Series 60 SDK for Symbian OS

Getting Started

1. Glossary

The following terms and abbreviations are used within this document.

Term	Meaning
API	Application Programmers Interface. System function libraries and other facilities that enable developers to produce applications that are suitable for a particular platform or device family
AVKON	Series 60 extensions and modifications to UIKON and other parts of the Symbian OS application framework
CSS	Cascading Style Sheets
IDE	Integrated Development Environment
GUI	Graphical User Interface
SDK	Software Development Kit
UI	User Interface
UIKON	A UI and control framework common to all Symbian OS devices
XHTML	Extensible Hypertext Markup Language

2. Introduction

Thank You for installing Series 60 SDK for Symbian OS. This guide includes an overview of the SDK components, gives pointers to other documentation included in the Series 60 SDK for Symbian OS and describes the basic programming and build model on how to create your first ‘Hello World’ application for the Series 60 platform. This document is targeted at developers taking their first steps with the Series 60 environment.

3. Series 60 Platform

Series 60 Platform is a complete smartphone reference design, including a host of wireless applications. It represents a rich environment for developers to create exciting content and innovative applications utilizing technologies such as MMS, Java™, WAP, C++ and Bluetooth. The platform builds on the Symbian operating system (Symbian OS), complementing it with a configurable graphical user interface library and a comprehensive suite of reference applications. A set of robust components and many varied APIs are provided for developers. The APIs supplied are used extensively by the suite of “standard” applications, but they have been designed to be re-used by third-party application developers as well.

3.1 Symbian OS 6.1

Central to the success of Series 60 Platform is Symbian OS; it is the foundation of the product.

Symbian OS is a 32-bit multitasking operating system, wherein 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 on. By complying with the platform architecture and software design guidelines, application designers can routinely manage such occurrences in the daily lives of smartphone users.

4. Series 60 SDK for Symbian OS

A wide range of tools, APIs, libraries and documentation enable third parties to develop new Series 60 applications written in C++ for inclusion in licensee products or as value-added, after-market applications.

Development is PC hosted, based on an emulator that provides a Microsoft Windows-based implementation of the actual phone. Technical requirements are:

- Windows NT 4.0 environment (other platforms such as Windows 2000 may be used but are not fully supported)
- 500 MB of free disk space or more
- Microsoft Visual C++, version 6.0

4.1 Contents of the Series 60 SDK for Symbian OS

- Build tools, instructions and environment.
- Debug and Release emulator environment.
- Target compiling environments for ARMI and THUMB
- Integrated Documentation. Combined Symbian & Series 60 documentation
- Java2 Runtime Environment v.1.3.1 by Sun Microsystems Inc.
- Active Perl Build 518 by Active State
- Series 60 Application Wizard
- Code examples.
- Code example documentation is embedded into main documentation set.

Figure 1 depicts the directory structure of the Series 60 SDK for Symbian OS. Series 60 SDK for Symbian OS is installed under Symbian Directory by default.

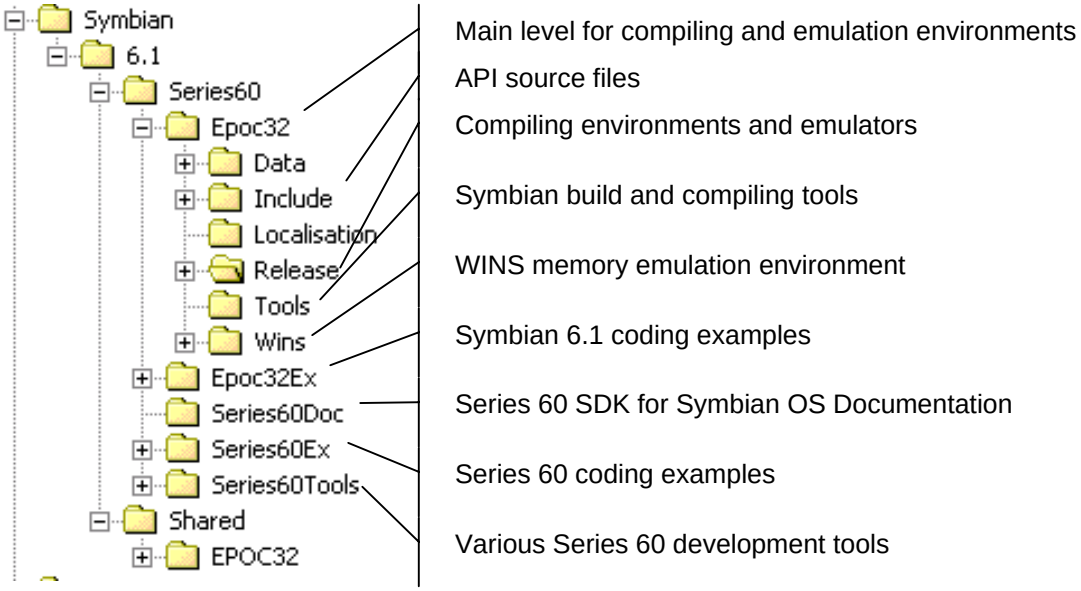

Figure 1. Directory structure of the SDK

After installation of the SDK you can find the basic tools from Start menu.

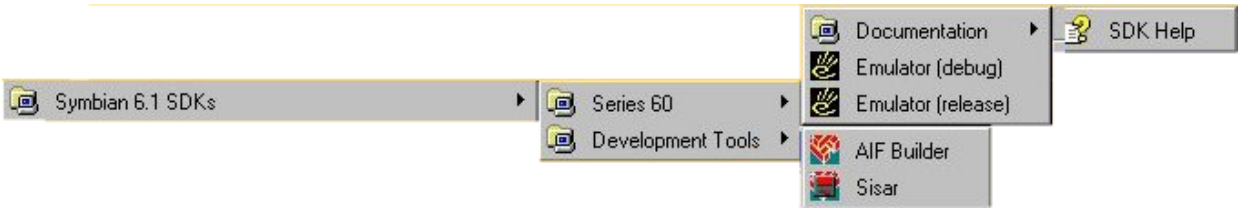

Figure 2. Start menu for the Series 60 SDK for Symbian OS.

4.1.1 Development Tools

At Start menu under Symbian 6.1 SDKs – Development Tools basic Symbian tools can be found. With AIF Builder You can develop application information files (.aif), with Sisar the Symbian installation packages. With these Java based tools the development of the applications has been made easier, faster and, no less important, visual. Read SDK help documentation for further information about these tools. These tools and also all Symbian tools including command line tools can be found from %Symbian SDK Shared%\EPOC32\Tools (if defaults used \Symbian\6.1\Shared\EPOC32\Tools).

A set of small and handy Series 60 development tools can be found from %EPOCROOT%\Series60Tools (\Symbian\6.1\Series60\Series60Tools). Read the installation instructions and read-me files from each tools directory before taking into use.

Series 60 Application wizard can be used for creating Microsoft Visual C++ 6.0 projects. All necessary files are created automatically and code skeleton is created. EpocSwitch can be used for switching from SDK to another if there is more than one Symbian development environment at your PC. EpocToolbar creates a tool bar to visual studio. With EpocToolbar you can create a new EPOC class, Update the visual studio workspace from the MMP, Build for target, Track Memory and Generate a pkg file for building a Symbian installation file.

5. Building 'Hello World' –application

5.1 Coding C++ for Series 60 platform

Instructions and guides on how to write C++ code for Series 60 and Symbian OS can be found from multiple sources. See chapter 6 for a list of Series 60 developer resources.

For most developers the emulator-based process accurately mimics the operation of a final application on a real phone so closely that the majority of their development work can be done before hardware is available.

For Symbian OS 6.1 C++ development, the edit/compile/build cycle is based on the Microsoft Visual C++ 6.0 Windows development toolset. However, instead of linking and building against Win32 or MFC, developers link and build against the headers and libraries installed by the SDK. The resulting binary is then run against the emulator implementation of the appropriate version of the OS.

The tools supplied with the SDK make this a simple process. Symbian's platform-neutral project (`.mmp`) file can be used to generate tool-chain specific project files for the VC++ IDE and the GCC tool-chain used for target device builds. During development, the Visual C++ project file manages all linking and building details. It also ensures that all outputs from the build, and any required resources including application resource files, icons, bitmaps, etc., are built into the appropriate location for debugging on the PC-hosted emulator.

For developers who prefer working at the command line, an essentially identical process is used to specify project information from which a makefile is created, which in turn manages the compile and linking steps.

5.2 Series 60 Emulator

The emulator provides a full target environment, which runs under Windows on a PC. This implementation of the Symbian platform is known as WINS — WINdows Single process platform. Applications run as separate threads within a single process under the PC emulator whereas on target hardware applications each runs in its own process, typically protected from one another by memory management hardware.

The emulator enables development to be substantially PC based. Only the final development stages focus on the target hardware. Typical exceptions to this rule are for low-level programming, where target hardware must be directly accessed — such as programming for a physical device driver and where the connectivity/communications requirements are not available via an emulator alone.

Figure 1: Series 60 emulator for C++

Two versions of the emulator are provided – a release build emulator and a debug emulator. The debug version of the emulator is used by developers the majority of the time. It is located in:.`\Epoc32\Release\wins\UDEB\epoc.exe`

5.3 Development Process Overview

A brief overview of the development process is provided here to give developers a sense of the essential steps.

- First, two platform-neutral project text files must be created, i.e., `bld.inf` and `project_name.mmp`, where `project_name` is the actual name of the project.
- The `bld.inf` file defines the component(s) to be built and each is specified in a project (`.mmp`) file. Typically, there are only two files and the build tools can use these files to create the platform-specific project files.
- Next, the initial source and header files required for the project are produced.
- Then the SDK build tools are used to generate a build command file called ABLD.BAT. From the command prompt, this ABLD command file can be used to build the project for a particular platform or to generate a range of platform specific project makefiles.
- Since most development projects are built and run from within the Microsoft Visual C++ 6.0 IDE, it is usual to use `ABLD` to create a `project_name.dsp` and `project_name.dsw` `makefile pair.`
- Opening the workspace file (for example`, project_name.dsw`) in Visual C++ allows the application to be further developed, built and debugged from within the IDE.
- To run a GUI application from within the IDE the executable specified should be `epoc.exe`: this will launch the debug emulator, the default for development projects.
- From within the Series 60 emulator the specific application being developed can then be found, under a folder called Other, where it can be selected and launched.

5.4 Application Wizard

An alternative method of creating new projects is via the Series 60 application wizard provided as part of the SDK. It offers developers a simple and convenient way to generate a basic application framework from within the Visual C++ IDE. The AppWizard will generate:

- A project based on one of three alternative UI display designs, i.e., dialog based, view switching or a control-based view
- Skeleton code and declarations for the four basic classes (App, AppUI, Document and View) associated with most Symbian OS applications
- All the build and project files necessary to build the project
- Simple resource files, and bitmaps that are used as default application icons
- The files needed to install the application onto a device

5.5 Tools Location and Installation

Tools for Symbian OS C++ development, with the exception of the emulators, are found in two directories:

- Tools developed by Symbian are in `epoc32\tools\` on the drive and folder where the SDK is installed.
- The gcc compiler and its supporting programs are in `Shared\epoc32\gcc\bin\` on the drive and folder where the SDK is installed.

Both of the tools directories should be in the system path. The SDK installation process normally updates the path and registry settings.

5.6 Resource Compiler

Resource files (for example, `appname.rsc`) are used in Symbian OS to define the way a GUI application looks on the screen. Much of the information that defines the appearance, behaviour and functionality of an application is stored in a resource file external to the main body of the program. Everything from the status pane, the menus, and the hotkeys, through to individual dialog boxes can be defined in the resource file. Individual resources are loaded as required at runtime and hence the memory requirements are minimized.

Application GUI resources are defined in text script files (typically with `.rss extensions`). They are compiled and compressed at build time into binary files that are used at runtime (by default with `.rsc` extensions). Resource files can be localized without recompiling the main program. For ease of localization, all user interface text is usually separated out into a separate header file (by convention with a .loc extension) that is #included into the main resource file. It is the `.loc` files that are later sent for translation into different languages.

Application caption resources may be used to provide captions (the application's name) in two different lengths and in multiple languages. The captions are displayed with the application's icon, i.e., for display in the "app launcher" of the Series 60 UI. These files are also processed by the resource compiler during a full build. See the AIF Files section later in this document for more details.

The resource compiler (`rcomp.exe`) is invoked from within the Visual C++ IDE or from the command prompt. Typically this is done through a command file (`epocrc.bat`) that combines the actions of passing a resource file through the C++ preprocessor, and then compiling it with `rcomp.exe`. Compilation generates a binary output file plus an `.rsg` header file that is `#included` into C++ code to identify specific user interface resources such as text strings, dialogs, menu components, etc.

5.7 AIF Files

Application information (`.aif`) files are used at runtime and store data concerning an application including:

- Icons in various sizes used by the system to represent your application
- Captions in the supported languages
- Capabilities, such as document embedding, new-file creation, whether the application is hidden or not, and MIME-type support priorities

Each application should own an application information file (or `.aif` file), which is used to contain a bitmap and captions associated with that application. Without an `.aif` file an application will use a default icon, and the application name as a caption (without the file name extension). It can be created independently of the application by the AIF Builder tool provided with the Series 60 SDK. The AIF Builder tool saves the details of `.aif` definitions in files with an `.aifb` extension.

The composition of `.aif` files can also be specified manually in resource script files with a `.rss` filename extension (for example, `HelloWorldaif.rss`) and compiled with the Aiftool utility. The Aiftool utility and the syntax of `.aif` script files are documented in the Series 60 SDK.

5.7.1 Icons

Icons are used to represent applications, and their associated files/documents, when they are embedded, or when they are shown on the application shell. Icons can be provided in a range of sizes; the most appropriate size for the current container zoom state will be displayed but icons of specific sizes are required e.g. the application grid launcher and application lists use icons which are 42 pixels wide by 29 pixels high whereas the application context pane of the status pane (shown at the top of the screen when an application has focus) uses icons that are 44 by 44 pixels in size. Supplying a variety of sizes helps to ensure that an icon will not have to be dynamically scaled when it is drawn at a particular size — scaling small bitmaps generally results in a marked loss of quality.

AIF Builder can launch an icon designer facility that will generate the bitmaps and masks that make up the icon. AIF Icon Designer helps you produce such icons in the required Symbian OS-specific bitmap file format, called the multi-bitmap file format (`.mbm`).

5.7.2 Captions

Avkon provides a possibility to associate a short caption with each application. By default, this will be identical to the caption found in the `.aif` file. However, it is possible for application authors to generate a separate caption file, containing a long caption and short caption. Two sizes may be provided to accommodate zoom levels of the UI. The short caption is used in application grid view, whereas the long caption is used in application list view. The caption file is generated in the same way as a normal GUI resource file. The resource structure used to create this caption file is defined in `apcaptionfile.rh.`

5.7.3 MIME Support

Multipurpose Internet Mail Extensions (MIMEs), define a file format for transferring non-textual data, such as graphics, audio and fax, over the Internet. A Symbian OS application may specify any MIME types that it supports in the `.aif` file, and the priority of support that each type is given.

For more information on `.aif` files, see Series 60 Tools, Designing Series 60 C++ Applications, and the Series 60 SDK documentation.

5.7.4 AIF File Localization

If an application requires different bitmaps for different languages, this is achieved by producing multiple copies of the `.aif` file, each containing the correct bitmap, using the aiftool. Each aif file produced must be localized by saving with the extension `aXX` where `XX` is the two-digit language

code associated with the appropriate language. The application framework (`Apparc`) software has been modified to attempt to load the `aif` file associated with the current language, selected by the user.

It is possible for application authors to generate a separate caption file for each language. Caption resource files should be named:

`<NAME OF APP>_caption.rss`, where `<NAME OF APP>` is the name of the application. The caption resource can be built as part of the normal build process by adding the additional line to an application's `mmp` file:

RESOURCE <NAME OF APP>_caption.rss

The compiled resource file will then appear in

`\system\apps\%appname%\appname_caption.rXX` where `XX` is the 2-digit language code specified in the `.mmp` file.

5.8 Context-Sensitive Help

A Series 60 device may have a Help application containing information about the system, the functions it offers, and advice on getting the most out of it. The topics in Help are arranged by application and in a hierarchical fashion so that they can be navigated in the usual way. The Help application is a normal application in regard to the ways it interacts with other applications.

Context-specific help can be obtained by selecting a Help item in the Options menu within applications. The Help menu item opens the corresponding topic in the Help application. The user can further navigate the Help application to find more information. In order to return to the used application, the application "Shell" or the fast application swapping shortcut can be used.

Context Sensitive Help (CS Help) provides users with the specific help topic for the context in which they have a problem.

The CS Help compiler GUI is a front-end user interface to the CS Help compiler (`cshlpcmp`) command line driven tool, which generates CS Help files. The GUI tool facilitates the creation and management of source text and graphics files, which form the content of Help files for any application that uses Symbian OS. It is recommended that the GUI be used for compiling Help.

The CS Help compiler is invoked using a project file, which defines the locations of all other files used by the project. These files include source `rtf` files, customization files, graphics files etc.

5.8.1 Customization File

The CS Help compiler (`cshlpcmp`) supports various customization options that can be used to change the appearance of the final help topics. These options are defined in a customization file, which is specified in the project file. If no file is defined, a default set of customization options is used.

5.8.2 Help Topic Source Files

Source (`.rtf`) files contain all the help information for a single application. Style marked paragraphs can be used to identify the following information:

Category name — the name of the category, or application, as it appears in the contents view of CS Help.

Category UID — the UID of the help file to be generated. This is unique for the help file, and allows Symbian OS to merge help files with the same name.

Topics — the individual help subjects, marked by topic titles that make up the help file.

In addition to the title and information, topics also have a number of fields to provide context sensitivity and searching. These include:

Context — a short unique context phrase that is used to create the link between an application context and the specific help topic.

Synonyms — alternative search words for the topic. Typically words with similar meanings or common misspellings.

Indexes — a string associated with a topic. An index phrase may relate to more than one topic, and hence need not be unique within a help project.

5.9 Symbian Installation System (SIS)

The Symbian Installation System provides a simple and consistent user interface for installing applications, data, or configuration information to Symbian OS devices. Developers and end users install components packaged in installation (`.sis`) files either from a PC, using the installer, or from a Series 60 device, using the Application Controller utility.

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 down-loadable software (i.e., 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.

An installation file can incorporate other installation files. This feature might be used for packaging files into logical components that are installed and removed as a complete set. For example, a shared library might be put in a separate installation file from the application files that use it.

A number of options may be offered to the user so that during installation the system will perform different actions depending on the user-selected option, for example, install a different set of files. A typical use of this facility might be to support the installation of different language resource files. Each resource file might contain the text used by the application (for example, in menu options) in a different language. The `.sis` file can bundle resource files belonging to a selected group of languages, but only the file supporting the user-selected language is installed.

Installation files can be configured to display text, or to run custom programs, during the installation and/or removal of an application. The most common use of this facility is to display a license agreement during installation.

Example installation source (`.pkg`) files are provided in the Series 60 SDK on all of the installation options described above.

After the installation process completes, a small "stub" `.sis` file remains on the device to control the un-installation of the application if required later.

5.9.1 Building SIS files

Two tools are provided to enable developers to build installation files:

The Installation File Generator (`makesis`) — creates the installation (`.sis`) files from a specification provided in a package (`.pkg`) file.

Note that within the package file, it is a requirement that a Series 60 Platform identification code be included. This enables a built-in mechanism to issue a warning if a user attempts to install non-Series 60 software onto a Series 60 device. For a fuller explanation of this requirement and the implementation details see:
`http://www.forum.nokia.com/040/1,,,00.html?fsrParam=3-3-%2Fmain.html&fileID=2461` – the relevant document is entitled "Series 60 Platform Identification Code".

The Certificate Generator (`makekeys`) — creates private/public key pairs, which may be used (optionally) by the Installation File Generator to digitally sign installation files.

The Sisar GUI tool provides a simple way to create `.sis` files: for full details see the Sisar Guide in the Series 60 SDK documentation. Both `makesis` and `makekeys` can be run from the command line or may be invoked from within Sisar. Sisar itself can also be invoked from the command line to build a `.sis` file. The `.pkg` text format files used by `makesis` can be imported into the Sisar

application. In addition, Sisar can also import `.aifb` files generated by the AIF Builder tool described earlier.

The Sisar tool is basically self-explanatory but has its own internal help mechanism and is separately documented in the Series 60 SDK. Note that like the other GUI-based tools on SDKs for Symbian OS, Sisar depends on a Java™ 2 Runtime, which can be installed from the same SDK installation package or CD.

Text source (`.pkg`) files specify the executable code files and other resources needed to install an application onto a device. It is possible to write the required `.pkg` files by hand, and build the `.sis` file using the tools described above.

5.10 Bitmap Converter – Bmconv

Bitmaps provide the pixel patterns used by pictures, icons and masks, sprites and brush styles for filling areas of the display. To optimize bitmap performance, Symbian OS uses files containing multiple bitmaps in its own highly compressed format. A tool is provided (`bmconv.exe`) with the Series 60 SDK that takes one or more Microsoft bitmap (`.bmp`) files as input and generates a single multi-bitmap file (`.mbm`), optimized for efficient runtime loading by Symbian OS. The Bmconv tool also generates a header file (`.mbg`) with symbolic definitions for each bitmap in the file for inclusion in C++ code to allow referencing of the individual bitmaps as required.

Two types of `.mbm` files are possible, ROM image bitmap files that do not use any RAM when being accessed and file store bitmap files that are loaded into RAM. During the conversion process it is possible to specify the number of bits per pixel for the converted bitmaps and whether they should be colour or grey-scale. The program can also split Symbian OS multi-bitmap (`.mbm`) files back into component bitmap files in Microsoft Windows bitmap (`.bmp`) format.

Developers do not always have to use the Bmconv tool directly to produce bitmap files that are dedicated to a particular application. Building the required bitmaps can be done as part of the standard `ABLD` project building process. A list of .bmp files can be specified in the project (`.mmp`) file. See the SDK documentation for details of the `START BITMAP` syntax statement in project definition (`.mmp`) file syntax.

5.10.1 Palette Support

For Series 60 SDKs the Bmconv utility has been changed so that the caller can specify a target palette for the conversion. Bitmaps with a target colour mode other than colour256 are not affected. Bitmaps with a target colour mode of colour256 are converted using the Series 60 palette, i.e., the 216 colours in the colour cube, and 10 grey shades. The palette file used for this can be found at: `..\epoc32\include\ThirdPartyBitmap.pal`.

6. Series 60 Developer Resources

6.1 Forum Nokia

Forum Nokia is Nokia's support forum for developers using its technologies and is found at http://www.forum.nokia.com/.

The forum contains a wealth of information on Symbian OS for Series 60 Platform, as well as SDKs free of charge. 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. However, 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 under 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

6.2 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 at:

www.symbian.com/developer/support.html

Professional support forums for subscribing members at:

www.symbian.com/developer/prof/index.html

The professional C++ forums are staffed by Symbian developer consultants.

The Symbian Developer Network Web site includes a technical library that can be found at: www.symbian.com/developer/techlib/index.html

Symbian DevNet provides:

- Discussion forums, code resources, and tips about Symbian OS development
- 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 that provides up-to-the-minute information about Symbian OS such as:
  - Answers to frequently asked questions
  - Defect reports and the corresponding solutions
  - Useful code snippets illustrating Symbian OS techniques
  - Other development information that is not available in the Software Development Kit documentation

6.3 Books

Professional Symbian Programming, M. Tasker et al, Wrox Press.

Symbian OS C++ for Mobile Phones, R. Harrison, Wiley

Symbian OS Communications Programming, M. Jipping, Symbian Press, Wiley

Wireless Java™ for Symbian Devices, J. Allin, Symbian Press, Wiley