

# 1

## Introduction

Welcome to the updated edition of *Symbian OS Communications Programming*! In this book we'll introduce you to much of the major communications functionality in Symbian OS and demonstrate how to perform common tasks in each area.

For this new edition we've started from scratch to produce chapters that are relevant to you as developers. Each chapter gives background information on the technology where necessary, an overview of the functionality provided in Symbian OS, and descriptions or examples of how to use the Symbian OS APIs. In cases where APIs or implementation differ between Symbian OS-based devices this is noted, and when the user interface platforms work differently then we'll show you what those differences are, or at least point you in the direction of some documentation that does.

### 1.1 What is in this Book

In this book we focus on using and extending Symbian OS functionality using the native C++ APIs. Whilst it is also possible to use Java to develop applications for Symbian OS devices, we do not cover that in this book. We also focus on APIs available in standard UIQ3 and S60 3rd edition SDKs – thus engineers at Symbian's licensees and partners will want to look elsewhere for details on the internals of the Symbian OS subsystems that we describe. However, the material in this book is suited to all developers – at third parties, Symbian's licensees and Symbian's partners – who wish to use the functionality described.

This book should also prove useful to newcomers to Symbian OS in the device creation community, providing a high-level overview of the communications side of Symbian OS, and an idea of how it all fits together. However, this is not likely to be sufficient for creating a device, for that you will need to look elsewhere.

There are three main user interfaces supported on Symbian OS – MOAP, S60 and UIQ. At present, only S60 and UIQ allow developers to extend functionality natively in the aftermarket, so we concentrate on those platforms in this book.

S60 and UIQ have, in some places, differences in the way they choose to use and expose certain Symbian OS functionality. As a result, some details given in this book differ between the different UI platforms. Where this is the case we will highlight this, along with tips on how to use the functionality on each platform. In some cases functionality might have an alternate implementation on a given platform, in which case we will point you to the appropriate developer documentation. In other cases, it might be missing entirely, which might mean you need to reconsider your development plans. In cases where functionality is missing or not yet exposed, it is possible that it will be available in a later release of that UI platform – check with the appropriate developer website for more information in these cases.

The scope of this book is quite broad – not only will we discuss the core communications functionality in Symbian OS – Bluetooth<sup>®</sup>, IrDA, TCP/IP and telephony, but we also look at some of the main areas where those technologies are employed – the messaging framework and plug-ins, the HTTP stack, the OBject EXchange (OBEX) stack and the OMA Device Management system. Therefore whether you need access to communications functionality at a high or a low level, there should be something in this book for you.

## 1.2 What isn't in this Book

Symbian's licensees have a lot of flexibility when creating a device – as is necessary in a market where there is plenty of differentiation between products. As such, the supported feature set in any given device depends greatly on the market segment at which that device is aimed. You can expect to find that some features are not supported in given devices – either where they are not suitable, or cost-effective enough to be included.

Equally, some devices have leading-edge features that have been developed for differentiation – in these cases, the generic implementation developed when the feature becomes widely available may differ from the original one, which is normally highly tailored for the lead device. Throughout this book we describe the generic implementations – the ones you can rely upon to remain compatible beyond the initial implementation. Therefore it is best to use these implementations wherever possible to minimize, or eliminate, the amount of rework your application requires when deploying it to a new device.

However, you may wish to use some of those leading-edge features, perhaps because your application is targeted specifically at the device containing them. In that case, we advise you to go directly to the device manufacturer's developer website for information on using such features.

### 1.2.1 Technologies not Covered in this Book

There are some notable absences from this book in terms of technologies that Symbian OS supports. The reasons for their omission vary, but below we'll summarize the main technologies that are missing.

#### **USB**

Symbian OS has included support for acting as a USB device (or client) since Symbian OS v7.0, and this support was backported to v6.1 for certain devices. However, extending this support is difficult due to the nature of the USB controllers used in devices. Typically these have a limited number of USB endpoints. By the time most devices are shipped, their USB configuration is such that most, if not all, of those endpoints are in use. This means that third-party extensions would not be possible, as they would not have any resources available with which to implement their functionality. As a result, Symbian defines the APIs required to extend the USB implementation as partner-only.

The one case where USB functionality might be of use to developers is the USB virtual serial port (more accurately, the USB ACM class). However, current devices are rarely configured to leave an available virtual serial port – in most cases they are already in use, providing the interface to use the device as a modem, or for use with debug agents. Check the documentation for the devices you are targeting to see if there is a USB serial port available – if there is, then it can be accessed via the RComm interface.<sup>1</sup>

#### ***Real-time protocol and real-time control protocol (RTP and RTCP)***

The Symbian OS implementation of RTP has been supported since Symbian OS v8.1, and has been shipping in devices since v9.1. However, it is not until v9.2 that it is present in both S60 and UIQ devices – in v9.1 it is currently only present in devices using UIQ3.

Various problems led to us leaving RTP and RTCP out of this book. Firstly, the RTCP API is currently marked as partner-only. Secondly, the Symbian OS RTP implementation currently lacks a few key features. There

---

<sup>1</sup> The Symbian OS Library contains more information on use of the RComm interface and opening a USB virtual serial port.

is no support for authentication or encryption of RTP packets – in fact, there is currently no support for any profiles, including the Secure Real-time Transport Protocol (SRTP) – this is left to applications to implement. Even if an application wished to try and implement SRTP, low-level cryptography functionality in Symbian OS is currently exposed via partner-only APIs. As a result, it is very difficult to deploy RTP in situations where secure audio support is required and the underlying transport doesn't provide its own security. In practice, this would mean that you would need to use IPSec for RTP over UDP and Bluetooth link layer security for any Bluetooth links using RTP, which may not be compatible with any system you are trying to integrate with.

Hopefully all of these problems will be addressed, and future versions of this book can include information on RTP. In the meantime, for those of you that wish to experiment, documentation can be found in the UIQ version of the Symbian OS Library, under 'Mm Protocols RTP'.

### ***Session initiation protocol (SIP)***

Symbian OS v9.2 contains a SIP stack. A similar implementation has been available as part of the S60 UI platform for a number of releases now, and has appeared in some, but not all, S60 devices. However, as with RTP, we have chosen not to write about SIP until the situation regarding availability in devices and recommended APIs has been resolved. For more information on SIP in v9.2, consult the S60 3rd edition, Feature Pack 1 documentation.

## **1.3 Expected Level of Knowledge**

This book assumes you have had some experience of developing for Symbian OS, and that you are aware of the basic paradigms – active objects, the client–server framework, and descriptors and have an understanding of the basic types of Symbian OS classes (e.g., C-, R-, T-, M-classes) and how they behave.

If you are new to Symbian OS, or want to take a refresher course in these concepts, there are several books that you could read, including *Symbian OS C++ for Mobile Phones*, and *Symbian OS Explained*, which offers an excellent guide to the basics of programming for Symbian OS.

In addition, we assume you're familiar with the concepts of the platform security model used in Symbian OS since v9.0. We describe capabilities required to perform operations in this book, as well as the other impacts that platform security has, but we don't dedicate space to the underlying concepts. For more details, you can read the *Symbian OS Platform Security* book, although it goes into a lot more depth than is required to understand our discussions in this book. There is a sample

chapter from the book available on the Symbian website – this contains a good description of the three key concepts of platform security, and is well worth reading. There is also a good FAQ about platform security available from the Forum Nokia website, <http://www.forum.nokia.com>.

In terms of required communications knowledge, we assume you're familiar with the usage of SMS, MMS, email and HTTP, so have an understanding of the features they provide. More mobile-specific technologies, such as OBEX, Bluetooth, IrDA and Device Management, have more background information as we assume that these technologies may not be as familiar to most developers, especially those from a PC background.

We also assume that you are able to set up a general development environment for Symbian OS development – you have installed the appropriate UI platform software development kit (SDK), an integrated development environment (IDE), if you choose to use one, and are at the stage where you can build software for both emulator (again, if you choose to use it – we strongly advise that you do) and the target platform (whichever phone(s) you choose to test on). However, we don't require you to know about setting up the emulator for comms development – we have a chapter towards the end of the book explaining how to do this for various different technologies. Obviously testing on actual hardware is also extremely important, and in some cases the only way to test your application. In this case though we have little to say in terms of setting it up – the hardware features of your phone are fixed, and most of the configuration issues we discuss in relation to the emulator simply don't apply – the settings have already been configured for your particular device.

## 1.4 Structure of this Book

This book is divided into four main sections. The first section – chapters 1 and 2 – aim to give an overview of the book and its contents. It also provides an introduction to the communications functionality in Symbian OS, and a high-level overview of how it all fits together.

The second section – chapters 3 through 7 – covers lower level communications technologies. Roughly speaking, these are technologies that exist, in Symbian OS, below a socket interface – Bluetooth, TCP/IP, IrDA – or technologies that provide simple data links, such as virtual serial ports,<sup>2</sup> or other core services such as telephony. Each of these technologies acts as the underlying layer for the functionality covered in Section III.

---

<sup>2</sup> Real serial ports, of the RS-232 variety, have died out completely from phones, and these days only exist on development boards. As this book is targeted at developers using phones rather than development boards as their target hardware, we only discuss serial ports in the context of the virtual serial ports provided by Bluetooth and IrDA.

The third section – Chapters 8 through 12 – covers higher level technologies, such as the messaging framework and plug-ins (for SMS, MMS and email), the SendAs service, OBEX, hypertext transfer protocol (HTTP) and OMA Device Management (OMA DM). As these technologies build upon those discussed in Section II, some APIs from that section will be revisited here – for example, the HTTP APIs expose the `RConnection` API. So whilst each chapter tries to stand alone, it might be necessary to read an earlier chapter to understand all that is being described. See section 1.7 for a more comprehensive reading guide.

The fourth section – Chapter 13 – contains practical information on developing with Symbian OS. This includes information on setting up the development environment for various types of comms-related development. Much development can be done using the Symbian OS emulator – we discuss various ways of connecting the emulator to an IP network, and using an IR pod, Bluetooth hardware and a phone with the emulator. Finally, there is a brief chapter about future developments, covering new features or areas of technology that are likely to be interesting in the future. If enough of these changes take place, it will be time for us to start writing a new book!

The sample code provided in this book is designed to be dropped straight into your application – this means it doesn't use the shortcuts seen in other examples, where the code will work in the standalone example but needs rework before you can use it in your application. Where there is additional work needed, for example where errors need to be handled in an application-specific way or we have omitted some code for the sake of clarity, this will be shown clearly – this means you can spend more time working on the interesting part of your application, and less figuring out why our code doesn't work in your application!

## 1.5 To which Versions of Symbian OS does the Information in this Book Apply?

This book targets Symbian OS v9.1 and v9.2. Symbian OS v9.1 is the version used in S60 3rd edition and UIQ 3.0 phones. In version 9.0 of Symbian OS, a new security model was introduced which affected the usage of many APIs. This book describes the APIs after the security changes were made. Some APIs have changed very little, some have had methods replaced, and some have been removed completely.

In cases where the API that we are discussing is significantly different in an older version of Symbian OS, information will be presented in a box like this one. Reference will be made to the old API, allowing you to look up the information in the appropriate version of the Symbian OS Library.

Information referring to future versions of Symbian OS – including that which applies to v9.2 but not v9.1 – will be presented in a similar style.

## 1.6 Example Applications

Most chapters have example applications to demonstrate the functionality we describe. You can download the source code from the Symbian website: <http://developer.symbian.com/main/academy/press>. Whilst the example apps have all been developed on S60 3rd edition or UIQ3 emulators and devices, the user interfaces provided by some apps use a simple text shell. This is in order to focus on developing the communications side of the application rather than designing a nice UI.

## 1.7 Reading Guide

Whilst we've tried to make each chapter stand alone, inevitably some chapters have dependencies on others in this book. These are our suggestions for reading each chapter.

You are reading Chapter 1 now, so that's covered. We suggest that everyone takes a look at Chapter 2 to understand the basics of the system structure.

Chapter 3, covering ESOCK, is rather abstract – we suggest you read one of Chapters 4, 5 or 6 (Bluetooth, IrDA, TCP/IP) first, then go back and look at chapter 3 to understand the basic framework.

Chapter 7, Telephony, stands alone.

Chapters 8 and 9 fit together well, as they cover the messaging functionality from two different angles. The example set of MTMs in chapter 9 use the example code in chapter 11 for some functionality.

Chapter 10, OBEX, requires some understanding of the material covered in chapters 4 and 5 – this is clearly indicated in the text.

Chapter 11, HTTP, refers to the `RConnection` API, which is explained in chapter 6.

The example in Chapter 12, OMA Device Management, manipulates settings for the example app in Chapter 6, so it's worth having a look at the application to see which settings need to be managed remotely.

## 1.8 Other Sources of Information

Several times in this book we refer to the Symbian OS Library (formerly the Symbian OS Developer Library). This is available online at

<http://developer.symbian.com/main/oslibrary>, and is also integrated into the documentation that comes with S60 and UIQ SDKs.

Information on specific UI platforms, as well as additional example code for Symbian OS, is available from the appropriate developer website [www.forum.nokia.com](http://www.forum.nokia.com) or <http://developer.uiq.com>.

## 1.9 The History of Symbian OS Communications

To finish up this chapter we'll give you some background on the development of the communications functionality in Symbian OS, and take a quick glance at how it has evolved over time – starting from the initial implementation by Psion.

The initial history of Symbian OS communications is closely tied to that of the Psion Series 5 organizer – one of the original palmtop computers. In 1997, Psion PLC released the original Series 5 organizer – a revolutionary 32-bit, ARM-based PDA. This was a follow on from their 16-bit Series 3 organizer, itself a popular product in UK and some parts of Europe in the early 1990s. A major strength of the Series 5 was an in-built suite of PIM applications. The other was an increasingly strong suite of communications-related functionality added during the product's lifetime. The initial device contained an infrared port to beam information between devices over IrDA (but not using IrOBEX, which wasn't standardized in time for the initial Series 5) as well as a serial port for connecting to PCs and modems. To access this functionality two of the components still in use today were first introduced – c32 (more recently referred to as the serial server) and ESOCK, which provides the Symbian OS socket API.

Released after the original organizer as an installable software component, the Message Suite added a messaging subsystem with POP3, SMTP and fax capabilities to the device.

In 1999, the follow up to the original Series 5 was released – the Series 5mx. This increased the amount of RAM and ROM in the device, providing space for more functionality. OBEX was added to replace the original Psion-proprietary IR transfer protocol – providing standards-based object transfer to other (non-Psion) devices. Also included was a new messaging architecture – the same as the one as you see today – designed to allow extension of the messaging functionality of the device using aftermarket plug-ins called Message Type Modules (MTMs), the first of which was an IMAP4 implementation.

Later came a web browser from Opera Software AB. By this time, the Psion organizer was sporting as much communications functionality as some PCs, all in a much smaller format and running off two AA batteries.

In 1998, Symbian Ltd was formed – a joint venture between Ericsson, Motorola, Nokia and Psion. The staff of Psion Software Ltd transferred

to Symbian Ltd, and started working to produce the first Symbian-based mobile phone – the Ericsson R380. This was based off a Unicode version of the ER5 release that powered the Series 5mx, and marked the start of the ‘one-box’ era in Symbian – the first time that the communications solution did not require a separate phone or modem in addition to the Symbian OS-based device, which had traditionally been referred to as the ‘two-box’ model.

One notable feature lacking from the R380 was the ability to install software. In 2001, the first open Symbian OS-based device was released by Nokia to replace their 9100-series communicator – the Nokia 9210. Again, this was a one-box design, with integrated GSM functionality.

By this point, Symbian was working on adding both GPRS and Bluetooth functionality, which eventually shipped in Symbian OS v6.1. The first product to take advantage of this was the Nokia 7650 – most notable for being the first Symbian OS-based phone with an integrated camera. This feature meant it also included support for multimedia messaging (MMS) as part of the messaging functionality. As part of the introduction of Bluetooth, OBEX was extended to run over the new transport, and had OBEX authentication added. Symbian OS v6.1 also saw the release of the new IP-based connectivity solution for Symbian OS – replacing the old Psion Link Protocol (PLP) for connecting the device to PCs. As well as serving as the basis for a series of Nokia phones, this version of the OS also went into a series of sophisticated handsets released by Fujitsu for the DoCoMo network in Japan.

The first Symbian OS v7.0 device was the Sony-Ericsson P800, released in 2002. New functionality in this release of Symbian OS included an HTTP stack, as well as the first release of the hybrid IPv4/IPv6 stack. The ETel multimode APIs were introduced, as was the first version of SyncML.

Symbian OS v7.0s was first used in the Nokia 6600. Building on v7.0, the new release added advanced multihoming capabilities to the IP stack – allowing multiple simultaneous IP-based network connections with no risk of IP address range clashes or ambiguous routing situations. This functionality is still a step ahead of many operating systems today, many of which have problems when attached to multiple IP networks. This feature is especially important if there are any ambiguities between the networks, such as address ranges that are operational in both networks – as is often the case where private IPv4 addresses are in use. Symbian OS v7.0s also introduced support for OBEX over USB, and the ability for OBEX to receive objects to file rather than RAM.

Symbian OS v8.0 saw major enhancements to the Bluetooth APIs for link management, often referred to as the Bluetooth v2 architecture. OBEX gained APIs for user-defined headers and the ability to use double buffering when transferring objects.

Symbian OS v8.1 saw the introduction of Bluetooth PAN profile, supporting the GN and U roles – although it was never shipped in a

device until v9.1 when it appeared in UIQ3 devices. Other changes included the SSL/TLS library being rewritten, providing new functionality such as client authentication, and including new cipher suite support including AES-based cipher suites. This version went into a series of phones released by Fujitsu and Mitsubishi in the Japanese market, as well as the Nokia N70, N72 and N90.

Symbian OS v9.0 never shipped publicly – instead acting as an interim release to allow Symbian OS licensees to adopt the platform security model.

Symbian OS v9.1 has been an extremely popular version of Symbian OS – shipping in over 12 S60 and three UIQ device models so far. It saw the introduction of the multithreaded version of ESOCK, capable of running each protocol module in its own thread to allow increased independence in scheduling of protocols. Also introduced was support for the Bluetooth A2DP profile – whilst the profile itself is not implemented by Symbian OS, much of the core technology is contained within the Symbian OS Bluetooth implementation. A new API on ESOCK, `RSub-Connection`, was also introduced – initially to allow quality of service (QoS) functionality to be exposed.

Features added in Symbian OS v9.2 are covered in the rest of this book.

## 1.10 Summary

In this chapter we've summarized what information is contained in this book and explained how it is structured. We've also mentioned which topics and technologies are not in this book, and where to go to find information on them. You should have some idea of the level at which this book is pitched – we assume you have some basic Symbian OS development experience, and are in a position to be able to create applications, but don't expect you to know everything about communications technologies or their implementation in Symbian OS.

The next chapter introduces the major communications subsystems in Symbian OS, and shows how they fit together. If you're interested in a specific technology, and are happy about basic communications concepts and the structure of communications in Symbian OS, feel free to skip ahead to the appropriate chapter. Otherwise, read on for an introduction to Symbian OS communications functionality.