will move customers to XHTML Mobile Profile (XHTML MP) and WAP CSS and provide important WAP extensions (such as WTAI links to make phone calls and WMLScript) to enable navigation between WML 1.x and XHTML documents and to ensure a richer XHTML experience. Both the XHTML Mobile Profile and WML 1.x functionality are native within the browser.

Content downloading is possible through standard Internet mechanisms and will not require proprietary technology such as a specialized “download ” server.

The Series 60 Mobile Browser has been designed to serve not only as the basis of the browser, but also to be used by other applications such as the multimedia-messaging client. To that end, Series 60 Developer Platform 1.0 ships with a fully tested WAP protocol stack.

7.4.1.2 Markup and scripting requirements

Again, the Series 60 Developer Platform will support two markups: WML 1.x and XHTML Mobile Profile. WML must be supported for backward compatibility; new development should be based on XHTML, and WML should not be developed further. Cascading Style Sheets (CSSs) should accompany XHTML to provide a superior, standard mechanism for content format and rendering.

WMLScript is the initial scripting language in the Series 60 Developer Platform. It should be maintained for backwards compatibility in WML 1.x applications but further development will be restricted to adding new libraries.

7.4.1.3 Advantages of XHTML and CSS for handset manufacturers

XHTML Mobile Profile and WAP CSS are important technologies for those who license the Series 60 Developer Platform. CSS in particular gives far greater control over the presentation of content delivered from a portal to a mobile phone.

With style sheets, a manufacturer can create a consistent look and feel across all supported devices. It is possible to control the layout of text as well as the font type, font size, margins, borders, bullet types, tables, icons, and even color. With this new degree of control it will be possible for manufacturers or their customers, the operators, to truly give their services a specific identity with their own “corporate stamp”. What’s more, these style sheets only need to be created once for any supported device and then they are automatically applied to all XHTML content.

XHTML and CSS bring this same (GUI) experience to mobile phones. The services look better and are easier to use, which will lead to an overall increase in usage by customers interested in maximizing the inherent value

of services. As a result, operators can expect to gain additional revenue from their wireless portals.

Like all browsers that support style sheets, the Series 60 mobile browser has a default style sheet that specifies how all XHTML MP elements will be displayed, in the event that the developers do not specify their own styles. The developers can specify their own styles in various ways: in an external style sheet, in a style element in the document head, in a style element in the document body, or using inline style. The recommended way is to use an external style sheet.

7.4.2 Other graphical user interface elements

The WML 1.2.1 standard, based on the WAP Forum June 2000 conformance release, does not mandate a standard for GUI form elements. XHTML Mobile Profile, the basis for WAP 2.0 June 2001, specifies form elements such as:

- Text and password input
- Text area (for long text input)
- Pop-up menus (select/option)
- Radio buttons
- Check boxes
- Submit and Reset buttons
- Hidden form fields for sending “state” with form data (for example,, user session ID)

XHTML browsers support all of these elements, and like desktop Web browsers, can display them inline, allowing the user to retain the context when selecting an action. The XHTML Mobile Profile specification uses WAP CSS to enable fine control over the appearance of all of these elements. This model, supported on all desktop Web browsers today, provides superior results compared to WAP 1.x, and must be supported by all wireless XHTML browsers.

7.5 Messaging

Developers can make full use of the messaging facilities in the Series 60 Platform. Messaging offers numerous opportunities to create messaging-enabled client applications, and to create plug-in modules to support customized messaging types. The built-in messaging framework in Series 60 provides an API for sending and receiving SMS, MMS, and e-mail (IMAP4, POP3, SMTP).

A useful feature for programmers is the Send-as API that helps create outgoing messages in client applications. It provides a common method of composing messages, no matter what type and transport are used, and features a simple user interface module for sending the message on its way.

7.5.1 MMS messaging

Multimedia Messaging is an emerging service for mobile messaging. Its service model has been copied from the existing Short Message Service, which has been very popular. MMS services are a clear and logical extension of SMS services with a similar business model and a great opportunity for

additional revenues. The Multimedia Messaging Service allows for the delivery of text and multimedia content with integrated presentation information among mobile users and between those users and the Internet.

Nokia Series 60 MMS SDK beta 0.1 for Symbian OS includes MMS emulator as a plug-in component.

7.5.2 E-mail

The Series 60 Developer Platform contains a standard POP3/IMAP4/SMTP e-mail client provided as part of the Symbian operating system. The client enables users to read, write, edit, and manage e-mail messages. The user will be able to reply including text, forward, attach files, including Java MIDlets, add recipients, add a subject, create a contact card, and add a signature file.

7.6 Bluetooth

The Series 60 Developer Platform enables Bluetooth application development for phones. Applications can range from simple chat applications to exciting multiplayer games. Several simultaneous links to other devices are also possible, enabling point-to-multipoint applications.

Bluetooth also offers possibilities for the use of wireless accessory appliances. A developer can write an application for a smartphone that uses a wireless accessory via Bluetooth. Possible accessories include bar code readers, digital pens, health monitoring devices, and GPS receivers.

Bluetooth applications can be developed using Series 60 SDK for Symbian OS. The SDK features the APIs and documentation for Bluetooth. The programming language used for Bluetooth applications is C++.

Documented Bluetooth examples help developers concentrate on the applications. Examples include:

- Device Discovery
- Service Discovery/Advertising
- Serial Port (point-to-point connection using RFCOMM)
- OBEX (Object Exchange)

7.7 SyncML

The Series 60 Developer Platform has a SyncML client agent and a synchronization engine. SyncML APIs providing access to synchronization services are available to developers.

8. Platform Compatibility

The Symbian platform is designed to be extensible, and is shared between all Symbian devices. This means that software typically needs to be developed only once for it to work on any Symbian device with the same UI. For different UIs, some extra work is needed to port and optimize the application for the new platform. A typical, well-designed C++ application will consist of 80% non-UI code and 20% UI platform-specific code. It is the UI specific code that has to be modified. A well-designed C++ application will have all of the UI code separated out and independent of the code for the engine, communications, and networking.

The open platforms based on Symbian OS will enable the creation of a “Symbian economy”. No longer will mobile devices only interact with a limited range of other devices (usually mobile devices from the same manufacturer, or PCs via connectivity software). Users will be able to interact with any device using the Symbian platform.

The philosophy behind the Series 60 Developer Platform is to take this concept a significant step further to provide complete compatibility between services and applications (Symbian OS, Java, messaging, browsing etc.) running on terminals from different manufacturers that are all based on the Series 60 Platform. As the platform evolves, Nokia, Symbian, and all Series 60 licensees are committed to maintaining binary compatibility (BC) wherever possible. This would allow applications developed for earlier versions of the platform to run on devices produced using a later version. However, it is possible that, in the future, breaks in BC will occur, and this can impact on the portability of some applications.

Developers will benefit from having access to a much larger market, as porting work is minimized between hardware, that is, developers have the freedom to create applications for a single platform available on phones from multiple manufacturers in a larger unified application market. Consequently, it is to be expected that a great deal of high-quality software will be produced by different vendors and will be available for download to wireless devices at a low cost. Ultimately the reduction in cost will mean that competition will become more intense, which is obviously in the interest of consumers. One of the many benefits to the consumer is that there will be more software to choose from, since the reduced porting costs mean lower barriers to entry for software developers. Consumers will also be able to seamlessly interact with many more users than ever before.

9. Developer Resources

9.1 Forum Nokia

Forum Nokia is Nokia's support forum for developers who are using its technologies; it is found at <http://www.forum.nokia.com/>.

The forum contains a wealth of information on Symbian OS and the Series 60 Developer Platform, as well as free SDKs. Access to these resources is available free to all registered developers. Resources also include discussion forums devoted to Symbian OS development. In order to gain full access it is necessary to register but there is no charge for registration or for participation in discussion forums devoted to Symbian OS development. The forum also contains numerous technical papers devoted to Symbian OS development, including papers devoted to C++ and general configuration issues; these can be found in the Symbian section.

The Forum Nokia Knowledgebase provides additional information about Symbian OS such as:

- Answers to frequently asked questions
- Useful code snippets illustrating Symbian OS techniques
- Information for developers working with Java, messaging and browsing

9.2 Symbian Developer Network

The Symbian Developer Network is Symbian's developer service organization, providing developers using Symbian OS with support and information to help them produce well-written and stable applications for Symbian OS devices.

The Symbian Developer Network provides:

- Free public discussion forums, code resources, and tips about Symbian OS development at: www.symbian.com/developer/support.html
- Professional support forums for subscribing members, staffed by Symbian developer consultants at: www.symbian.com/developer/prof/index.html.
- A technical library that can be found at: www.symbian.com/developer/techlib/index.html
- A portal to other such resources and technical and commercial services on Symbian DevNet partner sites
- A newsletter on what's new on the Symbian DevNet Web site and across partner sites
- The Symbian OS Knowledgebase, which provides up-to-the-minute information about Symbian OS, such as:
 - Answers to frequently asked questions
 - Defect reports and the corresponding workarounds
 - Useful code snippets illustrating Symbian OS techniques

- Other development information that is not available in the Software Development Kit documentation

9.3 Books

Professional Symbian Programming, M. Tasker et al, Wrox Press. (February 2000)

Developing Series 60 Applications: A Guide for Symbian OS C++ Developers, Leigh Edwards et al. Pearson Education (Addison-Wesley). (April 2004)

Symbian OS Communications Programming, M. Jipping, Symbian Press, Wiley. (July 2002)

Wireless Java™ for Symbian Devices, J. Allin, Symbian Press, Wiley. (September 2001)

9.4 Other Sources

Java information and downloads java.sun.com

XHTML and other Internet protocols www.w3c.org

WAP information www.wapforum.org

MMS, SMS, and other telecommunication standards www.3gpp.org

Information about SyncML www.openmobilealliance.org/syncml

vCalendar and vCard www.imc.org

10. Terms and Abbreviations

Term or abbreviation	Meaning
Avkon	Series 60 extensions and modifications to Uikon and other parts of the Symbian OS application framework
BC	Binary Compatibility
Bluetooth	A specification for short-range wireless communications
CLDC	Connected Limited Device Configuration (Sun J2ME)
GPRS	General Packet Radio System. A radio technology for GSM networks. Transmission rates with theoretical maximum of 171.2 Kbps
GSM	Global System for Mobile communications. Digital mobile phone system
GUI	Graphical User Interface
HSCSD	High Speed Circuit Switched Data. Data communications technology for GSM with transmission rates between 14.4Kbps and 57.6 Kbps
HTTP	Hypertext Transfer Protocol. Data transfer protocol
IDE	Integrated Development Environment
IMAP4	Internet Message Access Protocol. E-mail protocol
IrDA	Infrared communication. A suite of protocols for exchanging data via infrared
J2ME	Java™ 2 Micro Edition for consumer devices
Java	Industry standard object-oriented language and run-time execution environment
MIDP	Mobile Information Device Profile. An extension API to the CLDC, which addresses the specific needs of the mobile phone market
MMS	Multimedia Messaging Service Protocol defined by 3GPP. Messaging for text, images, audio
OBEX	Object Exchange – transfer of objects such as files or data between two devices, for example, by Infrared or Bluetooth
OTA	Over-the-Air
POP3	Post Office Protocol. E-mail protocol
SDK	Software Development Kit
SMS	Short Message Service Text. Messaging service in GSM networks
SMTP	Simple Mail Transfer Protocol. E-mail protocol

Term or abbreviation	Meaning
SyncML	Industry standard for universal synchronization of remote data and personal information
TCP/IP	Transmission Control Protocol /The Internet protocol. Internet protocol
UI	User Interface
Uikon	A UI and control framework common to all Symbian OS devices
vCalendar	Data format for exchanging calendaring and scheduling information
vCard	Electronic business card data format
WAP	Wireless Application Protocol. Optimized protocol for viewing content over mobile networks
WML	Wireless Markup Language. XML-based markup language optimized for mobile devices with small screen
WTAI	Wireless Telephony Application Interface. WAP tools for creating telephony applications
WTLS	Wireless Transport Layer Security. The WTLS layer provides privacy, data integrity and authentication for WAP
XHTML	Extensible HTML. XML-based markup language