

# Platform Security and Symbian Signed: Foundation for a Secure Platform

**Ben Morris**

**Published by the Symbian Developer Network**

**Version: 1.4 – January 2008**

<b>1</b>	<b>INTRODUCTION.....</b>	<b>2</b>
<b>2</b>	<b>THE PROBLEM.....</b>	<b>2</b>
2.1	Filtering the hype.....	2
2.2	What's in it for developers?.....	3
2.3	The need to be seen to have a solution.....	3
2.4	Security by obscurity.....	4
<b>3</b>	<b>PLATFORM SECURITY AND SYMBIAN SIGNED.....</b>	<b>4</b>
3.1	The signing process.....	5
3.1.1	<i>Publisher IDs.....</i>	<i>6</i>
3.1.2	<i>What's in a signing?.....</i>	<i>7</i>
3.1.3	<i>Symbian Signed Test Criteria.....</i>	<i>7</i>
<b>4</b>	<b>MORE THOUGHTS ABOUT PLATFORM SECURITY.....</b>	<b>8</b>
4.1	Capability groups.....	8
<b>5</b>	<b>WRAPPING IT ALL UP.....</b>	<b>10</b>
<b>6</b>	<b>REFERENCES AND RESOURCES.....</b>	<b>10</b>
6.1	Reading.....	11
6.2	Web resources.....	11
<b>7</b>	<b>AUTHOR PROFILE.....</b>	<b>12</b>

## 1 Introduction

This is the first in a series of papers which examine the Symbian OS native C++ idioms that developers love to hate. While you may not end up loving them, the goal of the series is to show you why they exist, what practical issues arise when using them, and how to use them effectively.

In particular the series aims to show you what problems the Symbian idioms solve in the context of writing native C++ for mobile phones and other devices based on the Symbian OS platform, and how to use them to solve those problems in your projects.

So without further ado, let's get started.

## 2 The Problem

Like celebrity, success can bring unwanted attention. For a successful software platform, that means the now-familiar crop of viruses, worms, trojans and the like, targeting the ordinary users of the platform. This is attention that mobile phone users can do without.

Symbian OS v9 introduces a new, system-wide platform security architecture, together with a signing program, Symbian Signed, that forms its public face and serves as gatekeeper to the platform.

For most users, this is probably not controversial. For Symbian OS developers, however, as anyone who has followed the developer forums will know, it is still a hot topic in the context of Symbian OS v9.

This paper aims to help developers through the 'pain barrier', as well as clarify the benefits which lie beyond.

Platform security, after all, has a simple goal: to keep out the kinds of intentionally rogue software, or 'malware', which plague the PC world. Developers ignore that point at their peril.

### 2.1 Filtering the hype

It is reasonable to ask how great the threat from malware really is on smartphones. For example, phones will never be candidates for spam relays in the way that 'zombie' PCs are. Even viral payloads in email are unlikely to be a significant problem any time soon, because most users (possibly surprisingly) don't yet do phone email. More importantly, there are some inherent properties which make Symbian OS in particular less vulnerable than desktop systems; Symbian OS is a ROM-based operating system, for example, which means that its system files can't be compromised in the way that those of a PC operating system can; and by design, Symbian OS offers protection against some common cracking techniques like buffer overruns.

Phones are also different because the physical access model is different; typically, when it isn't in a pocket or a handbag, a phone is in its user's hand. The transient network access model is also quite different from the desktop model; without a fixed presence there is no fixed target. If phones are at risk, certainly the risks don't cover the full spectrum of desktop vulnerabilities.

Viewed from this perspective, it is easy to think of the threats as being overhyped by a security industry keen to sell its solutions to end-users. Bluetooth snarfing is one obvious example; despite recurrent scares, 'bluejacking' is a rare phenomenon, and any Bluetooth-borne worm has the serious disadvantage that the user must actively reject no less than three system warnings to accept the viral payload.

Having said that, as anyone who has read the literature (or simply seen the movie *Catch Me If You Can*) will know, end-users - us humans - are the weakest link. People are basically helpful, as

security guru Bruce Schneier puts it, and that's why so-called 'social engineering' works. In the colourful words of another security pundit, Ed Felten, 'Given a choice between dancing pigs and security, users will pick dancing pigs every time.'

Finally, it is precisely because mobile phones are different that they really *do* present new possibilities to the bad guys:

- Network damage; clearly this is high on the list of concerns for the network operators, although it's not clear what exactly an attack on a mobile network would look like, or what role handsets could play.
- Premium service frauds and so called 'billing storms'; because mobile network usage is chargeable, potentially there is cash to be made through scams which defraud users.
- LAN perimeter security breaches; mobile handsets which interact with a local network may evade the standard perimeter checks. Camera phones were bad enough for some corporates; network connectedness is even more of a threat.
- Personal data theft; arguably phones are an even richer source of personal data than desktop computers, with call logs and address books there for the mining; with 'presence' data and wallet applications on the way, these problems will only get worse.

## 2.2 What's in it for developers?

Inevitably, developer forum discussions have focused on the negatives. Platform security introduces significant system-wide changes, and signing imposes new requirements on developers. The upshot is more work, more complexity, more pain.

The good news is that since the publication of Craig Heath's book (Heath, 2006 – see the references section at the end for more details), developers have access to an authoritative introduction to platform security on Symbian OS, complete with How Tos covering all the major use cases - writing apps, writing servers, writing plug-ins, sharing data. It may not make the evolution to platform security trivial, but it makes it doable.

Even better news is that Symbian Signed has evolved significantly since its first roll out, and continues to evolve to try to make processes easier for developers.

But above all, Heath makes the *positive* arguments about signing and platform security persuasively. Market building is an essential dimension of platform security and Symbian Signed; an important part of their intent is to create willingness amongst users to download software, and that after all is what an open operating system is all about. As he says:

*'The mobile phone market will flourish when software for phones flourishes.'* (Heath, 2006, p 20)

## 2.3 The need to be seen to have a solution

Market building is not just about end-users, and it is not just about third-party developers. Platform security is an interesting example of Symbian leading its customers (see for example Charles Davies' remarks - quoted in my book - about the difficulty of getting platform security into an OS release). Work on a security solution began as long ago as in the planning for a follow-on to v7.0s.

At the same time, it is clear that there is a genuine operator need. It's not just that the network operators are afraid of exposing their infrastructure to poorly written applications or even deliberate attack. Just as important is their need to build end-user confidence in their networks, and in the reliability, integrity, and safety of the handsets the public buys. Nor do the networks want users returning their phones to the high street shop because a downloaded third-party program has failed, or worse, trampled the software setup. Sooner or later, security was going to come, and

better that it come by securing an open platform than by trying to close it down, or put up walled gardens for content and applications.

The Japanese market in fact is a good example of what happens when operator confidence cannot be built; in Japan, the network operators have been extremely reluctant to embrace open, i.e. third-party native programmable, mobile phone platforms, and Symbian OS in Japan has remained a closed system from the first v6 phones to the latest phones on DoCoMo's FOMA platform.

Here in the UK, in its early days the 3 network was clearly wary of open phones and the possible threat they posed to network stability, hence the first 3G Symbian phones running on 3, the Motorola A920 for example, were also closed.

Linux, at first sight surprisingly given the sloganising, has not been open on phones in *any* market to date. Apple's iPhone may eventually be opened: but the original explanation Steve Jobs gave to Newsweek for keeping the platform closed was a familiar one: 'You don't want your phone to be an open platform. Cingular doesn't want to see their West Coast network go down because some application messed up.'

What's the lesson? That there are much more complex sets of commercial forces at play in the mobile phone context than, say, in the desktop one. For developers, understanding these forces and how to work with them is essential to unlocking the value in the phone market.

Put another way, understanding the technology is in some respects the easy part.

## 2.4 Security by obscurity

Up until the introduction of platform security and Symbian Signed, the only security option available to Symbian OS was security by obscurity - keeping system internals 'hidden', one way or another, and discouraging over-zealous exploration by developers. That may have been good enough in the early days of Symbian OS, when the mass market was still an aspiration. But with upwards of 165 million (and counting) Symbian OS based phones in the market, and a run rate which is currently adding a further 60 million or so per year (and rising), robust security is a requirement, not an option.

## 3 Platform Security and Symbian Signed

Conceptually, Symbian OS platform security consists of three connected mechanisms:

1. A software-level security model implementation which protects sensitive APIs and resources;
2. Hardware memory protection which underpins the software model;
3. A signing program that enables third-party access to protected APIs, and gives users a basis for trusting third party applications.

Fully 60% of Symbian OS APIs are in fact open i.e. *not* platform security-protected. Of the 40% which are protected, many are intended solely for internal or partner use, meaning not just that they are not *intended* for third-party developer use, but they are not even available (typically, no headers or libraries are provided in SDKs, and therefore code cannot be written against them).

It is quite possible to write simple applications which use only open APIs. Realistically however, to take full advantage of the platform requires access to at least some protected APIs. And what *that* means is that developers need to understand both platform security and the signing process.

Choose your analogy. Symbian Signed is like the gatekeeper to platform security, and signing provides developers with the key they need to get through the gate. Or, platform security is like a

firewall, locking down access to the system; and Symbian Signed is the mechanism which allows developers to negotiate entry through the firewall.

This is how it works:

- Sensitive APIs are protected by capabilities, which in effect are access tokens. To access protected APIs, programs must hold appropriate capabilities.
- Capabilities are granted via the signing process, and encoded into the digital certificates with which applications are signed.
- Application certificates are validated at install time. API accesses that require capabilities are also policed at runtime.
- Finally, additional security mechanisms including data caging provide resource protection and privacy for individual applications.

Symbian Signed adds an extra dimension to the security model by specifying test criteria which all applications should meet, providing a baseline for application reliability and stability.

Platform security without Symbian Signed would have locked out third party applications that require access to sensitive system resources. Signing is what keeps the platform open.

It is worth emphasising that, unlike some other Symbian OS idioms, platform security should not be thought of as a discrete architectural piece of the system. For example, there is no box in the System Model labelled 'platform security'. Instead, platform security is a set of pervasive changes at all levels of the system which provide, or enforce, physical protection mechanisms. The signing process is the developer's interface to those mechanisms.

It is also worth disposing of some illusions about signing. It is a long time since Fred Brooks told us that *'there is no silver bullet'*; the slogan applies as much to security as to any other aspects of software. Signing grants capabilities and authenticates developer identity via the mechanism of tamper-proof digital certificates. It therefore provides a basis for trusting software, but it is not the be-all and end-all of trust. On the other hand, knowing the authorship of a piece of software does provide traceability, at least in the digital domain, and traceability is the basis for revocation, or at the very least avoidance.

### 3.1 The signing process

The entry point for all software signing is [www.symbiansigned.com](http://www.symbiansigned.com). Three different signing options are available:

- **Open Signed** enables developers to sign their software using Developer Certificates for quick and easy deployment, for example for testing, onto a limited number of devices, restricted by device IMEI. Open Signed is available both with and without a Publisher ID.
- **Express Signed** enables developers with a Publisher ID to self-certify software for commercial distribution, without requiring independent testing, as long as they do not require the most sensitive System Capabilities.
- **Certified Signed** is the mainstream option for commercial software developers, requiring independent testing, with entitlement to use the *'for Symbian OS'* logo to aid differentiation and brand building.



In outline, the prerequisites and processes are as follows:

- For all options except online signing via Open Signed, developers should start by acquiring a Publisher ID, see below for more information. (Alternatively, investigate the freeware options provided by Symbian publisher certifiers.)
- For all options except online signing via Open Signed, developers will also need to have a Symbian Signed account. Registration is straightforward, via the Symbian Signed web site. (Online signing using Open Signed requires only a simple sign up.)
- Using Open Signed (with a Publisher ID), developers download a certificate request tool from the Symbian Signed portal and use it to generate a certificate request file, which encodes the capabilities their application requires and the device IMEIs to which deployment is required. From the request file, the portal generates a Developer Certificate which can be downloaded and used by the developer to sign an application locally.
- Using Express Signed, developers sign their application locally with their Publisher ID, and upload the application to the portal for signing. An online submission form captures relevant application information, including the set of capabilities the application requires. After validation of the developer's Publisher ID, signing is completed by the portal. The signed application is stored on the portal for the developer to download.
- Using Certified Signed, developers sign their application locally with their Publisher ID, and upload the application to the portal for signing. Again, an online submission form captures relevant application information, including required capabilities and the preferred Test House. The application is then forwarded to the Test House for independent testing. After validation of the developer's Publisher ID and successful completion of the tests, the application is signed by the Test House, and stored on the portal for the developer to download.

How Tos and an up-to-date online guide to the complete signing process can be found online at [www.symbiansigned.com](http://www.symbiansigned.com).

### 3.1.1 Publisher IDs

The developer's Publisher ID provides the basis for authenticating software authorship. A Publisher ID is a digital signature which is issued by a Certificate Authority (CA), and which can be revoked by that authority. In practice it is a variant on the kind of certificates we see through our browsers when connecting to secure web sites using protocols like SSL.

For all signing options except online signing using Open Signed, developers will need a valid Publisher ID to sign software using Symbian Signed. Publisher IDs can be purchased from TC TrustCenter, the Certificate Authority for the Symbian Signed program. For pricing details refer to [www.tctrustcenter.de](http://www.tctrustcenter.de); at the time of writing costs are in the region of \$200 per year.

For developers of freeware, open source, and other not-for-profit software, an alternative to purchasing a Publisher ID is to use the free software signing option provided by Symbian publisher certifiers. Check the Symbian Signed site for details.

### 3.1.2 What's in a signing?

Signing has been described as *shrinkwrap for downloadable content*, and that's a good way to think of it. Like shrinkwrap, signing guarantees that the contents of the package have not been tampered with. Since the contents includes the developer's Publisher ID, signing also therefore guarantees the origin of the software. The two things together provide the foundation for trust.

Signing is certainly not unique to Symbian. It is becoming the norm for online-distributed software. Thus Apple, JavaSoft, and Microsoft increasingly expect, and encourage, ISVs targeting their platforms to sign their software.

Typically certification just assures the end-user that the software they are about to install is untampered with, and that it really was originated by the developer or organisation whose name appears on it. Symbian Signed goes further by specifying test criteria which all signed applications, whether independently tested or not, should meet. While the tests do not guarantee application behaviour, they do provide extra assurances to end users.

How much are those assurances worth? They cannot guarantee every aspect of an application's behaviour, but they do give confidence that it will not block incoming calls, for example, or trash the user's memory card (because applications can only write to their own private data area), or try to bury malicious code somewhere on your phone (because applications must not leave any installation files behind when uninstalled), and that in resource-critical scenarios (like low memory) an application will be well-behaved.

### 3.1.3 Symbian Signed Test Criteria

The Test Criteria, by the way, include all of the following:

- Programs must avoid DLL name clashes; must have reasonable startup time behaviour, including progress indication; must create files only in allowed locations; must uninstall cleanly and completely, and reinstall successfully; it must be possible to see and terminate the application from the task list
- UIDs must be legal and proper and owned by the submitting individual or organization, the SIS file well formed, including correct and consistent versioning, and access to platform/manufacture granted capabilities (more on which below) must be properly approved, as must any test waivers; any Publisher ID (where appropriate) must be valid
- Programs must meet their function specifications (i.e. do what they say they do); must not interfere with system applications like Phone, Clock, Contacts; must not interfere with system events including alarms and high load events like camera or phone device initialisation or VoIP calls, nor with message or call transmission, including sending and receiving SMS and MMS messages, and notifying users of incoming calls
- Programs must survive stress tests including handling exceptional events like OOM (Out Of Memory) or power down while running, as well as rapid and repeated switching
- Programs must behave well with respect to privacy and billable events; must successfully backup and restore data (where appropriate); must demonstrate scalable UI compliance

Only Certified Signed requires independent testing against the criteria, but all submissions are required to list test results for the application being submitted. The Test Criteria are published on the Symbian Signed web site, so you can see for yourself what they are. Symbian Signed also distributes some useful tools to help developers validate applications against the test criteria.

In exceptional cases there is provision for waivers, so that applications which fail some test can apply for a special-case exemption, which will be considered on its merits. Waivers can be approved by licensees, by a network operator, or by a Symbian Signed Test House.

Although independent testing is not required to release Symbian OS applications, mainstream commercial developers should consider the advantages that it gives to applications, including the marketing value of being able to use the Symbian Signed logo. While Test Houses set their own fee scales, at the time of writing they can be as low as several hundred dollars.

## 4 More Thoughts About Platform Security

I have not said much about the details of platform security itself, because it is thoroughly documented elsewhere; Heath (2006) is the authoritative reference, authored by the Symbian architects who designed and implemented platform security, and led its integration into the OS (platform security probably counts as the single most expensive piece of system engineering that Symbian has ever undertaken).

However, it is important to understand the basic principles of platform security to use the signing process effectively. In particular it is important to understand how capabilities are grouped.

Going back to first principles, platform security is based on the ideas of **protection** and **trust**, which are calibrated in terms of **privilege**. These principles come together as follows:

- The protected entities in the system are resources.
- A running software process is considered the unit of trust in the running system.
- Each process needs privileges which are appropriate for the resources it wishes to access.

Platform security translates these ideas into practice by introducing capabilities as the tokens of privilege, and by strictly enforcing process protection (implemented at the hardware level, which makes it both efficient and extremely robust):

- Sensitive resources are protected by capabilities, which operate as the tokens of privilege - to access a protected resource, a process must hold the appropriate capability.
- Data stored in files is protected inside data cages on a per-process basis - only the owning process (or a process with appropriate System Capabilities) can access files in its private caged area of the filesystem, although it can choose to share them with other processes.

The choice of the process as the unit of trust is fundamental, and is where the software and hardware mechanisms come together. For the purposes of this discussion, processes can be thought of as the runtime instantiations of programs. 'Resources' in this model may be services the process provides (its APIs, for example), or hardware it abstracts (via its APIs) or data that it owns (files perhaps, or database content).

The result is a simple model, but a very powerful one.

### 4.1 Capability groups

Capabilities are unique and atomic, which means that no capability can substitute for any other, and in particular there is no notion of a 'stronger' capability somehow allowing access to resources protected by 'weaker' capabilities. (There is no 'superuser'-like capability, for example.)

Platform security distinguishes between User Capabilities and System Capabilities. User Capabilities are designed to be meaningful to mobile phone users; in particular they protect resources that users can be expected to understand and care about, so that asking users to grant or deny them to applications should make sense from the user perspective, for example reading and writing user data or initiating billable events.

System Capabilities, on the other hand, are not designed to be meaningful to end users, but are more concerned with resources which affect the integrity of the device or even the mobile network, for example capabilities that protect system services or device settings, or allow mobile network access.

The most sensitive capabilities, which protect the most sensitive system services, are classified as Device Manufacturer capabilities.

The distinction between the capability groups is important, because not all signing options are equal in terms of which capabilities they can grant. The table below summarises the capability groupings, and how they relate to the available signing options.

Capability type	Capability name	Description
User Capabilities	Local Services Location NetworkServices ReadUserData UserEnvironment WriteUserData	User Capabilities are designed to be meaningful to mobile phone users  Depending on device manufacturer security policies, users may be able to grant blanket or single-shot permission to applications which use these capabilities  Available through all signing options
System Capabilities	PowerMgmt ProtServ ReadDeviceData SurroundingsDD SwEvent TrustedUI WriteDeviceData	System Capabilities that protect system services, device settings, and some hardware features <ul style="list-style-type: none"> <li>Available through all signing options</li> </ul>
	CommDD DiskAdmin NetworkControl MultimediaDD	System Capabilities that protect file system, communications, and multimedia device services <ul style="list-style-type: none"> <li>Available through Open Signed (with Publisher ID) and Certified Signed options only</li> </ul>
Device Manufacturer Capabilities	All Files DRM TCB	Trusted Computing Base and System Capabilities that protect the most sensitive system services <ul style="list-style-type: none"> <li>Available through Open Signed (with Publisher ID) and Certified Signed options only</li> <li>Require device manufacturer approval</li> </ul>

Table 1: Symbian OS capabilities

As the table shows:

- User Capabilities are available through all signing options
- All System Capabilities (as defined in the table above) are available through Open Signed (with a Publisher ID) and Certified Signed options
- Express Signed does *not* allow access to CommDD, DiskAdmin, NetworkControl, and MultimediaSystem Capabilities

Symbian Signed refers to the most sensitive capabilities, specifically AllFiles, DRM, and TCB, as Device Manufacturer Capabilities. These are only available through the Open Signed (with a Publisher ID) and Certified Signed options and require device manufacturer approval.

It is still the case that each capability stands on its own so far as unlocking access to resources is concerned; for example, the NetworkServices capability is required to access protected network services APIs, and no other capability will do.

## 5 Wrapping it all up

If you want to write freeware, you don't need a Publisher ID, and you don't need to worry about the signing options; you can publish through one of the Symbian publisher certifiers. If you just want to experiment with code, and aren't (yet) considering distribution at all, then you can use Open Signed, with or without a Publisher ID. If you think your mobile game will put Doom in the shade, and you want to make money out of that, you should definitely invest in a Publisher ID, and weigh up the merits of Express Signed (lower cost) versus Certified Signed (fewer restrictions and access to the Symbian Signed logo).

Is Symbian OS still open? Yes, platform security and Symbian Signed keep the platform open to third party native applications, which is the critical definition of openness. In fact, they aim to protect the openness of the platform, by building market confidence and trust in third-party applications. (And 'market confidence' means more than just end-user confidence, it means operator and manufacturer confidence too.)

Is Symbian Signed really open? Yes, because any developer can take advantage of free and low-cost signing options to get applications out into the marketplace, scaling all the way up to full scale tested and signed applications for wide scale commercial distribution.

Meanwhile Symbian Signed has evolved a lot since its first roll out, and continues to evolve as it tries to strike a balance between ease of use for the individual developer and the needs of the wider marketplace.

## 6 References and resources

The Bruce Schneier references are to *Secrets and Lies*; and the quote from Ed Felten can be found in Heath (2006). 'My book' is Morris (2007). A good example of the network operator perspective on security can be found in the forward to Heath by Tim Wright of Vodafone. An excellent, all round primer for a host of security topics is Cheswick, Bellovin and Rubin (2003). And the Fred Brooks reference is to Brooks (1986).

Signing as 'shrinkwrap for software' is a nice phrase used by VeriSign: *'Digital IDs are virtual 'shrinkwrap' for your software; if your code is tampered with in any way after it is signed, the digital signature will break and alert customers that the code is not trustworthy.'*

The Steve Jobs iPhone quote is published on the Newsweek website. And the Symbian OS cumulative shipment numbers are based on Symbian's Q3 2007 results.

## 6.1 Reading

### Craig Heath (2006)

*Symbian OS Platform Security, Wiley/Symbian Press*

- The authoritative and highly-recommended introduction to platform security on Symbian OS

### Richard Harrison and Mark Shackman (2007)

*Symbian OS C++ for Mobile Phones, Wiley/Symbian Press*

- The latest version of the classic book, with an excellent section on platform security and signing

### Jo Stichbury and Mark Jacobs (2006)

*The Accredited Symbian Developer Primer, Wiley/Symbian Press*

- Runs through all you need to know to make sense of the Accredited Developer exams, including platform security and Symbian Signed

### Ben Morris (2007)

*The Symbian OS Architecture Sourcebook, Wiley/Symbian Press*

- The first in-depth guide to Symbian OS architecture.

### Bruce Schneier (2004)

*Secrets and Lies, Wiley*

- Classic and thoroughly readable introduction to computer and network security topics.

### William R. Cheswick, Steven M. Bellovin and Aviel D. Rubin (2003)

*Firewalls and Internet Security: Repelling the Wily Hacker, Addison-Wesley*

- Hands-on and nitty-gritty reference to security topics including PKI and the like

### Fred Brooks (1986)

*No Silver Bullet – essence and accidents of software engineering*

- Published in The Mythical Man-Month

## 6.2 Web resources

The Symbian Signed site is at [www.symbiansigned.com](http://www.symbiansigned.com).

Symbian Developer Network is at [developer.symbian.com](http://developer.symbian.com).

TC Trustcenter is the Certificate Authority for Symbian Signed as of Q2 2007, at [www.tctrustcenter.de](http://www.tctrustcenter.de). At the time of writing (Q4 2007), the URL for purchasing a Publisher ID is <http://www.trustcenter.de/order/publisherid/dev>.

The Symbian Signed Test Criteria, and more information about Test Houses, is published on the Symbian Signed web site.

## 7 Author Profile



Ben Morris freelances as a writer and software architect specialising in Symbian OS. He is the author of *The Symbian OS Architecture Sourcebook: Design and Evolution of a Mobile Phone OS*, published by Symbian Press. He can be contacted through [www.benmorris.eu](http://www.benmorris.eu).

Want to be kept informed of new articles being made available on the Symbian Developer Network?

[Subscribe to the Symbian Community Newsletter.](#)

The Symbian Community Newsletter brings you, every month, the latest news and resources for Symbian OS.