Reference information

Ericsson Mobile Companion MC 218

First edition (June 1999)
This reference information is published by Ericsson Mobile Communications AB, without any warranty. Improvements and changes to this text necessitated by typographical errors, inaccuracies of current information or improvements to programs and/or equipment, may be made by Ericsson Mobile Communications AB at any time and without notice. Such changes will, however, be incorporated into new editions of this reference information.

TABLE OF CONTENTS

1. **Purpose** 5

2. **Hardware and technical details** 6
   2.1 General specifications 6
   2.2 Technical Data 6
   2.5 Accessories 7
   2.6 Interfaces 7

3. **Compatibility and PC Synchronization** 9
   3.1 Compatible Phones 9
   3.2 Digital Camera Compatibility 9
   3.3 PC Software Compatibility 9
   3.4 EPOC Compatibility 10
   3.5 PC Synchronization 11

4. **Mobile Internet – WAP Browser** 15
   4.1 Features in Mobile Internet WAP Browser 15

5. **Third Party Software** 18

6. **Ericsson Mobile Internet** 19

7. **IMAP4 functionality** 20
   7.1 Limitations and omissions 21
   7.2 Glossary 21

8. **EPOC** 22
   Summary 22
   8.1 Introduction 22
   8.2 Core 23
   8.3 Applications 34
   8.4 Communications 42
   8.5 EPOC R5 New Features 59

9. **EPOC Software Development** 69
   9.1 C++ SDK 69
   9.2 OPL SDK 69

10. **Infrared communication** 70
   10.1 General Features of Infrared Communication 70
   10.2 Key benefits 70
   10.3 Ericsson Mobile Companion 70
   10.4 Ericsson Infrared Modem 70
   10.5 IrDA – Infrared Data Association 71

11. **IrDA** 73
   11.1 IrDA's New Full Range of Digital Information Exchange via Cordless Infrared Connections 73
   11.2 IrDA DATA - Hardware/Protocol Stacks 74
   11.3 IrDA CONTROL 74

12. **GSM Data** 76
   12.1 General Features of GSM Data 76
   12.2 Key Benefits 76
   12.3 GSM & GSM Data – An introduction 76
   12.4 How does GSM data work? 79

13. **Serial Terminal Support, Comms** 82
   13.1 How script commands are used 82
   13.2 To store information as a name 83
   13.3 Script commands 85

13.4 Glossary of commands 87
13.5 Commands 88

14. FAQ 102
14.1 General 102
14.2 Applications 103
14.3 Communication & Synchronisation 107
14.4 Hardware 109
14.5 Hints & Tips 109
14.6 Record 111
14.7 System 111

15. Index 113

1. Purpose

The Ericsson MC218 Reference Information is designed to give the reader a deeper technical understanding of how the MC 218 is constructed, and also how it interacts with other media.

People who can benefit from this document include:

- Corporate buyers
- IT Professionals
- Software developers
- Support engineers
- Business decision-makers

2. Hardware and technical details

2.1 General specifications

The MC 218 is a mobile companion, which works with the EPOC operating system. In addition it has a built-in software modem, extra compact flash backup memory slot, a PC synchronization cable and a phone cable. It is packaged in a stylish carrying case, giving a powerful communication tool the look and feel of a filofax.

2.2 Technical Data

Physical Specifications	172 x 89 x 24 mm 350 gr. with batteries
Memory	12 MB ROM 16 MB RAM Compact Flash memory slot 32 bit ARM SR 710 CPU running at 36 MHz
Touchscreen Display	640 x 240 pixels (½ VGA) 16:1 contrast, 16 greyscales Pen and touch interface Bright backlight
Input/output	Built-in serial interface for connection to an RS232 device Built-in infrared port. Compact Flash card slot Audio speaker Microphone and Voice Note capability
Power	Two standard AA batteries One 3v CR 2032 coin cell backup battery
Operating Requirements	Operating temperature 0° to 40° C(32° to 104° F) Storage temperature 0° to 60° C (32° to 140° F) Humidity 90% relative humidity at 40° C (104° F) maximum

2.3 MC 218 Design

The Ericsson Mobile Companion MC 218 has a sleek, oblong design. The sophisticated outer casing is made of dark grey plastic, with a lid of bluish grey plastic with the Ericsson logo placed on the front in an indentation within the plastic.

At the back of the unit there is a connection for RS 232 cables, an infrared port, a main battery compartment covered by the battery cover and a small speaker. On the left-hand side of the MC 218 you can find a plug for the mains cable and a slot for the 3-volt CR2032 backup battery. A touch pen slides into the right side of the unit. Located just beside the pen there is a slot for a compact flash card and a microphone. On the front of the MC 218 there are three buttons (record, stop and play) for the Voice note function.

The unit has two hinges covered with a piece of dark grey plastic at the back, allowing it to be easily opened and closed.

The MC 218 has a sizeable keyboard with large, dark grey keys and white letters. Several extra functions on certain keys are operated in conjunction with the Fn key. The extra functions are printed on the keys below the letter, digit or main function. By pressing the Fn key at the same time as the space key the backlight is switched on.

The black and white screen of the MC 218 has ½ VGA width, 640 x 240 pixels, 16:1 contrast, 16 greyscales and a bright backlight. Below the screen quick launch icons are located for some of the most important applications (Desktop, Contacts, Calendar, My Phone, Message, Internet, Word, Sheet and Extras).

2.5 Accessories

Accessories for the MC 218 are already on the market but they are not Ericsson branded. A mains adaptor and compact flash memory disks are also available.

2.6 Interfaces

2.6.1 Power Supply

The MC 218 can be powered either through batteries or a DC adaptor. Included in the MC 218 package are two LR6 alkaline batteries and a 3V backup battery. The backup battery is placed in the holder located underneath the MC 218 and the two alkaline batteries are placed in the holder at the back of the MC 218. You can also use an external power supply, a 6V DC power adaptor. **Important!** Only use an adaptor, which has the positive connector in the centre and the ground connector in the shield.

2.6.2 RS 232 Serial Port

The RS 232 port is located at the back of the MC 218. This port is used for serial communication with a PC to synchronize files. The RS 232 is a serial protocol and the MC 218 supports the standard RS 232 protocol.

2.6.3 Infrared Modem

The built-in infrared modem is located at the back of the MC218, next to the RS232 serial port. The modem has been developed according to the IrDA standards (Infrared Data Association). The IrDA Infrared data link is compatible with most Ericsson GSM 900/1800/1900 Phones (see the chapter about compatibility). IrDA 1.0 allows data to be transmitted at the speed of 115.2 kbps but currently the GSM system can transfer data at a speed of 9.6 kbps.

For the best performance, place the Ericsson Mobile Phone and infrared modem no more than a metre away from and at a 30° angle to the Ericsson Mobile Companion's infrared port.

The modem’s power consumption in operation is between 20-40 mA depending on compression rate and the power consumption in standby is less than 2 mA.
When using the Ericsson Mobile Phone with the Ericsson Infrared Modem attached the power consumption increases to about 10% additional power consumption compared to using the mobile phone without the infrared modem.

Both the 3V and 4V Ericsson Infrared Modem adaptor can be used together with the MC218. Included in the box is a 4V-modem adaptor. However, the customer will be given the opportunity to upgrade to a 3V-modem adaptor by means of a voucher supplied in the box.

3. Compatibility and PC Synchronization

3.1 Compatible Phones

The idea of the MC 218 is that it provides a mobile phone and palmtop, which work together in perfect harmony. The concept is one of a divided solution where you can "Take what you want where you want." This means that the phone is completely compatible with all Ericsson four-volt and three-volt GSM 900/1800/1900 phones. These phones are currently:

Model	Displayed as	Settings	AT+CGMM Response	Comments
628	628	None	1050702	
688	688	None	1050701	
768	768	None	1070603	old 768 e.g. PF 768 rev. 971114
768	768/788	None	1050601	old 768
CF 788	788	None	1020602	CF 788
GF768	768/788e	Full	1050612	e.g. GF 768 rev. 980910
GF768c	768/788	None	1050601	e.g. GF 768c rev. 981104
788	768/788	None	1050601	old 788 (before 788e)
GF788c	768/788	None	1050601	e.g. GF 788c rev. 980826
788e	788e	Full	1050601	first generation 788e e.g. rev. 980428
788e	768/788e	Full	1050612	second generation 788e
S868	868/888	Full	1100801	S868
SH888	868/888	Full	1100801	SH888
888	888	Full	0100801	
CF888	888	Full	1120801	CF888
I 888	888	Full	1140801	e.g. I 888 rev. 990224
A1018	A1018	Full	1100901	A1018
T18	T18	Full	1101001	T18
T28	T28	Full	1101101	T28 (pre-production rev.)
T28	T28	Full	1101101-BV	T28s

3.2 Digital Camera Compatibility

Included in the MC 218 is a unique Ericsson program, Postcard. This program allows the customer to send pictures, drawings and photographs via the e-mail system in a functional and user-friendly way. Together with this program it is possible to use a digital camera with an infrared interface to transmit photographs directly to the MC 218 and send the photograph via e-mail. The MC 218 should be compatible with all digital cameras that use the IR Tran-P. Cameras that have been tested and verified as compatible with the MC 218 are:

- Casio QV-7000SX
- Sharp VE-LC25
- JVC GC-S1

3.3 PC Software Compatibility

The MC218 is developed to be compatible with a large range of PC software. The compatible software is:

For EPOC Word:
- Microsoft Word 97, 95, 6.2
- WordPerfect 8, 7, 6, 5.1
- Lotus Ami Pro 3.0, 3.1

- Microsoft Works 4, 3

For EPOC Sheet
- Excel 97, 95, 5, 4
- Lotus 123 wk 4, 3,1
- Quattro Pro 8, 7, 6, 5

For EPOC Data
- Access
- Foxpro 3, 2.6, 2.5, 2.0
- Dbase 5, 4, 3
- CSV

For PIM (Contacts and Calendar)
- Lotus Organizer 97, 97 GS
- Schedule+ 7.5, 7
- Outlook 98, 97

3.4 EPOC Compatibility

The EPOC release ER5 used by the MC 218 is completely backward-compatible with earlier versions of the EPOC operating system, so most of the existing EPOC programs will work perfectly with the MC 218.
However, you should be aware of the following:

1. A few third party OPXs used undocumented and unsupported features of the OS (e.g. obtained by reverse-engineering INI files etc). This means that these third-party developers will need to change, rebuild and re-release their OPXs in order for them to work on ER5 devices.

2. However, one major third-party OPX developer (responsible for Systinfo.OPX etc) has agreed to transfer their OPXs to Symbian to develop and support. Symbian will be releasing these OPXs into EPOC World shortly, including a version of Systinfo that runs on ER5.

3. A new ER5 DBMS utility OPX called dbUtils will also be available. This OPX helps overcome certain incompatibilities in ER5 DBMS. This OPX (and the incompatibilities) are documented in the ER5 OPL SDK.

4. WINS OPXs are completely incompatible. This is due to BC changing between ER3 and ER5. The implications are that developers will need to rebuild and re-release any WINS versions of their OPXs (end-users are not affected by this). This change is documented in the ER5 C++ SDK.

3.5 PC Synchronization

PC synchronization, and interaction with a desktop PC, is extensive with the MC 218. The MC 218 is not only a perfect mobile communication companion but also a very good companion for your desktop PC.

3.5.1 Ericsson EPOC Connect

The basis for MC 218 to PC information exchange is Ericsson EPOC Connect. Ericsson EPOC Connect is built around a plug-in strategy to enable extra functionality to be included, as user needs change and extend. Although Ericsson EPOC Connect is packed with functionality which is described in each section below, we know that there will be consumers with more ideas and the plug-in strategy is an extension of the openness which is one of the leading design objectives for the MC 218. Already today there are a several of plug-ins available from Third party SW vendors.

Ericsson EPOC Connect runs on both Windows 98/95 and NT 4.0.

Connection can be done over a normal serial connection using RS 232 and the supplied cable or by the use of IR communication. IR communication works only on Windows 98/95 because Windows NT lacks IR support.

The number of units that can talk to each other are unlimited. A PC can be partner with several different MC 218s. This is vital if, for example, a family has one PC but each family member has their own MC 218.
One MC 218 can also be partner with several PCs. This ensures that information from both the work PC and the home PC can be synchronized with the MC 218.

3.5.2 Synchronization of e-mail

The MC 218 is a perfect product for mobile e-mail. The message client is easily accessed with a Quick Launch icon and the mail program handles both older and newer mail protocols (POP3 and IMAP4). When the user is close to the PC it is normally faster and cheaper to use the synchronization feature between the PC and the MC 218 for e-mail compared to downloading them through GSM. The e-mail synchronization on the MC 218 enables synchronization not only of the Inbox, but also of the Sent, Outbox and Draft folders. Synchronization can be configured in several ways to optimize use and time consumption. For example the size of a message and the time window for synchronization can be limited.

The synchronised e-mails are stored in a separate folder on the MC 218, which makes it easy for the user to remember where the e-mail message comes from.

3.5.3 Synchronization of documents

A mobile user needs reference information so it is important to be sure that the user has the same information in all his or her devices. The synchronization of documents offers the user an automatic way to ensure that the same information is always available on the MC 218. Document synchronisation can be set up to run automatically, which means that the user is always sure to have the latest info available.

Synchronization also works in the other direction. When the user has been away from the PC for a while and updated documents, taken notes and much more, the new updates are automatically synchronized on the PC thus ensuring that the user has the information available on both devices.

The synchronization of documents handles all formats supported by Ericsson EPOC Connect. These are:

For EPOC Word:
- Microsoft Word 97, 95, 6.2
- WordPerfect 8, 7, 6, 5.1
- Lotus Ami Pro 3.0, 3.1
- Microsoft Works 4, 3

For EPOC Sheet
- Excel 97, 95, 5, 4
- Lotus 123 wk 4, 3,1

- Ouattro Pro 8, 7, 6, 5

For EPOC Data
- Access
- Foxpro 3, 2.6, 2.5, 2.0
- Dbase 5, 4, 3
- CSV

3.5.4 Synchronization of Contacts, Calendar and To-Do List (PIM)

In everyday life access to an updated Calendar and addresses of friends and business colleagues is greatly appreciated. The Calendar program on the MC 218 can be synchronized with the Calendar/Agenda program on the user's PC. Contacts and the ToDo/task list can also be synchronized.

Synchronization can be configured to start at regular intervals, at each connection between the PC and MC 218 or when the user decides.

The growing use of groupware SW such as Microsoft Exchange and Lotus Notes means that more and more meetings are booked electronically in daily business life. This encourages users to have their calendar electronically stored on the server/PC and then to update data to or from their MC 218s.

The MC 218 supports the following groupware by default:
- Lotus Organizer 2.1, 97, 97 GS
- Microsoft Schedule+ 7.5, 7
- Microsoft Outlook 98, 97

From third parties there are plug-ins that support:

- Lotus Notes (Agenda and Contacts) - provided by TIME IS Ltd.
- ACT! (Agenda and Contacts) - provided by Advansys
- Novell groupwise (Agenda and Contacts) - provided by Advansys

More third party programs will be available at Ericsson Mobile Internet: http://mobile.ericsson.com/mobileinternet.

3.5.5 Backup & Restore

When the worst happens, it is always best to be prepared. Ericsson EPOC Connect includes a very resourceful backup and restore facility. This can save the user a lot of work if the batteries wear out or the user deletes information by mistake.

Backup can be done on a regular basis (user-configurable) or at each connection. The backup is intelligent and only the changes made since last backup will be recorded.

It is possible to restore not only the complete machine but also separate files. The latter feature can be a very helpful when a user deletes a file by mistake.

3.5.6 CopyAnyWhere

The MC 218 not only makes it possible to copy and transfer complete documents and files between the PC and the MC 218, but it also allows parts of a document to be transferred easily to the other device.

When the user has copied a text on the PC it is directly available to be pasted into a document on the MC 218 and vice versa.

This function works for all document types that Ericsson EPOC Connect handles (see the section "Ericsson EPOC Connect").

3.5.7 Offline web browsing

One way of making sure that important information can be with the user all the time is to use the document synchronization described earlier.
These days a lot of information is also stored on web pages and the offline web browsing synchronization makes it possible to download selected pages automatically to the MC 218. This means that the user can decide which pages are vital and updated versions of these pages are then automatically transferred to the MC 218. The information could be the latest news, a newspaper, business information or even the latest list of recommended restaurants.

This function requires the following:

- Microsoft **Windows 95** with Microsoft **Internet Explorer 4.0** or better, *or* Microsoft **Windows NT** 4.0 with Microsoft **Internet Explorer 4.0** or better, *or* Microsoft **Windows 98**.
- A working default **Internet connection** (working with Microsoft Internet Explorer).
- For scheduled web retrieval **Task Scheduler** must be installed (part of Internet Explorer 4.0 and Windows 98).
- **Ericsson EPOC Connect,** for the connection between your PC and your EPOC Device.
- **EPOC Web Browser** for offline browsing of your subscriptions.

3.5.8 File and document transfer

Documents can be synchronized as described in the section "Synchronization of documents".

Documents can also be moved/copied one by one or in groups. The transfer and copying can be done in both directions, both from the MC 218 to the PC and the other way around.

This functionality works for all formats that Ericsson EPOC Connect handles, which by default are:

EPOC Word and back

- Microsoft Word 97, 95, 6, 2
- WordPerfect 8,7,6,5.1
- Lotus Ami Pro 3.0, 3.1
- Microsoft Works 4,3

EPOC Sheet and back

- Excel 97, 95, 5,4
- Lotus 123 wk 4,3,1

- Ouattro Pro 8,7,6,5

EPOC Data and back

- Access
- Foxpro 3, 2.6,2.5,2.0
- Dbase 5,4,3
- CSV

4. Mobile Internet – WAP Browser

Included in the Ericsson software package delivered with the MC218 is the world's first commercial WAP browser. It uses a completely new way of communicating with networks, which gives a faster and better performance than any ordinary web browser. It enables the user to access data specially designed for the mobile user.

Mobile Internet is a browser using a standard called Wireless Application Protocol (WAP) that uses a language called Wireless Markup Language (WML). This standard was specially created for wireless communication via mobile companions such as the Ericsson Mobile Companion MC 218 and mobile phones.
The typical WAP client is a device with a limited user interface, low memory and computing power compared to desktop and laptop computers, connected to a (slow) wireless network. More specifically, this includes the following devices: mobile phones, pagers, smart phones (mobile phones with an improved user interface and hardware and software resources, compared to the standard voice/SMS phone), mobile companions and other small devices. WAP is not a browser meant for desktop or laptop computers. Thus, WAP will not appear in the majority of today's Internet WWW clients. Instead, WAP is created for the increasing fraction of Internet clients that are mobile companions and mobile phones, and mainly are used to *access* information, rather than to *create* information. When you access a web site built with WML, you will be able to download information quicker than you would be able to access HTML pages with a traditional web browser using the HTML standard. The WAP browser is constructed for WML (Wireless Markup Language) and cannot read ordinary HTML pages but it is suitable for interaction with customer services offering e. g. ticket reservation. It is also handy when you want to access text-based information, such as timetables, share prices and exchange rates, Internet banking and other interactive services. The MC 218 is compatible with the WAP Conference specifications generation 1.1 and uses circuit switched data as a bearer.

For more information please visit:
http://www.wapforum.org.

4.1 Features in Mobile Internet WAP Browser

Features	MI 1.06
Authentication	Full WAP (Server and Gateway)
Bookmarks	Hierarchical. Stored in a single file, internal format. Editing and management facilities.
Bookmark Export	No
Bookmark Import	No (Possible from initialising file)
Bookmark Subscriptions	No
Cache	User can choose to persist between sessions. User can select Size.
Clipboard	Support text into clipboard, not images. Text is plain. Cells of tables copied with paragraph breaks.
Colour	Images in 4 or 16 grey shades.

Find Text in Page	No.
History List	Yes. User can select number of days to save.
Bearer	UDP / IP
WAP / WML	Supports full WAP 1.1. WMLC support (scripting) The following layers are implemented: WAE, WSP(connection-less),WDP
HTTP	HTTP 1.1. Name sent as (TTP_USER_AGENT): MC218 2.0 WAP1.1
Image Animation	NO
Image Formats	GIF (interlaced + non-interlaced) images. WBMP (WML Bitmap) No transparent layers in images. All image formats in B&W.
Infrared	Send and receive from URL bar only.
Local Files	Support for reading compiled WML.
Logging	Yes.
MailTo	Launches mail application (no automatic tasking back).
Memory Usage	Installation: 500 kByte Running: 600 kByte + cache (0 – 1 MByte)
MIME Mapping System	Yes.
Other URL Schemas	File. MailTo.
Pen & Keyboard Input	Pen and keyboard support.
Printing	None.
Saving	Save as text.
Security	None
Send Document As	None.
Sound	None.
Text Files	None.
Unicode	No
Video	None.
View Source	None.
WML Deck size	4 KByte
URL size	256 in location bar. “WML Deck size” in WML code.

5. Third Party Software

A wide range of third party software already exists today and is available for the MC 218. They vary from entertainment to economics and navigation. In the MC 218 Software & Accessories Catalogue a large number of software vendors and their products can be found. Another place filled with EPOC applications is the Ericsson Mobile Internet site:
http://mobile.ericsson.com/mobileinternet

6. Ericsson Mobile Internet

As an Ericsson Mobile Companion MC218 owner, the customer gets instant and free access to the exclusive information on the Ericsson Mobile Internet, a site that is optimized for mobile use. The customer can access the information without having a personal Internet subscription. The address to the Ericsson Mobile Internet site is:
http://mobile.ericsson.com/mobileinternet

7. IMAP4 functionality

The IMAP4 MTM provides the following functionality to the Messaging Application:

Definable port number

The user may specify which port number s/he wishes to use to connect to the IMAP4 server. The default port number is 143, as specified by the IMAP4 protocol specifications.

Basic functionality

Create, delete and rename folders on the IMAP4 mail server.
Copy and move emails from one folder on the IMAP4 mail server into another folder (except for the inbox) on the same mail server.
Delete emails from the IMAP4 mail server.
Copy or move emails from the remote server into a mail folder on the EPOC device.

Automatic resynchronisation

When online, the IMAP4 MTM will check the contents of the remote inbox on a regular basis so that the user may see new emails in the remote inbox as soon as they arrive on that IMAP4 mail server.

Subscription

The IMAP4 MTM supports folder subscription; it will allow the user to subscribe to, or unsubscribe from, any folder on the IMAP4 server, except for the Inbox (which the user is always subscribed to). The list of subscribed folders is refreshed at the start of each connection to the IMAP4 mail server.

Simultaneous server access by multiple clients

If supported by the IMAP4 mail server, the IMAP4 MTM may connect to an IMAP4 email account which has already been opened by another user of the same mail account.

Simultaneous access to multiple IMAP4 mail servers

The user may connect and interact with more than one IMAP4 mail server at the same time.

MIME compatibility

The MTM is able to recognise and decode standard MIME emails; i.e. it is able to decode quoted-printable text and BASE64-encoded binary data that may exist within the header or in the body of an email message.
MIME-encoded information within an email header can also be decoded for the following character sets: ISO-Latin1, UTF-7, UTF-8.

Server Compatibility

The IMAP MTM has been tested successfully with the following IMAP4 Rev1 mail servers:

1. Cyrus IMAP4 Server
2. University of Washington IMAP4 Server
3. Microsoft Exchange IMAP4 Server
4. Netscape Messaging IMAP4 Server
5. CC:Mail IMAP4 Server

7.1 Limitations and omissions

- Plain text messages: Since IMAP4 is intimately related to the MIME standards, emails sent to an IMAP4 server which are not in MIME format will not be recognised nor decoded correctly by the IMAP4 server, nor by the IMAP4 MTM, resulting in the contents of some plain text emails being formatted incorrectly. This is a limitation of the IMAP4 protocol.
- Email priorities. IMAP4 does not easily allow the mail client to determine the "priority" (e.g. high, normal, low) of an email stored on the mail server. The IMAP4 MTM therefore cannot display the priority of emails on the remote mail server, or in emails that are downloaded from the remote mail server onto the EPOC device.
- Uploading emails: The IMAP4 MTM does not support copying or moving email messages from the EPOC device up onto the remote server (i.e. the IMAP4 "APPEND" command). Email messages may be sent to the remote mail server from the EPOC device via the usual route of an SMTP gateway.
- The IMAP4 MTM does not support "remote folder subscription" - it will ignore the subscription information retained on the remote mail server.
- Server defects: Since IMAP4 is a complicated transport protocol, many implementations of IMAP4 servers do not conform precisely to the IMAP4 rev1 standard. Errors may therefore occur whilst communicating with an IMAP4 mail server which are due to defects in the server software, as opposed to being due to defects in the IMAP4 MTM.
- Modified UTF-7: The IMAP4 does not at present support the modified UTF-7 text encoding scheme that is defined in the IMAP4 specification.
- V2.00 of the Messaging Application, and hence the IMAP4 MTM does not support MHTML Email; body text received in HTML format within an email message will be placed in an attachment file called "attachment" by the IMAP4 MTM.

7.2 Glossary

The following technical terms and abbreviations are used within this document.

IMAP4 rev1	"Internet Message Access Protocol, Version 4rev1" an electronic mail transport protocol which allows sophisticated mail operations to be carried out on a remote mail server by a client.
MTM	"Message Transport Module" a software "plug-in" which may be installed onto an EPOC device to add dynamically a new transport protocol service and new functionality to the EPOC Message Application V2.00
Remote server	The IMAP4 mail server
RFC	"Request For Comments" an Internet document which defines an Internet standard

8. EPOC

Summary

This chapter is intended to give sufficient information to form a sound technical evaluation of the 32-bit EPOC operating system.
EPOC is a whole operating system encompassing a base, graphics, applications, Java runtime, wireless communications protocols and applications, SDKs and many other features. The content of this chapter is based on EPOC Release 5.
For more information on Symbian, see:
http://www.symbian.com

8.1 Introduction

The first public release of EPOC, in April 1997, marked the beginning of a new generation of operating systems built upon the long track record of the Psion Group in the handheld and mobile computer industry.

Eleven years previously, in 1984, Psion invented the personal organizer. The Organiser II, which debuted in 1986, has sold over a million units to personal and corporate customers. It had an eight-bit CPU, and could be programmed in assembler or a high-level BASIC-like language dubbed Organiser Programming Language, or OPL.

From 1991, the Psion Series 3 range drove the rapid expansion of the personal organizer as a popular consumer device. The new system software was named SIBO, for "sixteen-bit organizer". Conscious that the Series 3 range would herald a new epoch in personal convenience, Psion named the OS core EPOC — the origin of the name now used for Symbian's 32-bit EPOC. SIBO was more portable than the Organiser II's OS, and pioneered the application/engine split, which has become a vital part of 32-bit EPOC. However, because most of its EPOC core was written in assembler, and the x86's 16-bit segmenting heavily influenced its architecture, SIBO's portability was restricted to the x86 processor architecture. SIBO included an object-oriented GUI and frameworks layer, while an OPL editor/compiler could be used to write powerful programs directly on the machine. This gave rise to thousands of useful third-party application programs. The Series 3 range was expanded and diversified for the consumer market, while the WorkAbout range, introduced in 1995, was created for ruggedized industrial applications. Altogether, the Psion Group has sold over 1.5 million SIBO machines. At their peak, the Series 3 range commanded 35% of the world market for personal organizers. This success is primarily due to the richness and usability of SIBO applications, the robustness and efficiency of the multi-tasking OS which ensures that users' data is always instantly available, the high quality of Psion's ROM-based software, very long battery life, and elegant hardware design.

Like other areas of technology, handheld computers continued to become more powerful. By 1994, the 64k limits of 16-bit systems were beginning to present a serious barrier to software development, and it was becoming clear that a world-class operating system had to be portable to a wider range of devices. A new EPOC was conceived: while it would carry forward the best features of Psion's previous operating systems, it would also be entirely 32-bit, and would be portable to any target CPU type and machine architecture. This would allow EPOC to become an open system, licensed for use by non-Psion original equipment manufacturers (OEMs). To facilitate this, Psion Software was formed as a separate company in 1996 with a mission to make EPOC the leading software platform for wireless information devices, by licensing it to a wide range of OEMs in these fields. EPOC Release 1 was ready by April 1997, when it emerged in its first product, the Psion Series 5.

Licensing activity to non-Psion OEMs had already begun before EPOC Release 1 was available. It became clear that EPOC's efficiency and flexibility, combined with Psion Software's no-nonsense technical and commercial culture, formed an outstandingly attractive basis for system software in wireless information devices. Psion Software was de-merged from the Psion Group, and became Symbian Ltd, owned by Psion, Nokia and Ericsson, with Motorola also intending to join.

Symbian will continue to develop software for smartphones and communicators, and will use the increased investment, resources and expertise of its owners to expand its development, participate more widely in setting industry standards, and increase its licensee and developer base.

8.2 Core

EPOC is a whole operating system encompassing a base, graphics, applications, Java runtime, wireless communications protocols and applications, SDKs and many other features. The components described in this chapter help to define the shape of EPOC and distinguish it from other OSs intended for desktop or hand-portable computers: in this sense, they are truly the core technologies of EPOC.
This chapter is part of an EPOC overview series: other chapters look at the whole of EPOC, communications, software development options including the Java environment, applications, and data synchronization with PC-based applications.

8.2.1 Introduction

EPOC is delivered as a set of components, which are built into:

- ROMs for execution in target machines (components may also be run from flash ROM, removable media, or RAM)
- the Windows-hosted EPOC emulator
- the WINC Windows utility package
- EPOC Connect, for PC-based data synchronization, backup, software installation etc
- tools and documentation that go into developer SDKs and OEM toolkits

The components described in this paper help to define the shape of EPOC and distinguish it from other OSs intended for desktop or hand-portable computers. The core components can be grouped into the following major groups:

Base	runtime and tools for building ROM, emulator and WINC components
Engine Support	components without any user interface, that are used by application engines
Graphics	components for drawing graphics, printing, fonts and re-usable views
System and GUI	graphical user interface, system shell, control panel and other components which define the look-and-feel of an EPOC machine, and provide a basis for its graphical programming

8.2.2 Base

The base provides the EPOC runtime, APIs for higher-level programs, and toolchain used for building programs and ROMs.

8.2.2.1 Portable runtime system

EPOC is a portable operating system with three major implementation families:

- target machines: EPOC is a full OS which boots on a ROM-based device and manages all aspects of that device — scheduling, memory, power, timers, files, keyboard, pointer, screen, PC Cards, CF-card removable media, etc. Only the ARM3 processor architecture has been made available for general use. EPOC has however run native on x86 PCs, ARM4 and StrongARM. Support for ARM's Thumb architecture, and for Motorola's M*Core architecture, is under development.
- windows-based emulator: EPOC presents the same user-side APIs as its machine implementations, but rather than running native on device hardware, EPOC calls Win32 services to provide a highly effective emulator for software development and demonstration purposes. File access is constrained to specified subdirectories, to prevent uncontrolled access to the user's PC files by software under development.
- windows-based tools, the so-called WINC environment: EPOC presents user-side APIs which are binary compatible with the emulator build, but the runtime includes no

graphics support and no file mapping. WINC is intended to provide EPOC APIs for higher-level Windows-based programs, including EPOC Connect (which uses it to access EPOC data through application engines already developed for the emulator), the EPOC help builder, and other utilities.

EPOC's runtime system comprises two components, cryptically titled E32 and F32. E32 provides the kernel executive and server, with a kernel API providing services to device drivers. F32 provides a bootstrap loader, file services, an API for writing new file systems, and a command shell for use primarily in test ROMs.

When implementing EPOC on a new target device, a major milestone is the implementation of a ROM, which runs E32, F32 and the text shell. In terms of capability, such a ROM is something like a clean and powerful DOS command line: the text shell provides simple utility and debugging commands, and a DOS-like batch language — but there is no legacy 16-bit architecture, and the multi-tasking infrastructure is much more powerful.

8.2.2.2 Kernel

EPOC has a very lightweight kernel designed to provide high performance, and to require the minimum amount of code to run in privileged mode.

8.2.2.2.1 Scheduler

The kernel supports pre-emptive multi-tasking using threads, which are the basic unit of execution. The scheduler runs the eligible thread with the highest priority. Thread priorities are not aged.

Processes are the basic unit of memory management: a process has its own address space, a primary thread, and any number of other threads. Context switching and inter-thread communication between threads in different processes is fast, and between threads in the same process is very fast.

8.2.2.2.2 Tick interrupt

A tick interrupt, every 15.625ms (64Hz) on ARM implementations, or 100ms (10Hz) on PC-based implementations, is used to drive timer queues and round-robin scheduling of the highest-priority threads. Some device drivers also use the tick interrupt, e.g. for keyboard and scanning and digitizer polling. User-side timer requests operate with a maximum resolution defined by the tick interrupt.

A microsecond timer is available for kernel-side use.

8.2.2.2.3 Memory management

EPOC supports a conventional two-level memory management unit (MMU). In other OSs, two-level MMUs use one page directory per process, so that process switching involves changing a single control register (and some cache flushing).

EPOC uses only a single page directory, with each process represented by as many PDEs (page directory entries) as are needed to hold the relevant page tables. This saves RAM, and is possible because there is no requirement for EPOC to support potentially very large virtual addressed spaces backed by swap files on disk.

Memory is allocated in chunks, which are consecutive in virtual address space. A thread typically uses a single chunk, with a stack at the bottom and a guard page table entry to detect stack overflow, and heap at the top. Such chunks may expand upwards, but not downwards. Chunks are typically private, but may be shared. A chunk requires at least one PDE and, as it expands, may require more PDEs.

Context switching on EPOC's ARM3 implementation involves moving all a process's PDEs from a "low" location, where the process runs, to a "high" location, where its data is accessible only by the kernel. EPOC R5 supports fixed processes whose PDEs do not move, so that switching between these processes, the kernel and one non-fixed user process is much faster. The practical limit to the number of fixed processes is four (a separate protection domain is used for each, and there are only four such domains). The fixed processes in EPOC R5 are the file server, serial comms server, window server, and font and bitmap server.

8.2.2.2.4 Executive and server

The kernel consists of an executive and a server. The executive is a privileged library running in the context of the calling thread (though with a separate stack). It handles kernel functions, which do not require resource allocation. The kernel server is the highest-priority thread in the system, and handles all requests, which do require kernel-side resource allocation.

8.2.2.2.5 Device drivers

The kernel manages calls from user code to device driver functions, which may run as either kernel-executive or kernel-server calls.
Device drivers are also driven by interrupts. Interrupt service routines (ISRs) are short, so as to minimize non-preemptible time. ISRs cannot call kernel services, but they may queue delayed function calls (DFCs). If the system was interrupted in user state, the kernel will run the DFC immediately after the ISR; otherwise, the DFC will be run when control would otherwise have returned to user privilege. DFCs are more powerful potentially than ISRs. Even so, many DFCs simply post a user thread to handle the event.

8.2.2.2.6 Power management

Power management is very important for a hand-portable machine. The user perceives the machine to be "on" when the screen is active. Internally, power handling is more sophisticated.
If user-thread processing is taking place, the CPU is powered up constantly; otherwise, the CPU is quiesced and only activated for brief processing in response to the tick interrupt. If the machine is off, its DRAM must still be refreshed, and it must be able to switch itself on in response to the ON key, the next alarm or incoming call etc. Devices such as comms links, phone lines, PC cards and other removable media have their own power management requirements.
The kernel and device drivers use a power model to keep track of power requirements and power sources, and close down devices and power sources when not required. Most power sources equate to one of the system clocks, eg the processor clock, and lower-speed clocks for peripherals, DRAM refresh, the tick interrupt etc. The kernel implements power management with a so-called "null thread", the lowest-priority thread in the system. When the null thread is scheduled, it simply informs the power model that the CPU is no longer needed: the power model then shuts down the CPU until the next interrupt.
Real electrical power comes from primary and backup batteries, as well as external power (from a mains-driven PSU). A non-maskable interrupt is used to process emergency power-down — i.e., removal of primary power sources.

8.2.2.2.7 Porting and layering

The kernel is structured in several layers to facilitate porting. Some device drivers also use similar layering.
A platform-independent layer handles communication with the executive, the basic kernel server framework, and much internal kernel processing. A platform layer defines the main differences between emulator/WINC implementations (in which services are implemented as calls to the Win32 APIs), and full EPOC implementations (in which EPOC drives real hardware).
Lower layers implement specifics for CPUs (eg ARM3, StrongARM, M*Core), ASSPs (application-specific standard parts, i.e. single chips comprising a CPU, MMU and various peripherals), and variants (more minor differences). An EPOC port from one variant to another is relatively easy; an ASSP port involves more work, and a CPU port significantly more.

8.2.2.2.8 Boot sequence

The kernel's boot sequence brings up the MMU, kernel server, null thread, all kernel devices and drivers, and finally the file server.
On target machines, the file server then loads the window server, which launches other necessary system processes. The base delivers a text window server, which can be used for testing low-level components, including the text shell. The full graphics window server launches a graphics shell.

On the emulator, the sequence is reversed: the window server is started first, but it then asks the Win32 version of the EPOC kernel to start. The full graphics window server is delivered as epoc.exe. Command-line test programs contain object code, which starts the kernel and text window server before proceeding with the main program.
On WINC, no EPOC window server is involved at all: command-line tools cause the kernel to start automatically, while GUI applications such as EPOC Connect must invoke the kernel start code explicitly.

8.2.2.3 File server

The F32 file server is released as a separate EPOC component but is in fact tied closely into the kernel.
Firstly this is because the file server provides the system loader: the kernel emulates the loader during boot but thereafter uses the file server for all loading.
Secondly the kernel provides support for the file server to allocate RAM for its c: disk: RAM-disk allocation is handled carefully so as to be able to recover data in a warm-boot situation. RAM-disk allocation is completely dynamic, with no user intervention.
The file server uses file system drivers to provide up to 26 disk drives on various devices. On a machine platform, the local file system maps ROM to drive z: and RAM to drive c:. Removable media are supported, on CF cards (compact flash, a small form-factor adaptation of the PC card standard).
Volumes are formatted using VFAT: filenames, including their drive and path, may be up to 256 characters long.
The file server implements the program loader that supports both exes and DLLs. Both are executed in-place from ROM, or loaded as needed from RAM or from removable drives. On machine platforms, DLLs are restricted to linking by ordinal, rather than by name: this prevents space being wasted by potentially long entry-point names. Writeable static data is not available for application program use: instead, object handles must be passed directly, or thread-local storage (TLS) must be used: there is one machine word of TLS per DLL per thread.
A distinctive aspect of the EPOC file system is the use of 32-bit UIDs (unique IDs), which allow the type of every executable (.exe or DLL) to be identified. This serves as a form of identification, used among other things for associating an application file with its owning application. It also protects against accidental loading of a DLL which is not the version or type required. UIDs are checked by the EPOC native loader: on the emulator and WINC, program loading is handled by Win32, so UIDs are not checked.

8.2.2.4 User library and file server APIs

The user library provides APIs and frameworks for higher-level programs. Some of these APIs are implemented purely in user-side code. Others call kernel functions, via the executive and, when required, the kernel server also.
The user library provides four distinctive APIs that set the flavour of any EPOC programming:

- cleanup stack and Leave() function which, along with some easily-learned programming disciplines, make it easy for programs to detect errors such as out-of-memory, and to clean up partially allocated resources. This framework is analogous to try/catch and throw in C++ or Java — but uses less memory.
- descriptors: a unified class hierarchy is used to support binary data buffers, strings of 8-bit characters, and strings of 16-bit characters.
- active objects: these provide an object-oriented event-handling framework which allows most multi-tasking programs to be structured effectively as event handlers running in a single thread. This is more CPU-efficient, memory-efficient, and programmer-friendly than conventional multi-threaded programming.
- client-server architecture: only the minimum of EPOC services are provided by the kernel or device drivers. The majority is provided by servers running in user mode. It is very easy to use servers, and quite easy to write them.

The user library also provides all the facilities you would expect of an operating system at this level. A conventional group of support functions for DLLs, threads and processes is provided. Data

structures include lists, dynamic memory buffers, and extensible arrays that build on these buffers. Good use is made of C++ templates to provide type-safe collection classes. Math functions manipulate IEEE754 double-precision floating-point numbers, and text functions include formatting akin to sprintf() in standard C, and a lexer akin to sscanf().
The file server API allows clients to manipulate drives, volumes, directories and files. Notification APIs are provided to detect changes in files and insertion/removal of removable media.
The user library and file server APIs are completely object oriented. The EPOC C Standard Library provides POSIX-like non-OO function wrappers *on top* of these: for instance, User::Alloc() may be accessed using malloc(), and RFile::Open() may be accessed through fopen().

8.2.2.5 Tools

The base delivers tools used for building EPOC. These are packaged into the EPOC C++ SDK, for general software development, and into the OEM Adaptation Kit (OAK), for building new ROM-based EPOC machines.
The EPOC emulator is a key tool for Windows-hosted development. Microsoft Visual C++ Version 5 or 6 are supported for building programs in the emulator or WINC environments. Microsoft Visual C++ must be bought separately from the EPOC SDK or OAK products.
For building programs for target machines, the GNU C++ Compiler gcc, delivered with the C++ SDK, is used, together with other tools for object and binary file processing. The petran tool translates gcc output into a modified PE (portable executable) format, suitable for target machine binaries and for building into ROMs.
EPOC presently supports two instruction sets (x86 and ARM3) and more are planned. Projects may also be built in release or debug variants, and in narrow (8-bit character) and wide (Unicode character) variants. C++ projects are specified using a simple EPOC project file format, and built into make files for the relevant toolchain by the makmake tool.
ROMs are built using the rombuild tool, driven by a so-called obey file whose extension is conventionally .oby. Dual-boot ROMs, supporting more than one hardware variant, are supported. Some EPOC machines, and all evaluation boards, use flash ROMs which can be reprogrammed wholesale, or a flash area which can be used for patching. EPOC OAKs, and OEMs, provide the appropriate tools.

8.2.3. Engine Support

Application engines provide the API to application data used both by application UIs, and by the converters and synchronizers found in EPOC Connect. The engine support APIs provide underlying support for application engines. Like the engines themselves, the engine support APIs consist mainly of data manipulation code, with no drawing or other user interaction.
Strictly, some graphics components are required by application engines — and even by some of the components listed here — and could also be included in the engine support category. Examples include the GDI, which defines the graphical interface including fonts etc used by the text content model, and the BITGDI, which renders graphics primitives to on- and off-screen bitmaps: bitmap processing, even without user interaction, is important for many application engines.
Engine support components may be categorized as data manipulation, application architecture, resource files and utilities, standard library and text handling.

8.3.3.1 Data manipulation

The F32 file server provides basic file services including the ability to read and write binary data to any position in a file. For application data formats, a higher level of utility is required. This is provided by two EPOC components: the stream store (STORE) and the database management system (DBMS).
The stream store provides a stream interface for externalizing and internalizing data. Stores consist of one or more streams. Three significant file stores are provided: a direct file store for load/save applications such as a word processor; a permanent file store for database applications; and a

dictionary file store used for initialization files. An embedded store provides direct file-store functionality and is used by the application architecture for object embedding.
The database management system provides a complete relational database implementation with API and SQL-based access. Each database may contain multiple tables, each with multiple columns supporting a wide variety of text, binary, numeric, date/time and autoincrement data types. The storage format is compact and robust. APIs and SQL statements support data definition and data manipulation. Transactions, encryption, indexes, space reclamation, rollback, and efficient views are supported. Databases may be shared between multiple concurrent clients, with locking and notification APIs.

8.2.3.2 Application Architecture

The application architecture (APPARC) provides the means to identify the program that should be used to open a file, an embedded object, an attachment, or some data that has not yet been written to a file (such as data being downloaded from the worldwide web). The application architecture supports both EPOC's native file formats (which use EPOC stream stores), and foreign file formats.
Native document file formats are recognized entirely by their UID: they do not require an extension. Foreign files may be recognized using a combination of content and filename/extension: a MIME type is then associated with the file and used to select associated applications.
The application architecture provides basic user interface elements for file launching. Icons and natural-language captions are contained — along with supported MIME types — in an application information file (AIF), which is generated by aiftool.
EPOC's native file format supports embedding. An application's embedding capabilities are specified in its AIF. Embedded documents are represented by a "door", which may be either an iconic door displaying the associated application icon, or a glass door which displays the application data before it is opened for editing by the user.
The application architecture also supports shell extensions (control panel icons), display of all running applications and open files, and a most-recently used file list for all document files.
The application architecture defines conventions for the locations of applications, control panel icons, file recognizers etc. For instance, applications must reside on some drive with a file and directory name of the form \system\apps\appname\appname.app. APIs are provided to interrogate the currently installed applications and recognizers, etc, and the currently running applications, currently open files, most recently used files etc. The application architecture provides a server, which caches lists of these objects, and monitors the file system and window server so that it can update these lists only when necessary. The enquiry APIs use the APPARC server's cached information and therefore execute quickly, without searching through the file system.
EPOC applications can be installed simply by copying files into their target locations. No system registry settings, with all their inconvenience, are required. When a CF card with applications installed on it is inserted into an EPOC machine, its applications are available for instant use. The APPARC server ensures that this high usability is delivered with high performance.

8.2.3.3 Resource files and utilities

Programs should be written in a manner, which does not confuse program code (in C++, say), with strings (which may need to be translated into different natural languages). This is achieved using resource files, built by the rcomp resource compiler, and read by programs using the facilities of the BAFL component (basic application framework library).
BAFL provides other utility APIs, for playing and recording sounds, some specific arrays, command-line parsing, and incremental matching. Other utility APIs are provided by EALWL (the alarm and world server) and DIAL, which resolves phone numbers against the current home location, and handles DTMF dialling.
WLDTOOLS and WLDDATA components are of interest only to OEMs: they provide the tools and data needed to prepare world country and city information for EALWL.

8.2.3.4 C Standard library

All EPOC's native code is written in C++. Symbian has achieved the benefits of object orientation that are often touted, but rarely achieved in practice: better-controlled designs, substantial code

re-use leading to smaller ROMs and less re-testing, and high performance from a C++ implementation.
The EPOC C Standard Library provides a layer *on top* of many of EPOC's native services, supporting a standard POSIX-like set of APIs, with which many programmers are already familiar. The standard library covers

- the most fundamental facilities provided by the user library: such as memory allocation (malloc() etc) and string functions (strxxx() and memxxx() etc)
- file i/o (FILE, fopen() etc)
- sockets and TCP/IP, including POSIX's unified file and sockets API (open() etc)
- processes, pipes, and blocking i/o

The standard library allows engine code from POSIX-type environments to be ported to EPOC with relative ease. The user interface (whether text- or graphics-based) must be coded in native EPOC C++.

8.2.3.5 Text

CHARCONV's APIs provide conversion functions between various character sets and are powerful enough for all Unicode requirements.
LEXICON provides spell-checking functionality for alphabetic languages. LEXICON implements the spell check algorithms in a server containing Lernout & Hauspie's International CorrectSpell™ technology, which provides a compact 100,000-word dictionary.
ETEXT is a powerful component providing manipulation of arbitrarily long text objects, with character and paragraph formatting, styles, headers and footers, fields such as page numbers, embedded pictures, and support for externalizing and internalizing to streams and stores. ETEXT text objects are used by higher-level EPOC components such as the database and word processor applications.

8.2.4. Graphic

EPOC's graphics components provide the basis for a higher-level GUI framework for applications and the system shell.
In EPOC R5, EIKON remains the only GUI framework available to developers in EPOC SDKs. EPOC has been designed to support device families with widely differing hardware specifications and UI requirements: different device families will include selected graphics components as described in this section. However, they will replace EIKON, the system shell and the application user interfaces with versions tailored to the specifics of the design.
Common graphics components include the central drawing and user interaction components, the font system, the print system, and views.

8.2.4.1 Drawing and user interaction

As the foundation for all other graphics components, the GDI defines graphics devices (to which all drawing is done), graphics contexts (through which all drawing is done), a print subsystem (supporting banded printing, on-screen preview, and model selection), and a range of ancillary classes supporting fonts, colour, measurement and zooming.
For user interaction, the window server (WSERV) shares the screen, keyboard, pointer and other UI resources, and provides the basics of an event-driven interaction mechanism for client programs. The window server thus mediates almost all user control of the EPOC machine. The window server is started by F32 at the end of its boot sequence. It launches the font and bitmap server while initializing. After all other initialization is complete, it starts the system shell, a GUI application which gives the user real access to the machine, its data and applications.
CONE, the control environment, provides a layer on top of the window server API for building higher-level application GUI frameworks. A control is a rectangular area of the screen and is the fundamental unit of user interaction. Controls may contain other controls. CONE provides an abstraction for keyboard focus, and an algorithm for offering keys to controls using a control stack. CONE provides an active-object-based framework for client applications. Window server events are converted into virtual function calls in control, environment or control stack classes: GUI and application developers implement these functions to provide required behaviour.

Between the GDI and the window server are two highly optimized components for implementing bitmapped graphics operations: the BITGDI renders drawing primitives onto bitmaps — either on- or off-screen, and the font and bitmap server (FBSERV) provides shared-memory implementations of large graphics elements — fonts and off-screen bitmaps. For efficiency, parts of the BITGDI are coded in assembler on ARM implementations — almost the only such code outside the base.

The font and bitmap server uses an EPOC bitmap format supporting multiple bitmaps per file, and optimized implementations in ROM. A tool, bmconv, is provided in EPOC SDKs which converts between EPOC and Windows bitmap formats.

EPOC's graphics components implement broadly similar functionality to those of Microsoft Windows. EPOC's use of twips as the basic measurement unit, combined with the specification of its basic drawing primitives, allow for easy conversion of EPOC graphics into Windows graphics. This makes it relatively easy to implement a driver for PC-based windows-hosted printers, with the PC connected to the EPOC machine by cable or infrared.

On the other hand, the EPOC model is fundamentally different and employs different terminology from the Windows model. The EPOC GDI specifies graphics behaviour; the Windows GDI is more like the EPOC GDI, FBSERV and BITGDI components combined. The EPOC window server and CONE and higher-level GUI such as EIKON act like the Windows graphics, windowing and event systems, plus MFC, all rolled into one: but the EPOC structure is object-oriented from the ground up, and EIKON does not merely add object orientation to existing graphics functionality: rather, it adds real look-and-feel to a lower-level, object oriented, graphics framework.

8.2.4.2 Fonts

The GDI architecture allows the list of available fonts to be queried and, for each font, a list of point sizes to be generated. The architecture requires fonts to be specified with a name and a size (in twips, one-twentieth of a point), and then finds the nearest matching font.

The font store (FNTSTORE) keeps track of all fonts available in EPOC. It can store both bitmapped and vector fonts. New fonts may be added to an EPOC machine at any time, without any need to re-boot.

Standard EPOC implementations release Arial, Times New Roman and Courier fonts in a variety of sizes, in the CP1252 code page. The Euro symbol is included at code point 0x80. Special fonts are also provided, eg symbols for the EIKON GUI and calculator application.

8.2.4.3 Printing

Printing was designed into the GDI from the beginning, with support for any number of printer drivers. The PRINT component handles printing, using the PDRSTORE printer driver manager. New drivers may be installed without rebooting. A selection of common printers is supported, together with a catch-all option which drives any printer via a PC (the PC must be running EPOC Connect), and printing to fax.

On-screen print preview is supported. Printing uses a banded print model which can be supported by application engines with relatively little modification to their drawing code.

8.2.4.4 Views

EPOC's graphics components contain powerful, re-usable, application-level views:

- FORM displays formatted text, honouring all the character, paragraph and style attributes supported by ETEXT, and with highly optimized display refresh and editing code. EPOC Word and many other applications use FORM.
- GRID displays data in a spreadsheet-like grid. GRID is used by the EPOC Sheet and EPOC Data applications.
- CLOCK displays animated clocks, either digital or analogue. CLOCK is used by the EPOC Time application, and by all EIKON toolbars.
- CHART is used by EPOC Sheet to draw data in popular business presentation formats.

8.2.4.5 Text entry

EPOC was originally built for keyboard-oriented text input. A front-end-processor (FEP) framework to text entry controls has been introduced, which supports handwriting recognition and entry of Far Eastern characters by conventional handwriting or keyboard entry methods. The FEPBASE component provides the basic APIs, and a BRDCST server provides broadcast services required to support FEP-related interactions.

8.2.5. GUI and System

EPOC's GUI and system components provide the environment and programming framework for applications, and define the look-and-feel of an EPOC machine for the end-user.
EIKON is a GUI framework providing standard controls and dialogs for programmer use, a dialog framework, an application and object embedding framework, and many utilities. EIKON builds on CONE, the control environment from EPOC's graphics layer, and APPARC, the application and data management component from EPOC's engine support layer.
The SHELL application builds on EIKON and is on the one hand a file/application browser/launcher, on the other hand a control panel which enables all the machine's hardware and software settings to be displayed and, where appropriate, changed. The control panel includes a software installer/uninstaller.
New applications and control panel extensions may be written using the EIKON and APPARC frameworks.
The EPOC help system displays help for the EPOC system and its applications. It provides convenient browsing and searching. The aleppo tool builds compressed help databases from RTF source documents.

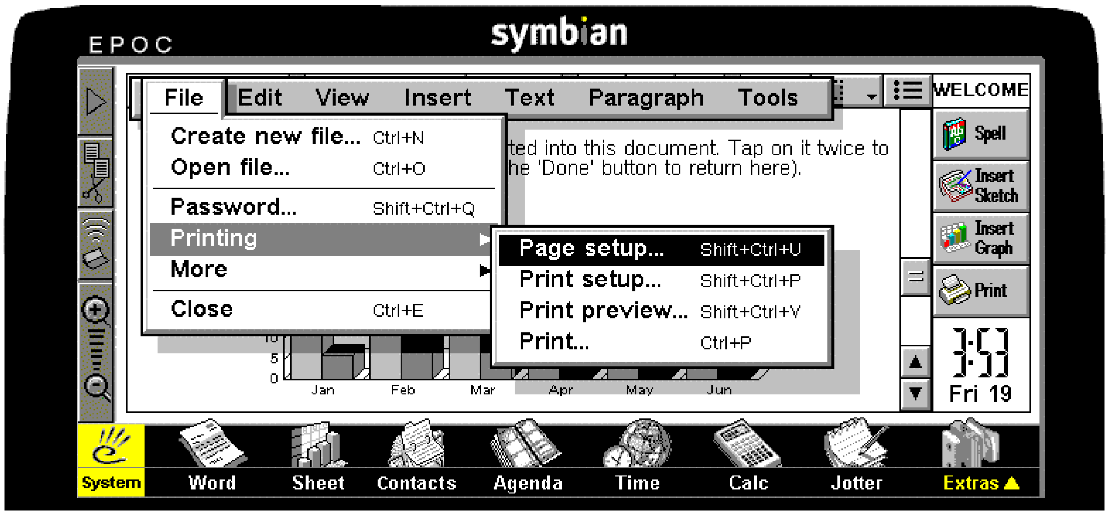
8.2.5.1 Device families

EIKON is designed for colour or greyscale displays, alphanumeric keyboard input, pen input via screen digitizer with extensions for sidebar and taskbar, and screen sizes of 640 wide by 240 or more high.
Slight variations on this specification are feasible (for instance, trackpad or mouse rather than pen, or alternative arrangements for the sidebar and task bar). Such variations would require only minor modifications to EIKON and the SHELL, and none to the applications.
However, EPOC has been designed to support devices with significantly different characteristics from those for which EIKON was envisaged — smaller screens, no pointer at all, different button arrangements, numeric pad or no keyboard at all, sound-based interaction mechanisms, and wireless information devices which look much more like a phone than a computer. Symbian and its partners are working on the specification of a small number of *device families*, for which it will issue reference designs for convenient adoption by EPOC OEMs and software developers.
Each reference design will be based on a GUI and shell tailored to its own requirements: these will differ from EIKON — in some cases, quite significantly. To obtain maximum re-use when implementing EPOC and its applications on a new device family, EPOC has been designed to allow EIKON, the shell and everything that depends on them to be replaced, with minimal impact on either EPOC's lower-level components, or the applications themselves. This is a key reason why Symbian practices — and recommends — the division of applications into two components, an engine and a GUI.

8.2.5.2 EIKON

EIKON presents a GUI designed specifically for its intended target hardware, rather than being scaled down from a GUI suitable for desktop PCs.
The screen real estate is used carefully. Menu bars are only displayed when necessary. Toolbars may be toggled on and off. Application data may be zoomed to suit lighting conditions, eyesight and the type of data. Greyscale or colour screens are supported: the colour scheme is modest and professional in appearance. OEMs may conveniently modify the colour scheme, but not by end-users.
A high-quality keyboard is assumed: EIKON allows novice users to use it easily, and provides shortcuts for expert users.

The pen is different from a mouse: it can draw very easily, it can select and point just as easily as a mouse, but it cannot hover or right-click, and double-click and dragging are error-prone. This led to EIKON's select-and-open paradigm, whereby a pen tap on an unselected item selects it, while a tap on a selected item opens it — regardless of when the item was selected or even whether it was selected by pen.
EIKON supports extensions of the touch-sensitive screen area to the left, and below, the LCD display. Icons printed on the display surround, in the touch-sensitive area, then serve as buttons. The left-hand sidebar is used to select menu, clipboard, infrared beaming and zooming functions. The bottom taskbar is used to select applications.
The figure, EIKON shows an EIKON screenshot. The look-and-feel is rich with pleasant features. Menus, scrollbars, buttons and other elements are drawn neatly and look good on all types of screen. Shadows are used to highlight menus, dialog boxes, calendar potentially pop-ups etc. Operation of all applications is convenient with either the keyboard or the pen.


EPOC
symbian
File Edit View Insert Text Paragraph Tools
WELCOME
Create new file... Ctrl+N
Open file... Ctrl+O
Password... Shift+Ctrl+Q
Printing
More
Close Ctrl+E
Page setup... Shift+Ctrl+U
Print setup... Shift+Ctrl+P
Print preview... Shift+Ctrl+V
Print... Ctrl+P
Spell
Insert Sketch
Insert Graph
Print
3:53
Fri 19
Jan Feb Mar Apr May Jun
System Word Sheet Contacts Agenda Time Calc Jotter Extras

ted into this document. Tap on it twice to
he 'Done' button to return here).

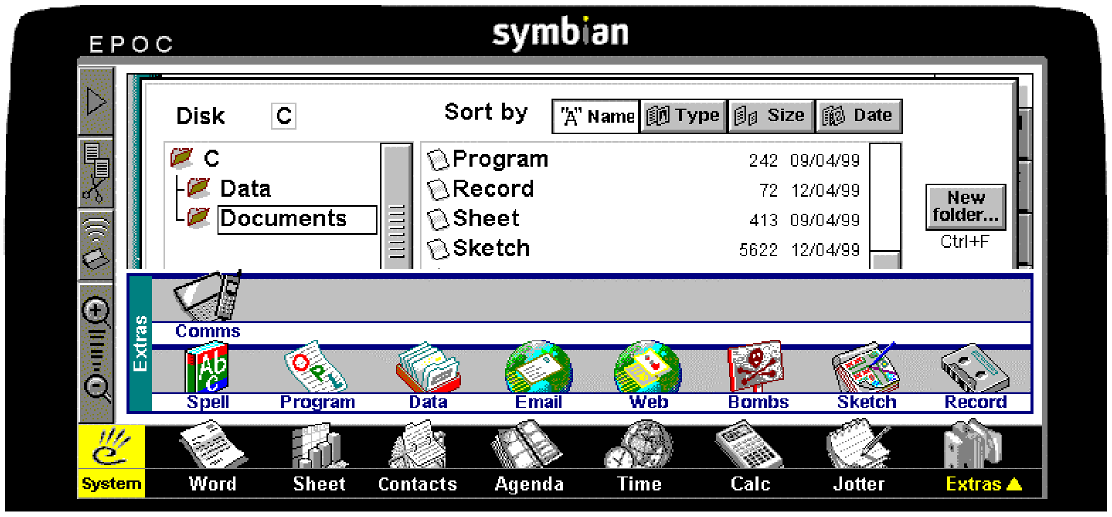
EIKON

Users can send commands to applications using menus, shortcut keys, or toolbar buttons. Menus support cascades and pop-ups. Dialog support includes single and multi-page dialogs and even scrolling dialogs for special applications.
Many standard controls are available, including number and text editors, list boxes and combo boxes, font and sound selectors, colour or greyscale selectors, and controls optimized for personal information management, such as convenient time/date navigation, and latitude and longitude editors.
Standard higher-level components include file browsers, Open and Save As dialogs, print settings and print preview. Scroll bars have dynamically-sized grab handles, and scroll application data while being dragged. A console is available for displaying comms terminals and text-based shells.
Many types of list are supported, including hierarchical lists, multi-column lists in which each entry has several aligned fields, and snaking lists, which wrap vertically into a container. A variety of standard lists are provided, and owner-draw lists may be written.
All these features are made conveniently accessible through APIs and resource files. The *EIKON Style Guide* in the SDK describes usage guidelines for producing EIKON applications, which conform with Symbian's look-and-feel standards for EIKON.
Each EIKON application runs in its own process, whose main thread has an instance of the EIKON environment, which controls all interaction on the screen and sidebar, and all keyboard interaction except a small number of hotkeys. Due to the effectiveness of the active object paradigm, secondary threads are rarely if ever used by native EPOC applications

EIKON also provides a single server, which runs in its own thread. The EIKON server is used to take control of the screen for higher-priority purposes than any normal application. The EIKON server uses the window server's hotkey feature to capture certain key combinations regardless of which application is running. The EIKON server also monitors the taskbar below the screen for application-switching buttons. The EIKON server is responsible for presenting alarm and power-on-password dialogs to the user, for handling screen capture (in response to shift+ctrl+alt+S), for launching the system help application (in response to the help key), for displaying error messages from the kernel (eg for unexpected death in some application or server thread), for running code to close down and restart all applications before and after system backup, and for application-switching interactions with the shell.

8.2.5.3 The shell

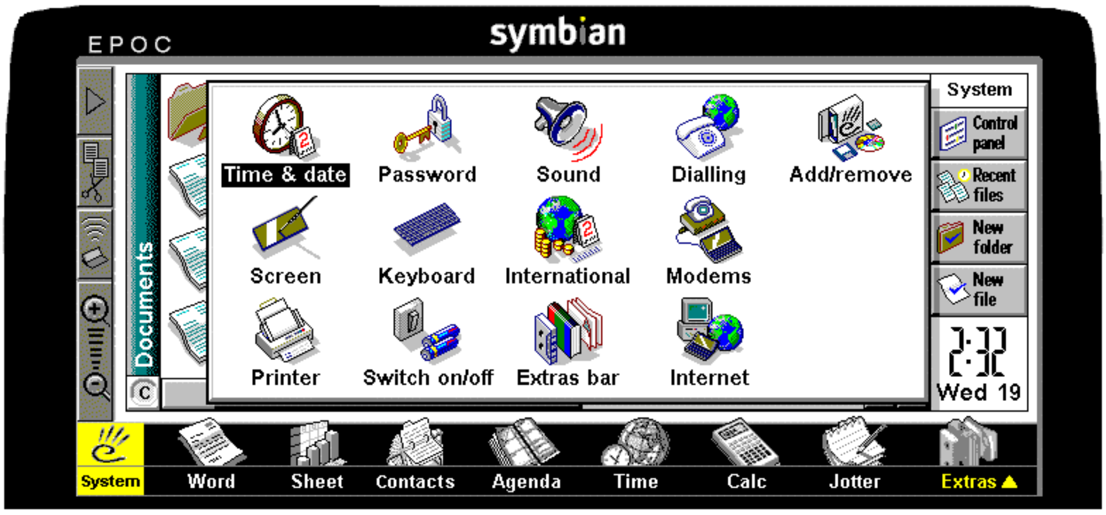
The system shell provides a file browser, which may be used to launch applications. The shell also controls the application launching/switching buttons on the taskbar (via the EIKON server), and the Extras bar which slides up onto the screen and acts like an extension to the fixed buttons.



Program	242	09/04/99
Record	72	12/04/99
Sheet	413	09/04/99
Sketch	5622	12/04/99

The shell showing browser

Instant display of system settings such as memory and disk usage, battery state and owner information is provided by menu items or hotkeys from the System application.



The shell showing control panel

A control panel provides access to settings for time/date, sound, screen, power, keyboard, printer, modems, dialling, Internet service provider, and locale including home city. The control panel also allows the extras bar to be configured, and provides access to an application installer/uninstaller for third-party software.
Application installation is very simple: as noted in earlier, the application architecture dictates file location conventions and simply scans the file systems to find installed applications. Applications may be instantly installed by copying their files into the right locations, or by inserting a CF card with pre-installed applications on it. The application installer allows application developers to deliver a single file that is expanded into multiple files in the correct directories, on any selected drive. The package file's extension is .sis: EPOC SDKs contain a makesis tool to build the packages, and .sis files may also be installed directly from a PC using EPOC Connect. After installation, a stripped-down version of the .sis file contains information used by the uninstaller. Versioning and dependency information are also supported.
The graphics window server starts the shell, which itself is started immediately after F32 has successfully booted. The shell starts the EIKON server, which monitors the task-switching keys. In turn, the EIKON server starts the APPARC server, which monitors the file system and caches information about installed applications, running applications, open application files, and most recently used application files. For efficiency, the shell runs the EIKON server and APPARC server threads in the same process as the shell itself. For reliable multi-tasking even when applications are busy, the shell threads run at higher priority than standard applications.

8.2.5.4 System help

Help for standard applications and the shell (referred to as the System application) is provided in a single compressed database, which is launched (or switched to) using the help key at any time. The database may be conveniently searched or browsed.
Add-in applications may provide their own help, by using a menu item to launch the help browser with their specific help database. The aleppo tool, supplied with EPOC SDKs, creates help databases from Microsoft Word RTF files. An SGML format used in the aleppo processing chain is also documented, and may be used for creating help databases using other editors.

8.3 Applications

EPOC is a whole operating system encompassing a base, graphics, applications, Java runtime system, wireless communications protocols and applications, and many other features. This chapter describes the application suite in terms of features offered to the user, and how the applications are implemented using EPOC's underlying technology.
This chapter is part of the overview series for EPOC R5, covering core components, communications, software development options including the Java environment, applications, and connectivity to PCs and PC-based applications.

8.3.1. Introduction

EPOC Release 5 includes a full application suite of office, PIM, messaging, utility, software development, system and connectivity applications, optimized for EPOC machines' portability, wireless communications, and synchronization with PC-based data.
Applications delivered by Symbian are:

Office	Word processor, spreadsheet, database, spell checker, Jotter
PIM	Contacts manager, Agenda, world map and alarm clock
Communications	Email, fax and SMS, web browser, serial comms
Utilities	Calculator, Sketch, Voice Notes
System	File/apps browser, control panel with settings for time/date, sound, screen, power, keyboard, printer, modems, dialling, internet service provider, and locale; user information including password, extras bar configuration tool, and program installer/uninstaller
Development	OPL program editor/translator
Game	Bombs

Connectivity	PC-based EPOC Connect for data synchronization, file management, printing via PC, application installation from PC, and other utility functions

OEMs may include additional applications with their EPOC products.

8.3.2. User perspective

Common application features include

- user interface designed specifically for the usability requirements of hand-portable computers and communicators
- data formats designed for efficient memory use, for synchronization with PC applications, and exchange of data with other industry-standard systems
- easy document embedding: for instance, word processor memos may be attached to agenda items to provide further detail, or spreadsheets or sketches can be embedded in word processor documents
- transfer to other applications on the same machine via clipboard, to PCs via CopyAnyWhere inter-machine clipboard transfer, and to other EPOC machines via infrared
- printing to directly connected printer, to printer via PC, or on-screen preview
- instant zooming to optimize for clarity of display or quantity of information, depending on lighting conditions and user preferences
- easy-to-use help
- instant task switching between concurrently running applications
- a level of user customisability appropriate to hand-portable computers and communicators
- robustness and speed

8.3.2.1 Office

EPOC delivers a rich suite of office applications, compatible with PC-based equivalents from Microsoft, Lotus and Corel, which can be converted into and from EPOC formats using EPOC Connect. EPOC's office applications are the best in class for hand-portable computers, with a greater feature set, better performance and lower memory requirements than those of peer systems.

Word	Best-in-class word processor supporting rich character and paragraph formatting, named styles, contents outline, templates, integrated spell check, printing including preview, embedded objects (chart, sheet, sketch etc).
Sheet	Spreadsheet with rich text, borders and shading, many general, scientific, financial and statistical functions, grid view and chart view.
Data	Single-table database built on EPOC's powerful database engine. Provides convenient search, data entry, column definition and adjustment.
Spell	Spell checker with 100,000-word dictionary, anagram, crossword and thesaurus functions.
Jotter	Quick-and-easy notepad for instant jotting when you don't have the time to think about where to put things.

8.3.2.2 Personal information management

Like its office applications, EPOC's PIM applications are compatible with PC-based equivalents. EPOC Connect supports synchronization with PC-based applications, without closing down the EPOC-based applications, thanks to new EPOC PIM server architectures. Like the office applications, the PIM applications are best in class for hand-portable computers.

Contacts	Contacts database with standard fields for home, office and mobile details, plus user-definable extras. Integrates with Email application for e-mails, faxes and SMSs. Contacts may be exchanged over infrared with other machines, e.g. Palm III or Nokia 9110, using the industry-standard vCard 2.5 format.

Agenda	Powerful schedule application with day, week, year, busy and anniversary views, to-do lists, appointments, whole-day and multi-day events, powerful and convenient repeats, alarms, tentative entries and entries which are private when synchronized with other schedules. Agenda entries may be exchanged over infrared with other machines, eg Palm III, using the industry-standard vCalendar 1.1 format.
Time	World map with information about over 700 cities, alarm clock with up to eight one-off or repeating alarms, functions for setting home city and date/time and for adding cities.

You can run your life from EPOC's PIM applications!

8.3.2.3 Communications

EPOC supports internet-standard e-mail and web browsing, plus fax and SMS messaging, and old-style serial communications.

Email	Internet-standard e-mail, fax and SMS, integrated with the Contacts manager application. Supports multiple internet service providers (ISPs), user-defined folders, and multiple protocols. Microsoft Word attachments may be viewed — without risk of virus infection. SMS is integrated with GSM phones. Fax is integrated with printing. Individual add-ins (such as forthcoming IMAP4 support) can be easily installed.
Web	Web browser to HTML 3.2 specification including frames, Java applet support, and a shared persistent cache for minimizing online time.
Comms	Old-style comms application supporting terminal emulator access to dial-up services. TTY and VT100 emulations are supported, with automated login scripting, and XMODEM and YMODEM file transfers.

8.3.2.4 Utilities

Utilities include a calculator with desktop and scientific modes, sketch program and voice-notes sound recorder/playback. The sketch program includes more than 70 clip art drawings, and makes full use of colour on devices, which support it. Sound recordings can be compressed. Sketches and recordings may be embedded in other documents. Recordings may be used as alarm sounds. Sketches may be used as wallpaper.

8.3.2.5 System

The system application provides a file browser that may be used to launch applications. System also controls the application launching/switching buttons at the bottom of the screen, and the Extras bar, which slides up onto the screen and acts like an extension to the fixed buttons.
Instant display of system settings such as memory and disk usage, battery state and owner information is provided by menu items or hotkeys from the System application.
A control panel provides access to settings for time/date, password and owner information, sound, screen, power, keyboard, printer, modems, dialling, Internet service provider, and locale including home city. The control panel also allows the extras bar to be configured, and provides access to an application installer/uninstaller for third-party software.

8.3.2.6 Development

EPOC's OPL language is a BASIC-like rapid application development language. EPOC provides a full programming environment, on the EPOC device, which you can use to edit, translate and run OPL programs, without requiring a PC-based SDK. A strong developer community has grown up around OPL application development.

8.3.2.7 Game

For whiling away those spare minutes, EPOC includes Bombs, a game in the minesweeper genre. It is written entirely in OPL and can be reconfigured by the user.

8.3.2.8 Connectivity

EPOC machines are fully capable for office applications, personal information management and communication using e-mail, fax and SMS. You can print directly from an EPOC machine, can exchange data with other EPOC machines — and standards-compliant handhelds — using infrared, and can install applications downloaded from websites, without using a PC.
For many purposes, however, PC connectivity and synchronization with PC- or corporate-based data is important. PCs also provide a convenient backup medium. EPOC Connect integrates with Windows Explorer and to provide these and other functions:

file management and conversion	Using Windows Explorer, you can drag and drop files between directories on either the PC or the EPOC machine. When dragging from PC to EPOC or vice versa, EPOC Connect automatically converts the file format. If you use Windows Explorer to open a file on the EPOC machine, it will be automatically uploaded to the PC, converted into a PC application format, and opened in the relevant PC application.
synchronization	You can synchronize your agenda and contacts whenever you want. You can optionally set EPOC Connect to synchronize automatically every time you connect your EPOC machine to a PC — or once a day, or once a week.
backup	You can backup your data whenever you want. You can optionally set EPOC Connect to backup automatically every time you connect your EPOC machine to a PC — or once a day, or once a week. Multiple versions of changed files are retained. Restore is rarely needed, but is simple and effective.
clipboard transfer	You don't have to exchange data using whole files: when your PC and EPOC machines are connected, you can use EPOC Connect to keep their clipboards synchronized too, aided by the CopyAnyWhere tool.
multiple machine management	EPOC Connect recognizes an EPOC machine's ID and can conveniently manage several EPOC machines from a single PC.
other handheld machine support	As well as EPOC machines, EPOC Connect also supports Psion's Series 3 (or SIBO) architecture and data formats.
application installation	Double-click on a `.sis` file from the Windows Explorer, and it will be installed onto your EPOC machine; these files are a way to deliver pre-packaged programs for EPOC devices.
printing	Select "Print via PC" as the printer driver on the EPOC machine, connect to a PC, and you can use EPOC Connect to drive any PC-based printer from your EPOC machine.

8.3.2.9 Expandability

As well as EPOC's built-in applications, EPOC OEMs and third-party developers are developing applications in C++, Java and OPL, which you can conveniently install on your EPOC machine. To learn more about EPOC software development kits (SDKs), see Symbian's developer support website at *developer.epocworld.com*.
Symbian also provides an SDK for EPOC Connect, which can be used to develop PC-based applications involving data interchange or control of EPOC machines.

8.3.3. Technology

It is interesting to understand the technology that goes into EPOC applications. There are two motives for this:

- the re-use of technology is fundamental to EPOC applications' power: in a relatively small machine, you can only deliver powerful applications by consistently re-using powerful underlying APIs.
- understanding the technology used in existing applications can be inspiring for creating new applications, or vital for writing software which interfaces with these applications

In this section, we describe how the standard EPOC applications use EPOC's underlying technology. Other papers in this series go into more detail on the technologies mentioned.

8.3.3.1 Power through re-use

EPOC applications gain significant power through re-use of EPOC's technology base. Two outstanding examples of this are

- the word processor is a very thin UI layer over EPOC's ETEXT (text content) and FORM (text views) components, which together provide rich text functionality. These same components are used widely throughout other EPOC applications, eg for database content, agenda entries or owner information. From the user point of view, this means that all rich text fields may contain anything that can be composed in the word processor.
- EPOC's native document format supports embedding, so that any embeddable document type may be embedded in anything that can embed. Rich text can embed. Thus EPOC applications easily support attaching Word memos to agenda items, embedding pictures or spreadsheets in Word, etc.

A classic mistake arising from re-use is to use the same UI for quite different purposes, when using the same API. EPOC avoids this mistake by tailoring the UI to the most likely usage patterns. For instance, in the case of rich text in the Word and Agenda applications,

- Word has a full UI with toolbar and toolband allowing control of many character and paragraph properties, insertion of embedded objects, etc
- most of the time, Agenda users enter only a simple text string. It would be quite unnerving to present the Agenda user with the full Word UI for editing entries. So users initially see a simple text entry field; but with a couple of taps they can get a rich text editor for the field if they wish to.
- in fact, for Agenda entries, the special case of attaching a memo is so common that a single keystroke causes the system to create or open an embedded Word document at the end of the entry rich text.

Some other examples of re-use:

- all applications use the EIKON GUI, which was designed specifically for the usability requirements of hand-portable computers
- the stream store provides compact storage of application data
- the clipboard supports transfer to other applications on the same machine
- in conjunction with the PLP server architecture, the clipboard supports transfer between EPOC machines and PCs, using CopyAnyWhere
- the pervasive use of rich text ensures that clipboard transfers maintain formatting faithfully rather than, as on many PC-based systems, introducing annoying formatting anomalies
- the print subsystem supports printing, print setup, page setup, print preview, with convenience from any application
- the GDI's device map class supports zooming to optimize for clarity of display or quantity of information, depending on lighting conditions and user preferences

Re-use enables Symbian to deliver very powerful, compact ROMs. EPOC R5, with all its applications and Java runtime, amounts to only a little over 12M. This is a significant figure for OEM manufacturing costs.

Finally, re-use contributes to robustness and speed. EPOC's code is well tested, and optimized where necessary for high performance. This optimization need only be done once in a highly re-used component, and the benefits are realised throughout the system.

8.3.3.2 Implementation patterns

In addition to general re-use, EPOC applications use a number of characteristic design patterns.

8.3.3.2.1 Engine/GUI split

A design goal of all EPOC applications is to allow the application to be ported to a non-EIKON GUI without significant change. All major applications (Word, Sheet, Data, Jotter, Contacts, Agenda, Email, Web, and Comms) are explicitly divided into an engine and a GUI. The GUI can be replaced without modifying the engine. The few applications which are not structured in this way are either so simple that it isn't worth doing or, as in the case of Spell or Time, build on

lower-level EPOC components of significant complexity, which effectively function as their engines.
Another benefit of splitting applications into engine and GUI components is that engines can be designed for optimum performance, data formats, APIs and robustness, and tested aggressively, without having to worry about GUI code or UI design considerations.

8.3.3.2.2 Object embedding

EPOC's application architecture is used to support straightforward but effective document embedding, for file-based applications that use direct file stores for their main document. Such applications can very easily be enhanced to support embedding.
File-based applications using the permanent file store format may not be embedded. Applications, which are not file-based — i.e., which don't work by editing a main document, even though they may use files in other ways — cannot be embedded.
An embedded document is accessed from the embedding document by means of a *door*. A door may be iconic (the icon of the embedded document's application), or glass (the application data displayed before the door is opened).
Word, Sheet, Sketch, and Record documents may be embedded. Sheet and Sketch use glass doors. Data, Jotter, Agenda and Contacts use permanent file stores, and may not be embedded.
EPOC's other applications are not file-based, and so cannot be embedded.
Any application that supports rich text may embed documents, unless explicitly disabled from doing so. EPOC Email prevents EPOC-style embedding for Internet e-mails, because standard e-mail protocols do not support such embedding. However, internet e-mails do support sending and receiving attachments such as Microsoft Word documents, and fax messages composed using EPOC Email may contain embedded documents with glass doors used to render their content on the fax.

8.3.3.2.3 PIM servers

The Agenda and Contacts applications serve to enter and display information that is potentially of use to many applications. But these applications are only one client for the PIM data associated with them.
PIM information is not owned by the clients, but by dedicated servers. The real owner of the Agenda data is an Agenda server; the real owner of the Contacts data is the Contacts server. These servers may be accessed not only by their associated applications, but also by

- EPOC Connect, for synchronization without closing down the applications
- E-mail, for finding the e-mail address of a named contact
- bespoke applications, which may need to query or add contacts or appointments for various purposes

The DBMS also delivers a server architecture, so that databases may be shared between multiple clients. Shared databases are used by several EPOC systems and servers. The EPOC Data application, however, does not use shared databases.

8.3.3.3 Application technology summaries

This section summarizes the technology used in the more important EPOC applications. The technology used to implement the System application (shell) is described in *EPOC Overview: Core*.

8.3.3.3.1 Office

Word	ETEXT and FORM provide the re-usable code base for Word. A small engine, WPENG, serves essentially as a container for a rich text object. The LEXICON API is used for spell checking. WORD provides an application GUI optimized for ease-of-use for intensive word processor users.
Sheet	SHENG is the sheet engine. GRID and CHART are used for data display. SHEET is the application GUI. Sheet cells use globally formatted rich text, meaning that character and paragraph format attributes apply to an entire cell rather than individual characters. Embedded objects are not supported.
Data	DBMS is the underlying data engine. DBMS supports multiple tables and shared access, but Data constrains the database to include only a single table and owns the file exclusively while it is open,

	preventing sharing. These simplifications deliver a useful application with a well-controlled user interface. DAMODEL is the engine, adding various settings and utilities to the DBMS data. Columns may be rich text and hence include pictures and other embedded objects. Data may be viewed a card at a time, in a grid, re-using the GRID component.
Spell	An application GUI built on the LEXICON engine.
Jotter	A modified form of the Data application.

8.3.3.3.2 PIM

Contacts	CNTMODEL provides the contacts engine and server which in turn is a client of a shared DBMS-managed database. CONTACUI provides re-usable GUI code, used by both the Contacts app and contacts-using clients such as Email. Contacts can be selected conveniently through CONTACUI, and new contacts may be added. CONTACTS is the EIKON application.
Agenda	AGNMODEL provides the engine and server. AGNVIEW is the application. FORM is used for rich text entries including embedded memos, or hand-written/drawn sketch entries. The alarm server EALWL is used for alarms associated with agenda items. AGNMODEL implements agendas using a permanent file store — not a DBMS database. Agenda entries use a very compact format.
Time	The alarm/world server, EALWL, is used for both alarm clock functions, and world country, city and timezone information. EALWL is effectively the engine for TIME, the EIKON application.

8.3.3.3.3 Communications

Email	A server is used to control all message store operations. Multiple clients may be simultaneously active. Incoming messages may be added to the store while client applications are running. Internet e-mails use globally formatted rich text. Attachments are stored as binary files separately from e-mails. The application architecture is used to launch converters/viewers for Microsoft Word attachments: macros, with their potential for viruses, are not supported. Native EPOC documents may also be attached and viewed. Other converters/viewers could be implemented to view other document types. Fax supports rich text and embedded documents, with content rendered using glass doors. A system of polymorphic DLLs of several types, known as message transfer modules (MTMs) is used to allow replaceable GUIs (other than EIKON), new message types (e.g. Usenet news), and new low-level protocols (eg IMAP4). New MTMs may be installed at any time to augment the Email application as delivered on an EPOC R5 machine. Email uses TCP/IP and other protocols provided by the sockets server ESOCK, and integrates directly with the telephony server ETEL to pick up incoming SMSs, and to send and receive faxes. Email integrates with dial-up networking and maintains a connection status display.
Web	An underlying web technology library (WebTL), supplied by STNC Enterprises Ltd, provides low-level network access. WEBENG provides higher-level rendering, and WEBUI provides an EIKON-specific GUI. WebTL's cache uses shared memory and lightweight mutexes for high performance. Together with the font and bitmap server, it is the only server in EPOC R5 to use shared memory. Several types of "plug-in interface" provide for expansion of the web browser: protocol plug-ins at the WebTL level, engine plug-ins providing dynamic content (this is how Java applets are integrated with the browser), and UI plug-ins to control engine plug-in settings. The converter architecture CONARC and EMIME may be used to associate a viewer with web content. Web provides MIME-type recognizers, which may be used to open files in the browser from the system shell. Like Email, Web uses TCP/IP, dial-up networking and ETEL for its low-level communications, and displays connection status continually.
Comms	Comms is divided into a COMMODEL engine, and COMMS application. These use the C32 communications server and the EIKON console to deliver old-style comms app functionality.

More information about communications can be found in *EPOC Overview: Communications*.

8.3.3.3.4 Connectivity

EPOC Connect uses a sockets protocol, PLP, implemented using an ESOCK protocol module `plp.prt` on both the EPOC side (ROM-based implementation), and PC side (WINC implementation). A higher-level protocol, PRC, sends client-server requests over a PLP connection.
On the EPOC side, turning on "the link" from the System application starts the PLP server, and reserves the serial link exclusively for PLP/PRC protocols (it cannot then be used for TCP/IP applications, such as Email and Web).
The EPOC-side PLP server handles PRC requests for file manipulation, file transfer, backup and restore, synchronization, closing down and restarting applications etc. It also provides for the implementation of custom server modules (with `.rsy` extensions) so that PRC may be extended.
Printing from EPOC machine to PC is handled by an EPOC printer driver that drives PRC. On the PC side, EPOC Connect provides a server to drive the default PC printer.
EPOC Connect's converters and synchronizers require access to EPOC application data. Rather than using application data files directly, they access them through EPOC's application engine, DBMS and STORE APIs.
The PRC, PLP and application engine APIs used by EPOC Connect are built from the same source code as their equivalents on a target machine. The binaries are identical to those used to build the emulator versions of these components. On the PC, EPOC Connect runs EPOC components under the WINC version of the EPOC base.
At a higher level, EPOC Connect provides COM APIs to allow converters, synchronizers and users of PLP's RPC servers to be written in Visual C++, or Visual Basic, using standard Windows programming paradigms.
PLP and PRC are legacy protocols from EPOC's 16-bit predecessor, SIBO — though in EPOC, all APIs are 32-bit. The use of PLP and PRC enables EPOC Connect to communicate with SIBO devices (mainly in the Psion Series 3 range) as well as EPOC devices.
More information about EPOC Connect can be found in the forthcoming *EPOC Overview: Connectivity* paper.

8.3.4. EPOC Connect conversions and synchronizations

The general capabilities of EPOC Connect were described in earlier. The table below shows all supported conversions. Asterisked conversions and synchronizations are new in EPOC Connect release 5.
You can convert EPOC files to and from the following PC file formats:

File format	**PC file formats**
EPOC Word	Word for Windows 2.0, 6.0, 95 and 97 Word Perfect 5.1, 5.2, 6.0, 7.0 and 8.0 Works for Windows 3.0 WP and 4.0 WP Ami Pro 3.0 and 3.1 Rich Text Format Plain Text (ANSI and ASCII)
EPOC Sheet	Excel 4.0, 5.0, 95 and 97 1-2-3 WK1, WK3 and WK4 Quattro Pro 5.0, 6.0, 7.0 and 8.0 Works for Windows 3.0 SS and 4.0 SS
***EPOC Contacts**	Via synchronization: Lotus Organizer 2.1 and 97 Schedule+ 7 and 7a Outlook 97 and 98
EPOC Agenda	Via agenda synchronization:0 Lotus Organizer 2.1 and 97 Schedule+ 7 and 7a Outlook 97 and 98
EPOC Data	Via database conversion:

	FoxPro 2.0, 2.5, 2.6 and 3.0
	dBASE III, IV and 5.0
	CSV (Windows Comma Separated Values)
	Access 95, 97
	Via contacts synchronization:
	Lotus Organizer 2.1 and 97
	Schedule+ 7 and 7a
	Outlook 97 and 98 (but not Outlook Express)
EPOC Sketch	Windows bitmap (.BMP)
EPOC Record and Voice Notes	Windows Wave (.WAV)
***EPOC Jotter**	from EPOC Jotter to Windows text

8.4 Communications

The companion chapter to this one describes the communications and networking facilities available in EPOC Release 5, from their requirements to the design of the major components. It explains why these need to be exceptionally flexible, in order to cope with the challenges inherent in mobile ROM-based computing.
This chapter should be read by anyone who wants more technical detail of the serial communication server (C32), the sockets server (ESOCK) and the telephony server (ETEL) and of the EPOC Email and Web applications. It shows how the use of EPOC client-server architecture, coupled with mature object-oriented design and careful attention to APIs, is used throughout the EPOC communications components to offer an exceptional degree of flexibility, both for the OEM porting the operating system to new hardware platforms and for the developer writing new application software.

8.4.1. Introduction

EPOC's communication components can cope with the design constraints of mobile ROM-based devices because one of their key characteristics is a high level of *dynamic extensibility*. This enables support for new hardware and new protocols to be added to EPOC systems without restarting or rebooting, and usually allows the system to be reconfigured without shutting down any running applications. This is generally obtained by combining EPOC's distinctive client-server framework for resource allocation with a mature object-oriented design of the associated APIs. EPOC permits the run-time loading of plug-in modules on request, at all levels of operation, from applications down to the kernel, irrespective of when they were installed on the system. This applies to

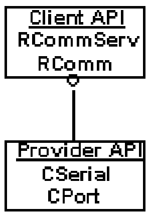
- the logical and physical device drivers (LDDs and PDDs) used by the kernel
- the communication control modules (CSYs) used by the serial comms server
- the telephone and modem control modules (TSYs) used by the telephony server
- the protocol modules (PRTs) and connection agents (AGTs) used by the sockets server
- the message type modules (MTMs) used by Email
- the Web plug-ins used by the browser

This chapter describes the implementation of this modular plug-in architecture.

8.4.2. Serial communications

There are no assumptions about any particular hardware facilities built into the operating system. To use a serial port, the kernel has to be told to load both a physical device driver (`.pdd`) which talks to a specific hardware port, and a logical device driver (`. ldd`) which implements low-level port policy, including interrupt service routines (ISRs), delayed function calls (DFCs) and routines for other essential items such as flow control and buffer management.
EPOC includes LDDs and PDDs for both RS232 and raw infrared as standard.
It is possible to drive specific ports directly. For instance, versions of EPOC running on the ARM are able to use the RDevComm API, which gives a user thread a handle to the kernel side

drivers for its built-in serial hardware. However, this doesn't implement a truly generic interface, which can be mapped to any serial device, and also doesn't allow ports to be shared. Only programs which require very fast and primitive access to hardware specific features should use RDevComm, and it should be noted that the API is subject to change in future EPOC releases.
Access to the full power of EPOC's generic serial communications can only be obtained by using the serial communications server, C32, which offers its clients an abstraction of a serial device which can be layered over any type of hardware, together with port sharing and additional features such as time-outs.



C32 APIS

8.4.2.1 Communications Server Provider API

The serial service provider API, defined in `cs_port.h`, consists of two abstract classes: CSerial, representing a serial service and CPort, representing an open and potentially shareable serial port.
Real serial services consist of dynamically loaded Communication Server Modules, which are implemented using a polymorphic DLL whose extension is conventionally `.csy` and whose second UID is 0x10000049. The first export from this DLL is a factory function for a CSerial-derived class, the main virtual function of which is NewPortL(), which creates a CPort-derived class. The virtual functions of CPort provide implementations of all functions required by the client API.
EPOC machines with built-in serial ports drive them using the `ecuart.csy` serial service. Machines with infrared ports drive them using `ircomm.csy.` Other serial port type services can be implemented as CSYs as needed. Typically, clients need no knowledge of individual CSY modules, since they ask the communications server to load whichever is defined as active in the Comms Database.

8.4.2.2 Communications Server Client API

C32's client API, defined in `c32comm.h`, consists of RcommServ, which gives the client a session to the server, and RComm, which provides an API to a serial port.
The facilities offered by RCommServ are used in a logical sequence. The client starts by calling Connect() to open a session with the server. Once this has been done, LoadCommModule() is used to load up the required CSY by name. NumPorts() returns the number of CSY modules currently loaded, while GetPortInfo() allows clients to find the details of all the ports that the CSY supports.
RComm provides many functions to access a serial port. Open() requires that a port be specified by name. There are several Read() and Write() functions with corresponding cancels, with all i/o functions capable of being set to complete on any of a number of conditions. Reading and setting serial port lines and breaks are supported, and the size of both transmit and receive buffers can be configured. RComm also provides several mode-setting and capability functions, and the API provides special classes to return port capabilities.

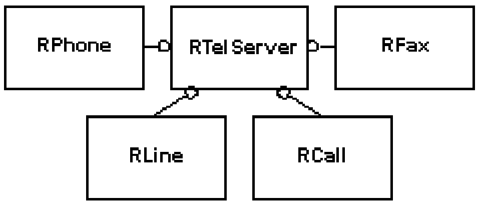
8.4.3. Telephony

ETEL is the EPOC telephony server, whose API is defined in `etel.h`. Just as C32 abstracts the different types of serial hardware into a common API that is used by all dynamically loaded .CSY modules, so ETEL performs the same function for telecommunication devices such as modems and phones, using telephony modules. These are implemented using polymorphic DLLs whose extension is conventionally `.tsy` and whose second UID is 0x100002A4. While the server at the request of the client loads TSYs, a client needs no knowledge of individual modules. They would typically instruct the server to load the default TSY, as specified in the Comms Database. EPOC provides generic TSYs for supporting either terrestrial modems (using the standard AT command set) or mobile phones. The type of TSY, which EPOC provides to support mobile phones, will vary with the location, and is under the control of the OEM. For instance, EPOC's `gsmbsc.tsy` is based on the ETSI 07.07 command set for GSM phones, and would clearly be unsuitable for use in those parts of the world where different standards, such as CDMA or PCN, prevail.
TSYs are available for handling data calls, fax calls and short message services, but not voice.

8.4.3.1 Client API

It is clearly possible for a machine to be connected to multiple phones, especially if it has more than one port. It is also feasible for a single phone to support different lines – not only do ISDN connections support concurrent multiple lines, but even ordinary POTS lines, with the addition of analogue modems, are able to handle either voice, fax or data, all of which behave quite differently. Finally, it is now quite common for a single line to support multiple calls – there are standard commands for placing open calls on hold while another is being made or answered.
ETEL therefore offers clients quite fine control over separate phones, lines, and calls that may all be available on a single device.

Clients initially connect to a root **RTelServer** session, at which point they tell the server to load a **TSY** using **LoadPhoneModule**(), find out the number of phones supported using **EnumeratePhones**(), and find out the phone name, the network type and the number of lines via **GetPhoneInfo**(). Appropriate subsessions can then be used to access the capabilities, status and separate functionality associated with phones, lines and calls respectively. **ETEL** clients are concerned only with the functionality reported by the server. A separate fax subsession is also provided, which caters for the real-time protocol requirements of the built-in **ETEL** fax server.



ETEL Client Classes

ETEL is concerned with telephony management rather than information transfer. Once calls are initiated and are under way, the server makes use of the port-sharing facilities of the C32 comms server to pass control back to its clients, and doesn't concern itself with the data that is flowing through the connection it has established.

8.4.3.2 Telephony Module design

The ETEL server uses a standard API to communicate with hardware devices via TSY modules, which are dynamically loaded and unloaded at the request of its clients. It is the TSY modules which encapsulate the actual commands used to control the hardware. The ETEL server provides abstract base classes from which all TSYs derive classes which provide the functionality needed to control the phone. Not surprisingly, they mirror the objects found on the client side. The

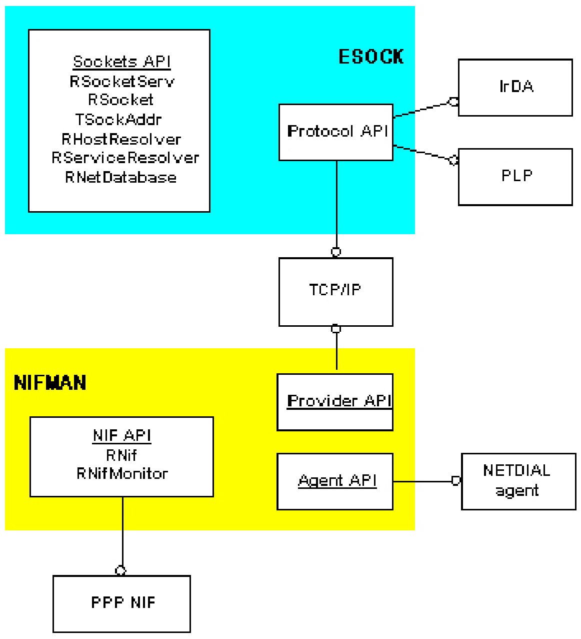
CPhoneFactoryBase class is used as a basis for the telephony devices themselves, after which classes derived from CPhoneBase, CLineBase and CCallBase are used to control phones, lines and calls respectively. Because of the real-time constraints of the ITU fax protocols, the CFaxBase derived class in the fax server, which provides the necessary fax functionality, has additional responsibility for the progress of the ITU-T.30 fax negotiations and the transfer of ITU-T.4 fax data.

8.4.4. Sockets and TCP/IP

EPOC networking communications is provided by the socket server, ESOCK, together with the network interface manager, NIFMAN.

8.4.4.1 Sockets

ESOCK's client API is defined in `es_sock.h`. The main classes are RSocketServ, a session to the server, and RSocket, a communications socket. The sockets API is modelled on the BSD sockets API and provides a protocol independent interface to networking functions.
TSockAddr defines a 32-byte socket address whose format depends on the socket protocol in use.
RHostResolver, RServiceResolver and RNetDatabase provide asynchronous lookup facilities for host names, services and general queries on network databases.
EPOC's native sockets API, like all its APIs, is object oriented. Programs written in EPOC C++, using this API, can be very efficient. Legacy programs written in pure C are also supported through the EPOC C Standard Library, which provides POSIX-like system, calls. These are layered over EPOC's object-oriented sockets and file server APIs to provide an implementation of the BSD C API, including unification of file and sockets services. Although this only supports single sockets, it does allow existing engine code to be ported to EPOC with relative ease.



Implementation of ESOCK and NIFMAN APIs

8.4.4.2 Protocols

ESOCK's provider API is defined in `es_prot.h`. It consists of classes providing server-side functions for the client API.
A sockets protocol is implemented using a polymorphic DLL whose extension is conventionally `.prt` and whose second UID is 0x1000004A.

ESOCK scans its `esock.ini` file (in `\system\data\`) each time it is started, to see which protocols and services are defined, and which protocol modules can be loaded.
As with Communications Server modules and Telephony Server modules, Sockets Server protocol modules are loaded and unloaded dynamically. A single protocol module can contain one or more related protocols (a protocol family). They must be able to identify themselves and their capabilities, and at least one protocol in a family must provide socket services. Protocol families may also provide service registration and name space resolution services. Examples of protocol families are TCP/IP, IrDA (consisting of raw IrMUX, IrTinyTP, IrLAP, IrLMP and IrObex), and the PLP Link Protocol used by EPOC Connect.

8.4.4.3 TCP/IP

The TCP/IP sockets protocol family is the basis of all Internet applications, including Email and Web. The stack EPOC provides in `tcpip.prt` implements the following protocols:

- TCP (Transmission Control Protocol) as specified in RFC 793, together with the RFC 2001 Slow Start Algorithm
- UDP (User Datagram Protocol) as specified in RFC 768
- IP (Internet Protocol) as specified in RFC 791, together with datagram broadcasting as specified in RFCs 919 and 922 and the RFC 1144 Low Speed Link Header Compression
- ICMP (Internet Control Message Protocol) as specified in RFC 792, with RFC 950 Standard Subnetting
- DNS (Domain Name System), consisting of those parts of RFCs 1034, 1035 and 1101 which are appropriate to IP clients.

The header file `in_sock.h` defines additional constants and types relevant to TCP/IP networking. For instance, TInetAddr is derived from TSockAddr to represent an Internet address: it provides functions which set and parse the underlying TSockAddr data according to the rules of Internet addressing.

8.4.4.4 Network interface manager

A socket provides communications to a remote network entity specified by an address. In order to send a packet to an address, a route must be identified. If a protocol (such as IP) has no route available to some destination machine or destination network, the system uses the network interface manager (NIFMAN) defined in `nifman.h` to start up an appropriate network interface. In turn, NIFMAN uses an *agent* to establish an interface. The standard agent provided is `netdial.agt` which enables the use of dial-up networking via an Internet service provider and EPOC's NETDIAL component, which is defined in `netdial.h`.
For users of dial-up applications, the network connection status is of some importance. Firstly, it shows whether the application is able to immediately perform network-oriented functions. Secondly, it indicates whether connection charges are being incurred — particularly important on EPOC machines integrated with mobile phones. For this reason, NIFMAN provides a monitoring API which allows applications to monitor the status of the network connection, and display it to their users. At any time, the RNif:Progress function can be used to obtain the last reported connection status. Alternatively, making asynchronous calls to RNif:ProgressNotification() ensures that an application will be kept informed whenever the network connection changes status or encounters an error.

8.4.4.5 PPP

Once an agent has established a connection, a data link protocol is used to drive that connection. EPOC provides as standard a comprehensive implementation of the Point to Point Protocol (PPP) as specified in RFC 1661. Among the features implemented are:

- the use of LCP echo/reply packets to ensure link quality
- PPP Authentication Protocol (PAP), and Challenge Handshake authentication protocol (CHAP) with the Microsoft proprietary extensions
- HDLC-like framing

- IPCP with both DNS and NBNS (NetBios Name Service) extensions
- LCP identification and time-remaining extensions
- PPP compression control protocol (CCP).

EPOC also includes Microsoft proprietary CBCP callback types 1 - 4. These are extensions to the PPP protocol which can be negotiated after the server has been connected to in the normal way. The EPOC machine hangs up the line, whereupon the server dials it back, and communications are re-established. The advantages of callback are that it saves callers money and provides an added level of security for RAS servers.

8.4.5. Messaging

EPOC's comprehensive Email application caters for all types of messaging. Its architecture is influenced by some characteristic EPOC design goals:

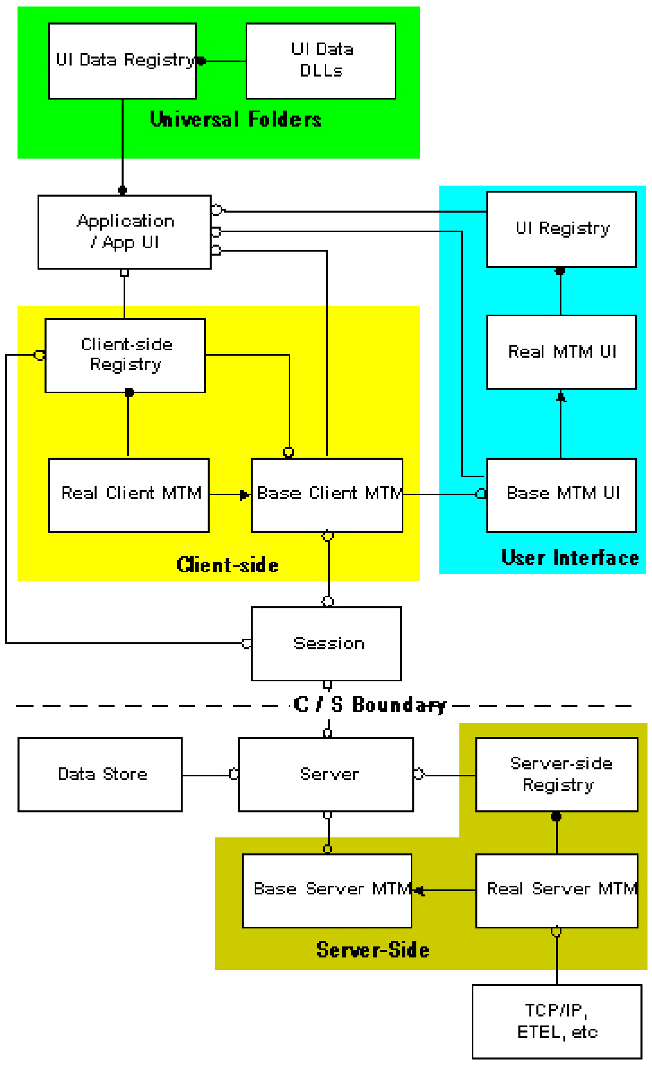
- it splits the user interface from the rest of the application – this facilitates the task of porting to different UIs, and is a characteristic that Email shares with many other EPOC applications.
- access to data is handled via EPOC's client-server mechanisms – this enables multiple applications to open simultaneous sessions to the messaging store.

Email provides a UI-independent Universal Messaging Engine, which is capable of handling all types of message in terms of their common properties (such as address and content), irrespective of any differences in either the way they represent data internally or the mechanisms they use for transport.

8.4.5.1 Messaging Type Modules

Dynamic extensibility is provided by the run-time loading of Message Type Modules (MTMs). The functionality encapsulated within these MTMs is comprehensive enough to allow completely new messaging sub-systems to be added to Email. Each one is implemented as a set of four polymorphic DLLs with separate APIs. Each DLL is derived from an abstract class whose methods need to be implemented for each message type.

- Server-side MTMs are concerned with formatting and transporting of messages. The implementation of all such message specific protocols (including SMTP, IMAP4, SMS, POP3, and fax) is handled within the server-side MTM.
- Client-side MTMs both interpret the contents of a message and provide a minimum set of generic functions. For example, a Reply() function is always available, which makes it easy to implement a consistent user interface for replying to any message irrespective of its type.
- User Interface MTMs provide appropriate editors and viewers.
- Universal messaging folders are made possible by a memory-efficient layer of User Interface Data MTMs. These provide the engine with sufficient information to enable message types to be correctly identified and represented to the user without needing to load the UI and client MTMs



EPOC Messaging Architecture

8.4.5.2 MTM registries

Since MTMs are not system-wide components requiring user intervention, the comms database is not used to store details of those available at any time. Instead, the messaging system maintains four *registries*, which correspond to the four base MTM types. The registries don't simply maintain lists of available MTMs and message types, but also include the necessary functions for loading DLLs and then unloading them when they are no longer needed. In fact, the registries have all been equipped with timers, which are used to unload DLLs, a given period after they have become unused. This avoids repeated loading and unloading when the user is switching between messages in the inbox.

8.4.5.3 Message storage

Most message data is stored in EPOC filestores, each with a number of separate streams to allow for the separation of generic and message-type-specific data. For instance, a generic stream in the filestore is always used to hold the message body as EPOC rich text. This allows generic application modules, such as the Messaging Application itself, to access body data without knowledge of the message type.
Message-type dependent data (such as a recipient list) is stored in a type specific stream in the filestore. Protocol-dependent data, such as ISP settings and addressee data, is stored in a separate filestore associated with the relevant service, and which only MTMs of the appropriate type can access.

Any message attachment can be saved as a normal file containing ordinary binary data. Conversion and viewing of attachments is handled by EPOC's Application Architecture, which makes use of an extensible range of dynamically loadable recognizer DLLs to identify the MIME type of a file and invoke suitable applications to handle it.

8.4.5.4 Observers and notification

The server API includes a pair of interface classes, which define *session observers* and *entry observers*. A class which is defined as a session observer will be notified of events such as shutdown warnings, new message arrival, and new MTM registration. Entry observers will be notified when the data representing an individual entry changes, or when requested access to a message store has been successfully gained.
Session observers are of particular interest in terms of the dynamic extensibility of the messaging architecture - they allow new MTMs to be registered across the system, without shutting down the messaging application. Client processes already in existence will also be notified - since all registries are defined as session observers, they will be able to update their lists of known and available MTM types.

8.4.5.5 External message API

The Send-As interface, which is not UI dependent, permits any application to become a client of the messaging server in order to save messages in its outbox on a store-and-forward basis, for later dispatch. EPOC Web uses this API for supporting HTML features such as mailto: URLs.
Send-As provides a simplified method of finding out what MTMs are available, what their capabilities are, and what services they support. The most obvious capabilities are those relating to conventional email, such as whether attachments are supported, whether the message body is restricted to plain text, and the maximum possible message size. However, it is well worth noting that EPOC's messaging engine is a universal one, and that the need to establish other capabilities is less immediately apparent – for instance, it cannot be assumed that all MTMs are capable of both sending and receiving. An outgoing pager service is an example of a send-only transport, while GSM CBS broadcasts constitute a message type that can only be received. Other types of messaging may have their own highly specific requirements, such as an image rendering capability for fax.
The Send-As API permits messages to be validated by the messaging server without forcing the client application to launch any message specific user interface – this is particularly useful for checking that an address is valid for the chosen message type before committing it to the outbox.

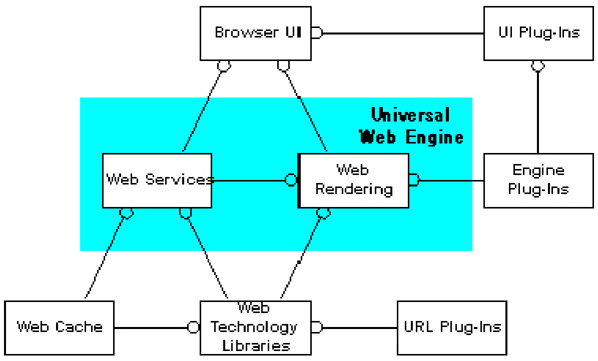
8.4.6. WEB

EPOC's Web application is implemented as separate UI and engine components, which lie above the Web Technology Libraries (WebTL) supplied by STNC Ltd. The architecture is dynamically extensible and fully supports Plug-Ins.

8.4.6.1 The Web engine

This engine is delivered as two sets of components:

- The Web Rendering engine deals with the manipulation and rendering of the components of a web page. It relies on the WebTL libraries to fetch data tokens from the network, and also makes use of the Web Services Engine for supplementary functions.
- The Web Services Engine contains the remainder of the browser functionality. This includes authentication, history lists, network monitoring, bookmark control, setting up of proxies and configuration of both session-based and global settings.



EPOC Web Browser Architecture

8.4.6.2 The Web Technology Libraries

The WebTL layer obtains document data, either over the network connection or from a local file, and converts it into tokens suitable for rendering by the Web engine. It also manages the storage of data tokens in the Web Cache, which can also be directly accessed by the browser when displaying pages to the user.

8.4.6.3 Web Plug-In components

Plug-Ins are used for extending the basic functionality of the application in areas such as:

- displaying specific MIME types (such as video images)
- additional URL schemes (such as ftp: or telnet:)
- additional mark-up languages (such as XML)
- active content (Java applets)

EPOC Plug-Ins are thus rather broader in concept than (say) the Netscape version, which can only be used for MIME implementations.
Symbian can provide facilities for producing and adding third-party EPOC Plug-In components, which may need to be implemented at all levels of the Web design.

8.4.6.3.1 WebTL Plug-Ins

WebTL Plug-Ins provide support at the lowest level for new protocols or formats (such as streaming audio). There is no direct communication between engine Plug-Ins and WebTL Plug-Ins, and all interaction is handled by the main browser-WebTL API.

8.4.6.3.2 Web engine Plug-Ins

Within the Web engine, Plug-Ins provide additional rendering services and need not necessarily have a UI component. They are small modules which control a rectangular area of the browser window, into which they can either render themselves directly or embed other 'child' Plug-Ins. All Plug-Ins are accessed via a registry, which maintains a list of those available and will dynamically load and unload them as required. Engine Plug-Ins are essentially handlers for either HTML tags or specific MIME file types. They all support the same generic API to the browser engine, which supports loading, drawing, printing, event handling and streaming.

8.4.6.3.3 UI Plug-Ins

These augment the user interface, providing suitable control mechanisms for a corresponding Web engine Plug-in. They are typically responsible for obtaining settings, preferences and other inputs from the user, with the results being passed back to the engine.

8.4.7 Design

This chapter should be read by anyone who needs an overview of EPOC's communications and networking facilities. It consists of three parts.
The first part outlines the key concepts underlying EPOC communications, which have been designed specifically for mobile users and mobile devices. The middle section describes the individual components from the user's point of view, starting from the low level device drivers and servers and ending with EPOC's communications applications. The chapter concludes with a more detailed description of how the various components interlock to provide EPOC applications with dial-up networking facilities.

8.4.7.1 Introduction

EPOC is a powerful software platform for wireless information devices. Its general aims are to provide compact, fast and fully-featured applications to the users of ROM-based wireless information devices; to support Symbian's licensing strategy by allowing the component mix to be tailored for different EPOC devices; and to provide a basis for future application and component development. Among the currently available communications components are:

- serial and infrared device drivers
- serial communications server
- sockets server
- infrared protocol suite, including IrDA, IrCOMM and IrObex
- TCP/IP with dial-up support
- PPP with callback support
- Internet mail, including SMTP, POP3 and IMAP4
- support for MIME attachments
- fax send and receive
- SMS send and receive
- web browsing
- PC connectivity and synchronisation
- end-user applications for terminal emulation, web browsing, and integrated messaging

Since Symbian licenses EPOC for a range of devices encompassing both smartphones and communicators, not all components are necessarily suitable for inclusion in all platforms.

8.4.7.2 Design goals

EPOC's communication components have to be exceptionally flexible, in order to cope with the challenges inherent in mobile ROM-based computing. It is critically important that end-users are able to tailor the active mix of components on any individual machine quickly, easily, and safely, without restarting or rebooting, and without losing any data.
The highly mobile nature of wireless information devices means that the services and devices that they find readily available are likely to change as they move from place to place.

- EPOC has been designed for machines which use dial-up connections to Internet Service Providers (ISPs) . Users can switch between ISPs, phones, and modems with minimal difficulty, and can then dial out reliably from wherever they are. No substantial reconfiguration is needed for each new location.
- EPOC is also designed to cater for the fact that as machines move about, any applications providing services such as connectivity, synchronisation, and printing may need to support different hardware and software at each location.

The fact that wireless information devices are ROM-based means that special care has had to be taken to allow for the fact that communications protocols and standards continually evolve.

- EPOC is designed to allow the installation of new features such as additional protocols as they are needed. The fact that both the operating system and the core applications are built into the ROM does not mean that machines have to be reprogrammed or returned for service if an application has to be enhanced to access a service that did not exist when the machine was built.

EPOC's communication components can cope with all the design constraints of mobile ROM-based devices because one of their key characteristics is a high level of *dynamic extensibility*. This is generally obtained by combining EPOC's distinctive client-server framework for resource allocation with a mature object-oriented design of the associated APIs. EPOC permits the run-time loading of plug-in modules on request, at all levels of operation, from applications down the kernel, irrespective of when they were installed on the system. This applies to

- the logical and physical device drivers (LDDs and PDDs) used by the kernel
- the communication control modules (CSYs) used by the serial comms server
- the telephone and modem control modules (TSYs) used by the telephony server
- the protocol modules (PRTs) and connection agents (AGTs) used by the sockets server
- the message type modules (MTMs) used by Email
- the Web plug-ins used by the browser

The way that these modules implement extensibility is discussed in more detail in the companion to this paper EPOC Communications: Implementation.

8.4.7.3 End-user goals

EPOC delivers two capable end-user Internet applications:

- The EPOC Email application supports POP3/SMTP/IMAP4 Internet email, and fax send/receive. This is integrated with fax printing. When used with a suitable mobile phone, SMS is also supported.
- EPOC Web supports HTML to HTML 4.0 specification, including support for forms, frames, GIF and JPEG graphics, animated GIFs, and downloadable Java applets.

Both Email and Web are designed as universal engines, with details of add-on modules being kept in associated registries. This means that service-specific and protocol-specific components can be installed and then loaded on demand.
In addition to the above:

- A serial communications application provides old-style terminal emulator access to dial-up services. TTY and VT100 emulations are supported, with automated scripting, and XMODEM and YMODEM file transfers.
- EPOC's communications facilities provide support for connection to PCs and other machines, for backup and restore, data synchronisation, printing and file conversion.
- Infra-red beaming of industry standard vCard and vCalendar objects is supported to and from any machine that implements the IrObex protocol, while any object can be beamed between two EPOC machines.

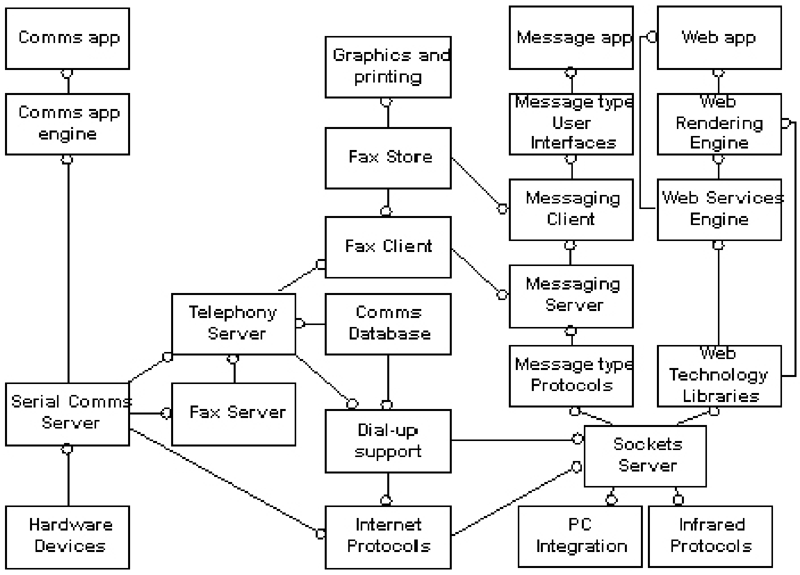
8.4.7.4 Implementation goals

EPOC is available in a number of different configurations, called DFRDs (Device Family Reference Designs). As their name implies, these EPOC variants are aimed at families of machines with common hardware characteristics. For each configuration, a different GUI has to be implemented to properly handle the wide variations in screen sizes and available input devices. In order to ease the task of porting between DFRDs, the communications applications have been split into GUI-independent and GUI-dependent parts. Only the latter need to be replaced when porting to a new DFRD.
In keeping with Symbian's established practice, GUI-independent parts of applications are often further split into data-handling and graphics sections. For instance, separate components are used for fax send/receive, and for viewing fax data.
Like everything in EPOC, the communications facilities are implemented efficiently so as to use as little ROM and RAM as possible. In addition, the component granularity is quite fine to allow OEMs to configure the exact components used in a particular EPOC machine. The system architecture also permits OEMs to choose which components should be ROM-based and which should go in RAM. Especially where ROM budgets are tight, this added flexibility can be used to ensure that memory on any specific EPOC implementation is used as efficiently as possible.

8.4.7.5 Future goals

Symbian's strategy for wireless information devices includes support for Wireless Application Protocols (WAP), a suite of networking protocols which makes efficient use of both mobile communications bandwidth and limited display size; Bluetooth, a short-range radio protocol suite for communicating and synchronising with other wireless information devices and with PCs; and continued messaging, browsing, Internet and security enhancements.

8.4.8. Components



Communications Components, and their main relationships

The major EPOC communication components are shown above. We will examine each in turn.

8.4.8.1 Client-Server architecture

At the heart of EPOC's communications are its servers — the serial comms server, the telephony server and the sockets server — together with the comms database.
Accessing system facilities via servers instead of using them directly means that they may be shared by multiple clients. This is a highly significant architectural advantage. The use of servers also encourages a high degree of functional abstraction; this means that clients can rely on being able to use the same API without needing to know the exact hardware device or software protocol which might be used to provide the service required.

8.4.8.2 Serial comms server

The serial comms server provides a serial port API to clients. It also specifies a provider interface which allows any serial-like protocol to appear as a port. EPOC uses this to access both real serial communications over an RS232 line and also IrCOMM over an infrared device.

8.4.8.3 Telephony server

The telephony server provides a standard API which enables its clients to initiate, control and terminate data, fax and voice calls using the same methods for any hardware. EPOC provides, as standard, generic modules for GSM mobile phones and also for terrestrial modems using AT command sets. The architecture makes the task of adding support for devices implementing different protocols quite straightforward.

8.4.8.4 Sockets server

The sockets server provides a BSD-like sockets API to clients. It also specifies a provider interface which allows sockets to be implemented using a variety of protocols. EPOC provides Internet and

Infrared protocol suites as standard, together with the PLP Link Protocol, which is used for PC integration.

8.4.8.5 Comms database server

The comms database is implemented using EPOC's DBMS server and is the central repository of all information regarding phones, modems, protocols, network access phone numbers, Internet addresses and physical locations, together with details of device drivers available to the system servers and the EPOC kernel. The control panel is generally used to create and update entries in this database, though it can also be written to by other software components such as installation programs. The database also records default settings current at any time.
This makes it possible to write components which are both easy to use and genuinely device independent. For example, a program that needs access to a serial service can consult the comms database to identify the default module to load, with a simple optional dialog box being used to allow the user to confirm the setting. No lines of code need be changed to allow such a program to function with standard RS232 serial lines, or IrCOMM infrared protocols, or any other current or future type of serial port.

8.4.8.6 Fax

The telephony server contains a separate fax server, which implements the fax transport protocols, provides raw send and receive support, and offers its clients a dedicated fax session API. On the client side, a fax data store is provided, together with views for stored faxes which provide fast navigation and zooming. A fax printer driver allows any application that can print to send faxes. Faxes are integrated with the EPOC Email application. Outgoing faxes are stored in the outbox and sent when convenient; while incoming faxes are placed in the inbox.

8.4.8.7 Infrared

Infrared support assumes the availability of a serial infrared (SIR) device on EPOC hardware. A full IrDA stack is supported by EPOC. The sockets server provides IrLAP, IrLMP, IrTinyTP, IrObex and IrMUX; IrCOMM is provided as a serial service, which uses the underlying IrDA sockets services.
EPOC fully supports "beaming" data over the infrared connection. All kinds of data can be sent from one EPOC machine to another. Additionally, information from applications such as Agenda and Contacts can be transferred to and from non-EPOC systems provided they support the IrObex mechanisms for infrared object exchange and are capable of understanding the industry standard vCalendar and vCard data formats.
Infrared printing is also supported.

8.4.8.8 PC integration

The EPOC side of PC integration consists of the PLP Link Protocol. Using PLP in EPOC builds on an established and proven protocol, with a wide base of existing users of Psion machines. It was specifically designed for mobile ROM based computing, and combines a very small memory footprint with ease of use and reliability. The data link and transport portions of PLP run over a point-to-point serial connection, and are loaded by the sockets server as the `plp.prt` protocol module. Both RS232 and IrDA links are supported.
PLP accesses the protocol module via a session server, `plpsvr.dll`. As well as being a client of the socket server, this component is a server in its own right, providing facilities for both local and remote clients. As well as controlling the remote link on the EPOC device, it offers clients access to a remote filing system. Importantly, it also specifies an interface for loadable `.rsy` remote server modules which can be invoked by its clients during a communications session; it is therefore possible to extend the system by adding custom modules. Those provided include a remote procedure call server and Windows printing. These facilities are directly used by EPOC Connect and MacConnect to provide backup to desktop machines, an interface for application installation, and the means for file format conversion and synchronisation.

On the Win32 side, running under either the EPOC Emulator or WINC, the PLP Remote Communications (PRC) component layers on top of the PLP components, and provides Win32 API to EPOC facilities.

8.4.8.9 Comms app

Basic serial terminal support is provided by the EPOC Comms application. It is split into an engine and an application; the application requires that the GUI provide console support to display the emulator.
The engine supports login scripting, XMODEM and YMODEM file transfer, and emulation of TTY and VT100 terminals.

8.4.8.10 EPOC Email app

EPOC integrates supports for all types of messaging in a single Email application. The architecture implements sets of layered Message Transfer Modules (MTMs) for specific message types, which are accessed through APIs derived from abstract classes.
EPOC provides standard MTMs for Internet mail (supporting SMTP and either POP3 or IMAP4) together with fax and SMS. At any time it is possible to enhance existing functionality or add support for new message types, views or protocols simply by registering additional MTMs. Completely new messaging sub-systems can also be added in this way. For example, newsreader applications requiring NNTP facilities would be possible if MTMs implementing the appropriate RFCs (977 and 1036) were to be added to the messaging system.
As with other EPOC communication components, the client-server architecture makes it possible for many applications to have multiple simultaneous access to message data.
EPOC Messaging also includes simple programming interfaces which allows any application to place messages in the outbox for sending at a later date.

8.4.8.11 Web app

The EPOC Web application provides a powerful HTML browser, which includes support for downloadable Java applets.
In common with other EPOC applications, the Web application relies on UI-independent engines. The engines themselves use EPOC's Web Technology Libraries (written by STNC Enterprises Ltd of Bury St Edmunds, England) to fetch and tokenize the Web data , either from the Internet or from local files.
There are two Web engines. The Rendering engine interprets the tokenized data and turns it into a Web page which can be displayed by the UI-dependent part of the application. The rest of the browser functionality is included in the Services engine. All levels of the Web architecture can be extended by means of Plug-Ins. Unlike most browsers, which are limited to Plug-Ins that enabling the browser to understand new MIME file types, EPOC Web Plug-Ins can extend the functionality of the application in other areas. For instance, completely new markup languages (such as XML) could be implemented in this way.

8.4.9. Dial-up networking

8.4.9.1 Introduction

Since EPOC machines have no permanent connection to the Internet, one must usually be established before any network data packets can be sent or received. The sockets server, ESOCK, contains a network interface manager (NIFMAN) responsible for route setup for sockets protocols. NIFMAN selects the correct network interface for a route, and initialises it when necessary. Initialisation typically involves establishing a dial-up connection as well as starting up an Internet protocol such as PPP.
NIFMAN hands responsibility for establishing new connections to its NETDIAL agent. On behalf of NIFMAN, NETDIAL obtains the current default settings for dial-up from the comms database, and optionally requires users to confirm these before proceeding to dial and connect. The algorithms used for dialling automatically pick up the correct dial codes and modifiers for each physical location, including PABX and chargecard (calling card) support. The telephony server, ETEL, controls the mechanics of the dialling process, after which NETDIAL establishes

the connection to an ISP. This can either be governed by a log-on script, or else handled automatically within the Internet protocol suite.

8.4.9.2 Connection logic

NETDIAL's purpose is to dial up the Internet service provider (ISP) and establish an authenticated connection before handing control back to the TCP/IP protocol stack, which can then use that connection via the PPP network interface. When this sequence has completed, the data packet that caused the connect sequence can finally be sent. NETDIAL handles these tasks separately from any individual application, so that it is not necessary to duplicate the same code in every component that might need to make an Internet connection.
Dialling involves:

- looking up the current location, the current modem settings, and the current ISP in the comms database
- invoking the NETDIAL dialog server to obtain confirmation of these default items, as required
- in conjunction with EPOC's DIAL component, ensuring that the phone number for the current ISP is resolved, taking into account the current location and any other relevant settings, such as phone type and charge card
- establishing a connection to the telephony server, ETEL, which is responsible for controlling the phone or modem and for establishing a telephone connection
- after connection, optionally handling any logging on by using a text-mode script if neither CHAP nor PAP authentication protocols are available over the data link protocol.
- returning control to NIFMAN

8.4.9.3 User interface

8.4.9.3.1 At setup time

Because dialling involves lookup from the comms database, the relevant items must be set in advance. As described previously, these is done through the Control Panel. Once all applicable settings have been entered, the user can connect to the Internet from any Internet application on the EPOC machine.
The Control Panel applications have been made as simple to use as possible. Whilst setting up connections to most Internet service providers (ISPs) is now relatively trivial, setup information on many popular ISPs — including telephone numbers and any special scripts — is nevertheless made available with EPOC Internet applications, and is also shared on the Web sites of EPOC OEMs. Likewise, settings for popular modem types are supplied. This means that most setup is a simple matter of selection from lists of modems and ISPs.

8.4.9.3.2 At connection time

As well as initial setup from the control panel, it is important to provide some visible user interface when a connection is made. There are several reasons for this:

- for security reasons, the user may not have wished to include their user ID and password in the ISP settings. If this is the case, they must be typed in at connect time.
- as a connection is made, it is reassuring for the user to see feedback about the state of the connection
- because EPOC users are highly mobile, it is essential to be able to present a choice of connection options *at connect time* rather than forcing the user to use the control panel before starting to connect

For connection options and for authentication, the NETDIAL dialog server (NDDLGSVR) provides a user interface. NETDIAL specifies an API which the dialog server then uses. All functions in this API are asynchronous: they complete when the user dismisses the dialog.
Progress of a connection is not reported using the dialog server. An application which wishes to report the progress of a connection must use the NIFMAN interface directly.

8.4.9.3.3 Typical user setup behaviour

A typical user of EPOC will perform the following setup tasks:

- when first installing software that uses dial-up network, the user will specify a modem and an ISP; also, the user will check that the default dial settings (home, office and mobile) are appropriate
- when making a connection, the user will select between home, office and mobile: this is all that is needed to use dial-up networking from any mobile phone, and from a fixed home and office
- if travelling abroad and using a "home" type connection to the public phone network (when no prefix is required before dialling out to the ISP) then the user must use the Dialling icon on the Control Panel to set up a new location, such as "Home, Korea" and enter the country from there
- similar changes must be made for other types of location when travelling

Thus, EPOC has been designed to minimise the work necessary to maintain dial-up networking functionality for mobile users.

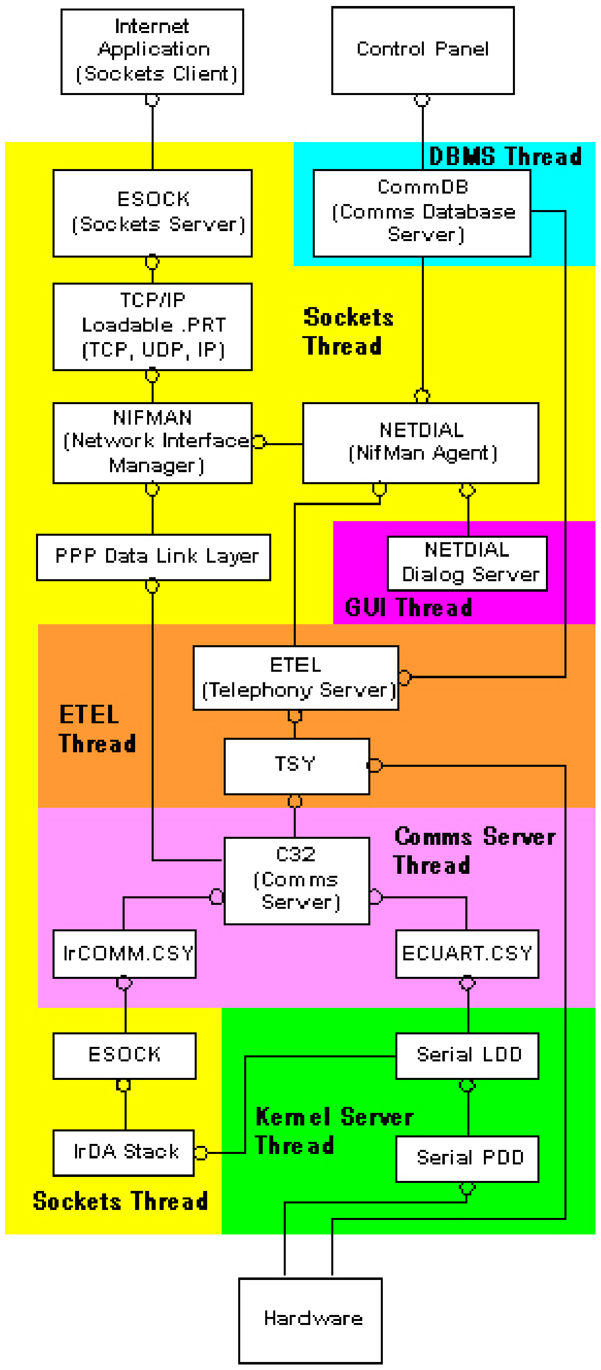
8.4.9.3.4 Reconnection

If a connection is inadvertently terminated, but a user later wishes to send or receive data over the connection, they may be asked whether they wish to reconnect. Reconnection always uses the same settings as the original connection.

8.4.9.4 Dial-up Networking Architecture

We've seen that establishing and using an Internet connection from scratch involves many different EPOC components. We conclude with a graphical overview of the way that they might interact, showing the following links:

- A component opens a session with the sockets server, ESOCK and tries to connect to a specific IP address.
- ESOCK dynamically loads up the TCP/IP protocol family.
- IP asks NIFMAN for a route.
- As no connection exists, NIFMAN loads its NETDIAL Agent to set one up.
- Optionally, NETDIAL invokes its dialog server to check with the user that the comms database settings they had previously written via the control panel are acceptable.
- At this point, the EPOC DIAL module may invoke the World Server to resolve the phone number for the current location .
- NETDIAL opens a session with the telephony server, ETEL, and asks it to make the call and connect.
- ETEL looks up the relevant details for the current port, location, ISP, and phone or modem in the comms database
- ETEL loads the appropriate TSY module.
- The TSY opens a session with the communications server, C32
- The TSY tells C32 and the EPOC kernel to load the correct CSY comms server module and the appropriate device drivers.
- At last a hardware connection exists. ETEL can talk to the phone or modem and tell it to dial up the ISP.
- On connection, ETEL loans the port back to NETDIAL, which can now run any scripts necessary to log on to the Internet.
- The NETDIAL Agent hands control back to NIFMAN, which now activates the PPP link protocol for any authentication, and to control the remainder of the call.
- When PPP asks for the line to be dropped, NIFMAN returns control to NETDIAL, which returnes the port to ETEL for termination of the call, and the whole stack now unwinds.



EPOC Components and Dial-up Networking

The figure above illustrates how control is passed from the originating application down to the kernel and from there to the hardware. Every request to a server passes control to a different thread. The servers themselves are extremely versatile objects that can contain many different components, and are also quite happy acting as clients to yet other servers.
Other items worth noting are:

- Some connections, such as the use of the World server to resolve phone numbers, are described but not shown.
- Other interactions, such as the possibility of a TSY controlling the hardware directly, and the different ways that the infrared and serial CSYs work, are shown but not described.
- Various other useful features of EPOC's dial-up networking, such as the use of NIFMAN to display dial-up progress within an application, are neither described nor shown.

- Since the Control Panel icons and the NETDIAL dialog server (NDDLGSVR) are GUI dependent components which are needed to provide direct access to the comms database, different implementations of these are required for each DFRD.

8.5 EPOC R5 New Features

8.5.1. Introduction

EPOC Release 5 contains many new features. In this chapter, we provide more technical detail on those features as they affect the core components of EPOC. This chapter contains sufficient detailed information for established EPOC users to plan how to use EPOC R5 with some confidence. Some of this information assumes detailed familiarity with EPOC SDKs. Much of the paper also to EPOC terms and components: for an introduction to these, see *EPOC Core chapter*.

8.5.2. Pervasive changes

EPOC Release 5 incorporates many changes which affect almost all EPOC components, and almost all software development.

Performance improvements built into the base result in better context switching between favoured processes, faster message passing, and many other improvements in kernel and executive performance.

Every component has been affected by work preparatory to a full Unicode release. EPOC has since its beginning contained support for a Unicode build with no source code changes, provided that application developers made the appropriate distinction between binary data (always 8-bit bytes) and character data (8-bit characters in the narrow build, 16-bit characters in the Unicode (wide) build). Further support for Unicode requires that this assumption be checked in all source code, by building and testing the entire EPOC system in Unicode. Toolchain improvements have been implemented, together with improved text formatting, and frameworks for front-end processors (FEPs) to allow entry of Far Eastern characters using handwriting recognition, or keyboard techniques that have evolved in the relevant locales. This represents a major advance in EPOC's Unicode-readiness. However, EPOC R5 machines and SDKs support narrow characters only.

Colour support has been built into the GDI, bit blitter, bitmap tools, window server and EIKON. EIKON now supports two colour schemes — one suitable for greyscale displays, one for displays with at least 16 colours. Application colour schemes, and colour icons, have also been added.

The emulator is much more flexible than before. Any international PC keyboard variant is now supported, the emulator can run from any directory including on a CD-ROM, and changes to allow easier launching of Java applications have been made.

The debug and release builds of EPOC are now interoperable. This means that, say, a debug version of an application may be run on a release-mode EPOC machine. For the first time, this enables debug builds to be built and debugged on real users' machines. This interoperability was achieved while preserving binary compatibility in the narrow ARM REL build, narrow WINS DEB and narrow WINC DEB builds — but in no other builds.

Many changes have been made to the kernel to improve its portability to other ARM and non-ARM CPU architectures, and to other hardware variants. As a result, kernel-side binary compatibility has been broken and all device drivers must be re-written: old device drivers will be recognized and prevented from loading.

These changes have resulted in toolchain changes which have implications for most EPOC C++ projects, if only at the `.mmp` file level. However, very few changes have been made in source compatibility, which means that almost all source code can be rebuilt for EPOC R5 without change. In terms of binary compatibility, all but a very few changes to user-side APIs have been backward compatible, so that programs built for EPOC R3 will generally run without rebuild under EPOC R5.

Some changes in the user library provide an opportunity to write type-safe code, or code with higher performance and smaller space requirements. These include descriptor literals, and range-checked C arrays. Code written with an EPOC R5 SDK, using these features, will run successfully under pre-EPOC R5 machines, provided that no other EPOC R5-specific features have been used.

EPOC R5's emulator and WINC components were built with Microsoft Visual C++ Release 5, and the C++ SDK also includes support for Microsoft Visual C++ Release 6. The emulator debug databases released with the C++ SDK are in Microsoft Visual C++ Release 5 format, and cannot be browsed using Microsoft Visual C++ Version 4. Thus, developers who still use Version 4 must upgrade to Version 6 (Version 5 is no longer available). The GNU C++ Compiler and associated toolchain, used for ARM builds, are the same as for previous EPOC releases.
EPOC R5 is the first full, externally-delivered, Symbian product. All source code, help, bitmaps, text etc has been changed to reflect Symbian's distinct identity.

8.5.3. Changes by component

8.5.3.1 Base

The user library has been enhanced by additions of APIs for range-checked C arrays, better literal string handling, provision of a console size which always produces a full-screen console, transfer of cleanup-related functions from STORE to E32 headers. It is recommended that new programs take advantage of these new facilities and, where possible, old programs be updated to do so.
Project files required by `makmake` have changed to support Unicode builds of EPOC, kernel changes, and more hardware targets than MARM. As a result, almost all `.mmp` files constructed for SDKs prior to EPOC R5 must be changed. In most cases, the required change is a trivial one-liner.
The kernel has changed significantly to provide better support for a wider range of target architectures, better interrupt-handling, power management and peripheral bus architectures for i/o devices, and other API changes. As a result all device drivers must be rewritten. UID changes have been used to ensure that device drivers written with a pre-EPOC R5 SDK will not even load on EPOC R5.
The PC Card interface supports i/o cards now (in addition to compact flash (CF) memory cards). The architecture has been changed to use a peripheral bus interface which is an optional part of an EPOC build, allowing it to be omitted for machines which do not need it.
The emulator has been changed to allow it to function effectively in any directory (not just `\epoc32\release\wins\deb\` and its peers, together with `\epoc32\data\` and `\epoc32\wins\c\`), and to allow much more flexibility in configuration. Changes in WINC make it more suitable for writing convenient utility programs. The EPOC toolset has not been updated for location independence, so that when developing using the C++ SDK, the old directory locations must still be used.
Many performance improvements have been built into E32 and F32, including machine-coding some functions, using faster algorithms for math, and changes to memory management which reduce cache flushes required when context switching between system servers.
Math functions have been revised to give results which have been certified as conforming with Java Compatibility Kit™ requirements, which are more specific than their previous IEEE754 conformance.
Unicode is supported with more collation and character-comparison functions, Unicode character tables, and more supported locales.
Files use UTC timestamps (rather than local time), to provide improved synchronization support.
Many changes and fixes to loading of `.exe`s and DLLs have been implemented.
Limited support for writeable static data has been added, needed by core subsystems such as Java, but is not available for application use.
ROMs may be constructed which support dual kernels, so that a single image can be tested on two CPU/ASSP variants (though not two instruction set architectures).
Many toolchain changes have been made to support the above. Most `makmake` and `rombuild` project files must be modified as a result.

8.5.3.1.1 E32

8.5.3.1.1.1 User Library and APIs

All user library APIs are backward binary compatible with the user library from previous EPOC releases.

- _LIT provides support for literal descriptors, which is highly recommended in preference to _L
- default screen size KConsFullScreen definition added, to create text consoles filling the entire screen regardless of its size (KConsDefaultScreenSize was good only for 640x240 displays)
- RPointerArray<T> and RArray<T> classes now available, which destroy their objects when Close()d
- CleanupClosePushL() and similar functions are now in E32, rather than in STORE: binary compatibility is maintained, since these functions are inline templates
- TFixedArray<class T, TInt S> provides an effective wrapper on C++ arrays

Text and language support:
- Unicode character tables, collation etc added
- Euro and z-caron characters added.
- more Far Eastern languages added to TLanguage enumeration

Improvements in math functions required to attain full JCK certification (more tightly specified than the previous IEEE754 certification).

8.5.3.1.1.2 Performance

Many optimizations:
- machine-coding many math, memory and other functions
- specialized arrays are used where the generality of CArray-type classes was not needed
- MMU code has been redesigned to reduce context-switching overheads by supporting system processes whose address mapping is fixed with respect to execution context (this reduces the need for cache flushes and PDE moves on context switching)

8.5.3.1.1.3 i/o handling and porting

Kernel changes:
- layering improvements in the kernel for better porting
- support for many different hardware variants

PC Card control code now supports i/o cards as well as memory cards.
Better i/o handling:
- interrupt chaining now supported, many DFC-related improvements
- new power model in which devices report their requirements, and the kernel keeps track of power requirements and power sources, calling power management functions appropriately
- PC Card support has been generalized to cover any peripheral bus; RBusDev classes introduced, eg RBusDevComm which replaces RDevComm. PBus code separated into own DLL to allow machines without a peripheral bus not to load it.
- changed UIDs on all device drivers because of the resulting incompatibilities

8.5.3.1.2 F32

8.5.3.1.2.1 User API

- RFs::NotifyChange() can now notify when changes occur in a directory, a file, a directory and all subdirectories, or a directory across a number of drives (eg `?:\system\apps\` and subdirectories), and several other combinations. Several notification requests may be issued by a single client. The corresponding cancel API may cancel all or only selected notifications.
- RFs::ReadFileSection() reads bytes from a file without opening it and without checking locks. This function allows rapid heuristic identification of file types, for the

APPARC recognizer system. Its results are not guaranteed to be valid if the file is open. However, APPARC usually either already knows the type of open files, or does not care.

- RFs::IsValidName() has been overloaded to provide a variant which returns the first invalid character in a name

8.5.3.1.2.2 Provider API

- TLocalDrive has been replaced by TBusLocalDrive, and the capabilities information TLocalDriveCaps by a package buffer template, in line with standard EPOC patterns

8.5.3.1.2.3 File formats

UTC is now used for file timestamps, rather than local time, to improve synchronization.

8.5.3.1.2.4 Program loading

Better support for executing programs:

- many improvements in loading combinations of `.exe`s and DLLs into RAM.
- pre-loaded DLL list, prevents RAM-based DLLs from overriding ROM-based DLLs, intended for use when RAM-based DLLs may be "inherited" from an EPOC R3 machine. The support has been implemented in such a way as to prevent any further override of "pre-loaded" ROM DLLs. The list of pre-loaded DLLs is maintained in `\System\Data\DLL Preload List`.
- performance improvements
- new RLibrary::FileName() function returns the the full path and filename of a DLL (the only exception is that those loaded statically from ROM are returned as `z:dllname.dll` without a path).
- loader keeps file-server session open continually

A safe reset option has been added to force the window server to be loaded only from ROM, useful on devices in which `c:` persists even across cold resets (eg those with flash RAM).

8.5.3.1.2.5 Text shell

Some usability improvements:

- a batch language has been added
- case is now preserved on all command lines (instead of their being forced to upper-case)

8.5.3.1.3 Emulator and WINC

Improvements to the emulator including:

- allowing the emulator to run from any directory, (not just `\epoc32\release\wins\deb\` and its peers, together with `\epoc32\data\` and `\epoc32\wins\c\`): this means that many emulators can be installed onto a single drive
- there is generally more versatility in mapping from EPOC drives to Windows directories
- console implementation has been changed so that console-based programs may run under either a text shell or a GUI environment, without any configuration change in the emulator
- command-line switches to select fascia, initialization file, and drive mapping properties
- comms drivers deadlock less frequently, and sound drivers take full advantage of PC soundcard support (rather than the piezo speaker used for beeps)
- alt+F4 can be used to close the window
- applications can now run from emulated `c:` or `d:` as well as `z:`
- `c:` drive can be specified using command-line arguments, or set to the system temp path so the emulator can be booted from a read-only device such as a CD-ROM

- _EPOC_DRIVE_ statement in `.ini` file maps drives additional to `c:`, `d:` and `z:`
- better support for just-in-time debugging of EPOC panics
- international keyboard variants are now supported (previously, US and UK keyboards were mapped correctly; everything else was mapped to UK)
- fascia bitmap filename must match `.ini` filename
- other improvements to path handling
- the graphics emulator has been renamed from `wsexe.exe` to `epoc.exe` (strictly a WSERV change rather than E32)

Improvements and fixes to WINC including
- in narrow builds only, it is possible to redirect console output to stdout
- can write to `z:\` instead of treating it as read-only (a hangover from WINS)
- default directory is now the current Windows directory on invocation, not `c:\`: this makes writing DOS batch files to call WINC utilities very much more straightforward (another hangover from WINS)
- no memory limits (yet another hangover from WINS)

The emulator's new flexible directory structure enables you to run the emulator (and applications built for it) from any directory. However, these changes are not yet reflected in the C++ toolset, so that the C++ SDK must still be installed using the old directory structure.

8.5.3.1.4 Tools

Wide-ranging changes to `makmake` to support new system facilities including:
- deb/rel interoperability: `.def` files are now identical for both debug and release builds, and hence no longer have `-d` suffixes for debug builds
- support for Unicode projects: `-u` flag passed to `rcomp`, UNICODEUID statement added
- AUTOUID processing is now compulsory, therefore old #pragma data_seg directives must be removed from C++ source files
- __SYMBIAN32__ macro is now added to all builds
- EXEDLL processing is improved
- support for limited writeable static added for restricted purposes
- support for more "EPOC" targets — ie, those running native on hardware, rather than under an OS such as Win32 — than MARM, and consequent renaming of some statements eg STACKSIZE which were only allowed within START MARM blocks, to EPOCSTACKSIZE which are allowed anywhere (but are ignored for non-machine builds).
- EPOCPROCESSPRIORITY and EPOCFIXEDPROCESS parameters now allowed for `.exe`s (they are passed through to `petran`), for OEM use
- source directory now searched for user include files
- FIRSTOBJECT allows `eexe` or `edll` to be replaced by any specified object file
- support for app, LDD, PDD, FSY by creating `.def` file automatically
- multiple resource file support through LANG, RESOURCE and SYSTEMRESOURCE statements

As a result, all EPOC `.mmp` files must be changed. The impact on applications is lowest (mainly AUTOUID), while that on some kernel-side or system objects is higher.
`evalid` tool added to compare EPOC binaries — executables, DLLs, resource files etc — to check certain types of build compatibility. `evalid` runs on Windows NT only.
`rombuild` and `petran`, as well as `makmake`, have been altered to support dual-variant ROMs, fixed processes and writeable static data, and some tidying up. As a result, `.oby` files used for ROM building have changed substantially.

8.5.3.2 Engine Support

The most significant changes to engine support components are the support of Unicode, the application architecture's support of non-EPOC documents identified by MIME type, the client-server implementation of the DBMS, and multiple process support in the C Standard Library.

8.5.3.2.1 Unicode support

The resource compiler now supports source files in a variety of code pages (including Shift-JIS), and generated resource files in 8-bit or Unicode character sets. However, the resource compiler's output generates only 8-bit characters, with high bits padded to 0x00.
A new front-end processor (FEP) architecture, represented by the FEPBASE component, allows entry using methods other than standard keyboard and CONE controls — without changing existing application code.
A broadcast server, BRDCST, has been designed to send FEP-related messages to client applications.
ETEXT supports entry using front-end processors.
CHARCONV is a new component which supports character conversions between code pages, with sufficient generality for Unicode requirements.
In Unicode builds, the server side of LEXICON is essentially unchanged, since spell checks are only supported for alphabetical languages. The client side has been changed to convert 16-bit characters down to 8-bit for handling by the server.

8.5.3.2.2 Application Architecture

In EPOC R3, the application architecture can only launch a document if it is of a recognized EPOC file format. This format relies on recognition by UID and a stream-store-based data structure, which allows for lean-and-mean embedded objects. In EPOC R5, document embedding is still supported and requires native EPOC file formats.
In EPOC R3, non-EPOC files (such as plain text) had to be imported into a running EPOC application, by a `File | More | Import` menu item. In EPOC R5, non-EPOC files can be recognized by the application architecture, and associated with one or more MIME types, which are in turn used to associate an application with the file format.
A converter architecture has also been added to provide for conversion between native and non-native formats. A library of standard converters is also provided, including

- ASCII text to ETEXT internal representation and back
- MIME quoted-printable encoding and decoding
- base-64 encoding and decoding
- plain text to embeddable EPOC Word document stream and back

`aiftool` has been modified to allow the mappings between file formats and applications to be specified. APPARC has been modified to provide the relevant support. New components CONARC and EMIME provide the basic framework for converters and MIME type recognition.
The application architecture server encapsulates the programming interfaces and should now be used to broker all application launching — whether by the EPOC Shell, by the GUI launching embedded objects, or by applications such as e-mail and web clients launching other documents.

8.5.3.2.3 DBMS

DBMS has many improvements. The client API has been enhanced, in an backward binary compatible manner.
The most significant enhancement is a new DBMS server which can own databases on behalf of multiple clients, thereby enabling well-controlled sharing while the database is open.
A new database factory provides the ability to open a database by name: either for fast, exclusive, client-side access or for sharing with other programs using the server. The former is much like the earlier store-database but without direct access to the store object, while the latter supports transaction locking, and a notification interface indicating when (for instance) locks have been released.

The DBMS supports compression, so that read-only compressed databases can be constructed, typicaly 70% smaller than without compression. This facility is used for help databases and is supported by `aleppo`, the EPOC help builder.
Column types:

- column length can be increased dynamically
- Unicode text column types are now fully supported
- only long binary must be used to storing embedded stores (not long text)

SQL has been improved:

- ORDER BY no longer requires an index (and is often faster without one!)
- queries are optimized by the DBMS so that they are evaluated in the most effective manner
- a full range of DML SQL has been added
- DDL and DML operations may be mixed in a single transaction

Encryption is supported, using E32's CSecurityBase-provided features, and support newly added to the stream store. The stream store has new encryption support for page-pools, which are used by B+trees, and performance improvements in the encryption filters.

8.5.3.2.4 C Standard library

The C Standard Library now supports popen3() for process launching with pipes, and associated calls such as waitpid(), wait() and system().
The standard library is integrated better with Windows for command-line tools, including the ability to attach pipes from Windows, and to parse Windows command lines.
The standard library has been aggressively tested in a much wider variety of contexts, and is now much more robust.

8.5.3.3 Graphics

The major new functionality in the EPOC R5 graphics system is colour support. Prior to EPOC R5, TRgb values supported colour, and settings were honoured by printer drivers. Now, EPOC's raster graphics components support colour, and a logical colour mechanism has been built into the EIKON GUI.
An open font system has been implemented, which allows vector fonts such as Truetype to be implemented by adding in new rasterizers. Printer drivers also support rasterizers as well as built-in printer fonts. A rasterizer, compatible with Truetype, has been written to demonstrate this framework, but is not presently available for commercial release.
Unicode-related work has impacted graphics. The front-end processor (FEP) framework and Open Font system were strongly motivated by Unicode, though they also have other applications. Text wrapping logic has been separated from the main FORM DLL in order to allow non-Western wrapping requirements to be supported.
The font and bitmap server has been made a fixed process, to improve performance.
Display drivers have been improved to use displays of arbitrary size.
The window server supports power-off notification, hollow-rectangle cursors, fading (intended to diminish the brightness of non-foreground windows), dynamic screen size changes. The window server is also now a fixed process.
CONE has been enhanced by FEP and colour support. Other detailed enhancements include ability to set whether a control takes focus or not dynamically (not just when a control is created), ability to prevent CONE flushing the window server buffer when waiting for a non-window-server event, support for swapping GCs while printing, ability to create a control on a backed-up window, and reference-counted resource file management which allows a resource file to be added multiple times (and then deleted a matching number of times, only disappearing when the reference count reaches zero). All EPOC fonts now include the Euro symbol at 0x80.
Various printer driver issues have been fixed, and a Canon BJ printer driver has been added.
Many fixes have been applied to FORM, the text view. FORM also supports the new hollow-rectangle cursor type.

8.5.3.3.1 Colour support

8.5.3.3.1.1 GDI, FBSERV and BITGDI

The TDisplayMode enum has been enhanced to include EColor4k display mode so that EPOC now supports 2, 4, 16 and 256-level greyscale displays, and 16, 256, 4k, 64k and 16 million colour displays. The font and bitmap server FBSERV has been enhanced to support bitmaps in these display modes, the BITGDI has been enhanced to draw to them, and the `bmconv` tool has been enhanced to produce bitmaps in them all. Given a TDisplayMode, the TDisplayModeUtils class returns whether a display mode is colour or not, and how many colours/grey values it supports.
EPOC constrains the physical colours supported by its colour display modes as follows:

- 16-colour displays use the EGA colour set: black, white, and then both light and dark versions of grey, red, green, blue, cyan, magenta and yellow
- 256-colour displays support 216 colours made up of 6x6x6 RGB values, each containing all possible multiples of 51 for R,G,B values. Additionally, all remaining 10 shades of pure red, green, blue and grey are represented, by adding all remaining multiples of 17. This use of 256 colours is sometimes known as the Netscape colour cube.
- 4096-colour displays effectively support RGB values with 4 bits per primary colour
- 64k-colour displays effectively support RGB values with 5 bits allocated to red, 6 to green and 5 to blue
- 16M-colour displays support true colour with 8 bits allocated to each primary colour

This means that, in each display mode, a unique index can represent the physical colours supported and can be mapped onto a full RGB value. The TRgb class has been augmented to include this mapping function and also a mapping function from RGB to the nearest physical colour. With this design, EPOC does not need to support hardware palettes with their associated hardware overheads, programming complexities, and occasionally strange visual effects.

8.5.3.3.1.2 Window server

The window server has been enhanced to support the requirements of machines with colour displays with different modes, windows with different display requirements, and backwards compatibility.
A `wsini.ini` file specification indicates to the window server the default display mode to use for new windows: this need not be specified on machines which support only one display mode, but most OEMs would wish to specify it on hardware with multiple display modes. On the PC-based emulator, it should be chosen to emulate the required hardware.
Windows are given a default display mode on creation, which may be altered throughout the window's lifetime (except backed-up windows: their display mode is specified on creation and cannot be changed). The window server optimizes the display mode in use to be the least demanding (in terms of memory and power) for all of the windows currently displayed.
When using off-screen bitmaps, best performance is obtained by creating the bitmap in the same mode as the window to which it will eventually be blitted. You should use the RWindowBase::DisplayMode() function to determine this. Legacy code, written without this in mind, will function correctly, albeit slightly slower because of bitmap format conversions during blitting.

8.5.3.3.1.3 EIKON

Prior to EPOC R5, EIKON and applications used hard-coded greyscale values for all their drawing. This is still appropriate for some purposes (eg, rendering pictures), but for UI element drawing colour creates much more flexibility, and it is better to write drawing code in terms of *logical colours* (eg dialog title bar background) rather than RGB colours (eg light grey, or dark cyan). EIKON supports this requirement with a kind of palette known as a *colour list*, to support this. The colour list

- supports logical-to-RGB colour mappings loaded from resource files or specified programmatically

- supports sections for EIKON and for applications, independently: a section is identified by application (or EIKON) UID, and a logical colour by an enumerated constant
- supports mappings for both four-greyscale and sixteen-colour colour schemes: the sixteen-colour scheme will be used if the screen mode supports 16 or more colours, otherwise the four-grey scheme will be used

These mappings may be overridden on a per-control basis using functions in CCoeControl.
EIKON's "grey selector", CEikGrayPalette, allows the user to pick colours. It is now augmented to support colour as well as grey selection — but the name remains unchanged.
A TEikColorUtils class provides static functions for colour brightness adjustment.

8.5.3.3.1.4 Tools and utilities

The bitmap converter `bmconv` has been updated to support conversion between Windows `.bmp`s and EPOC multi-bitmap files (`.mbm`s) in any display mode.
When developing an application intended for use on both a greyscale and colour device, it is very important that the colour schemes for the application, its icons and other bitmaps be tailored for the specific device. A bitmap that looks great in 16- or 256-colour mode will be converted to greys if displayed on a greyscale display, but may be aesthetically quite unusable.
It is therefore recommended that application developers produce application information files and multi-bitmap files in both colour and greyscale variants and test them both on the emulator. You are recommended to deliver AIFs as both `.abw` and `.acl`, and MBMs as both `.mbw` and `.mcl`. Then, use the `instcol` tool to copy the intended variants to `.aif` and `.mbm` respectively.

8.5.3.3.2 Unicode support

CONE supports reading both 8- and 16-bit resources — in narrow as well as wide builds.
CONE supports front-end-processors (FEPs), which may be used transparently with respect to application programs. The FEP framework is designed to support input of characters from large Far Eastern character sets by methods which, by now, have been established in the IT industries in the relevant locales. The FEP framework could also be used for handwriting recognition in Western alphabets.
Various Unicode fonts have been tested. Unicode was a primary motivator to add vector font support to EPOC.
FORM has been enhanced to support more general line-wrapping rules, and split into `form.dll` and `wrap.dll`. In non-Western configurations, the `wrap.dll` is replaced by DLLs implementing the rules for Chinese, Japanese etc.

8.5.3.4 System and shell

8.5.3.4.1 EIKON

Many fixes, usability and API improvements. The major work on EIKON, as described in §3.3.1.3, is the addition of support for colour.
API improvements:

- an extra CColumnListDrawer function returns the width of items in pixels.
- a new CUnifiedFileSelector class provides support for browsing files across drives, eg all files in `?:\system\apps\`
- choice lists support font override
- CEikEdwin can return average values for lines/edit rectangle, and characters/line, for better approximate scrollbar sizing
- in preparation for Unicode, CEikAppUi::ProcessCommandParametersL() now takes a (binary) TDesC8 parameter, rather than a (text) TDesC parameter
- CEikTelephoneNumberEditor moved from EIKCNPNL into EIKON proper

Internal structure improvements:

- the EIKON console uses a separate heap, which makes debugging easier since, previously, any EIKON console program in debug mode would fail on exit with heap imbalance

- the EIKON console has been renamed `econseik.dll`, and co-resides happily with the E32 console, `econs.dll`. This removes the need to move or rename files when changing between text and graphics console-based applications in the emulator.
- source code is managed differently: utility programs reside in EIKONEX, and test code in EIKTEST top-level projects
- backed-up windows are always used in systems without an active scheduler, to support new OPL-related requirements
- in preparation for Unicode, EIKON builds on CONE's support for the front-end-processor (FEP) framework

Functionality improvements:

- IrDA will now beam multiple files/folders: this is delivered to the user through better Shell functionality.
- the task list is significantly faster.
- scrolling menu panes are supported (for Java compatibility: they are deprecated for C++ or OPL use, or even Java use when an application is designed specifically for EPOC)
- thanks to a reconfiguration of the APPARC server in the EIKON server, and new APPARC APIs, EPOC Connect can now restart applications when it stops them for backup purposes

A new EIKCNPNL project contains control panel extensions for dialling, network and modems. It also delivers `eikxtra.dll`, containing an IP address editor, and a better secret editor than that in `eikon.dll`.

8.5.3.4.2 Shell

Many usability improvements including:

- ability to send multiple files and folders using IR to other EPOC machines
- much faster control panel launch, and other performance enhancements
- ctrl+Z to close a folder while shell browsing
- ctrl+shift+J to display a recently-used file list
- displays icons for, and can launch, non-EPOC file formats (eg, text is associated with either a text editor or, if one is not available, with the Web browser)
- the fixed task bar has been reconfigured to give more prominence to the Contacts and Jotter applications
- background wallpaper can be set by the user

The application installer now reads information from installed `.sis` files on the EPOC ROM (drive `z:\`): this prevents the installer from installing older versions of software than those already in the ROM.

8.5.3.4.3 Tools

`eikrs`'s syntax has been enhanced to distinguish between narrow or Unicode operation.

9. EPOC Software Development

There are two EPOC development options available, each with its own Software Development Kit (SDK). More comprehensive information, and how to obtain the SDKs, is available through:
http://developer.epocworld.com/

9.1 C++ SDK

The C++ Software Development Kit includes all user-side class libraries, the WINS emulation in several configurations, tools for project control, resource file compilation, bitmap and sound manipulation etc, plus documentation and example code. The source code to certain system components is also supplied.
Object orientation results in tightly interwoven class relationships; in turn, the documentation needs to be tightly interwoven also. The documentation is delivered in HTML, and includes many navigational aids including links to class definitions from all their references.
System Requirements:

- a high-spec IBM compatible PC.
- Microsoft Windows NT 4.0, or Microsoft Windows 95, and disks formatted using NTFS or VFAT.
- A Web browser, such as Netscape 2.0 or later, or Microsoft Internet Explorer 3.0 or later, which supports long filenames, tables and frames.
- Either Microsoft Visual C++ 4.0, 4.1 or 4.2; Standard, Professional or Enterprise Edition or Microsoft Visual C++ 5.0; Learning, Professional or Enterprise Edition

9.2 OPL SDK

OPL is a BASIC-like language that was developed by Psion in the mid 1980's for programming its Organiser range of handheld computers. On these machines, OPL was the only alternative to assembler language.
OPL is also available for EPOC16. OPL allowed rapid application development, and even programming directly on a Series 3 without using another PC. This ease of use, and universality, fostered a large market in personal, professional and industrial applications written in OPL.
The same facility is available in EPOC. OPL programs written for EPOC16 need only slight modification to run under EPOC. The OPL language gives access to many EPOC facilities. Any facility available to a C++ program may be made available to an OPL program through an appropriate OPL extension, or OPX. Since OPXs are written in C++, the C++ SDK is needed to develop OPXs. An OPL programmer does not need the C++ SDK in order to use OPXs.
The OPL SDK supports more comfortable development of OPL programs on a PC. It includes the WINS emulation, plus documentation, tools and examples not found on EPOC machines. OPL programs written for EPOC16 need only slight modification to run under EPOC and a conversion guide is supplied.
System Requirements:

- an IBM compatible PC with 25MB of free disk space
- Microsoft Windows NT 4.0, or Microsoft Windows 95, and disks formatted using NTFS or VFAT
- a Web browser, such as Netscape 2.0 or later, or Microsoft Internet Explorer 3.0 or later, which supports long filenames, tables and frames
- A word processor that can read files in RTF format

For more information on OPL please visit:
http://mobile.ericsson.com

10. Infrared communication

10.1 General Features of Infrared Communication

Infrared communication creates a data link between two communications devices through an infrared beam of light. On the Ericsson Mobile Companion, this link can be used to communicate with desktop computers, printers, other Ericsson Mobile Companions, Ericsson Mobile Phones, cameras and all other hardware supporting the standard. Infrared Data Association (IrDA) set the hardware and software standards that form the infrared communication links. See the relevant chapter.

Ericsson has used this technology to create a truly unique wireless communication solution for the Ericsson Mobile Companion and Ericsson Mobile Phones. The IrDA infrared data link secures the connection between them. This wireless communication solution is compatible with all Ericsson GSM 900/1800/1900 Mobile Phones in the 600, 700 and 800 series and it consists of the Ericsson Infrared Modem and the MC218's Infrared Port.

10.2 Key benefits

- True wireless communication.
- Secure data transmission with the IrDA–SIR standard.
- You can send and receive e-mail, faxes and SMS.
- Connection to the Internet as well as your local network.
- You can manage your Phone book.

10.3 Ericsson Mobile Companion

The Ericsson Infrared Communications Solution
Ericsson has used the IrDA-SIR Data Link Standards to create a unique wireless communications solution for the Ericsson Mobile Companion and Ericsson Mobile Phones. The IrDA Infrared data link secures the connection between the Ericsson Mobile Companion and the Ericsson Mobile Phone without the use of cables.

This communication solution consists of:
- Ericsson Infrared Modem
- Infrared Port

The IrDA Infrared data link is compatible with all Ericsson GSM 900/1800/1900 Mobile Phones in the 600, 700 and 800 series. One of the IrDA data link's biggest advantages is its low power consumption. That is why infrared transmissions are well suited for Ericsson Mobile Phones. The Ericsson Mobile Companion is capable of bitrates up to 115 kbps.

10.4 Ericsson Infrared Modem

Included with the Ericsson Mobile Companion is the Ericsson Infrared Modem. The Ericsson Infrared Modem is the adaptor that you attach to the Ericsson Mobile Phone when you want to communicate between the Ericsson Mobile Companion and your mobile phone. The Infrared Modem is a unique solution, which takes advantage of IrDA technology for true wireless communication between an Ericsson Mobile Companion and an Ericsson Mobile Phone. What earlier required a PC-Card GSM modem and a cable between the Mobile Companion and the

Ericsson Mobile Phone, now fits into a small device attached to the mobile phone. Two major advantages of the Ericsson Infrared Modem are its low power consumption and compact size.

10.4.1 Contents of Ericsson Infrared Modem

The Ericsson Infrared Modem consists of three parts: the data plug that connects to the mobile phone's system connector, the built-in GSM modem, and the IrDA-compatible infrared port that replaces the cable found on other GSM modem solutions. Since it uses the IrDA-SIR protocol to communicate, it is compatible with all devices equipped with IrDA-compatible IR ports.

10.4.2Power consumption

When using the Ericsson Mobile Phone with the Ericsson Infrared Modem attached the power consumption increases by about 10%.

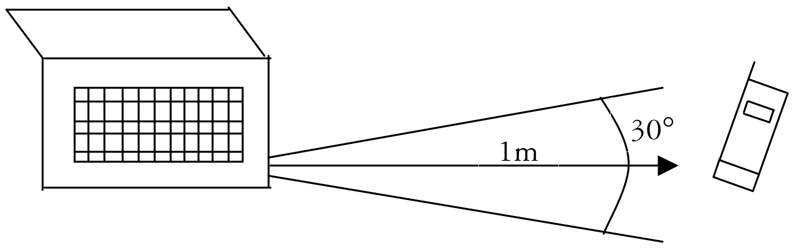
10.4.3 Size

The Ericsson Infrared Modem adaptor is so small (23x46x14 mm) and light (9.7 grams) that it does not interfere with an Ericsson Mobile Phone's flip, nor with audio performance or functionality.

10.4.4Infrared Port

IrDA is a point-to-point communication link between two infrared ports. The Infrared beam has to be directed towards the target Infrared Port and as long as the two infrared ports are within sight and range, the devices can exchange data. For the best performance, place the Ericsson Mobile Phone with the Ericsson Infrared Modem attached range of a metre and at a 30° angle from the Ericsson Mobile Companion's Infrared Port. One advantage of this Infrared gap is that the risk of transmitting data to other devices nearby is minimized.

10.4.5 Connection



The infrared link is a serial connection, which means that the data bits are sent one after another in a long stream. The IrDA-SIR Data Link Standard is a protocol that makes transmission of data faultless. The IrDA-SIR standard provides a high level of noise immunity, which means that it is not sensitive to fluorescent light, sunlight and electromagnetic fields. This makes it suitable for a modern office environment.

10.5 IrDA – Infrared Data Association

In 1993, a group of companies founded theInfrared Data Association (IrDA). As a leading high technology standards association, IrDA is committed to developing and promoting infrared standards for the hardware, software, systems, components, peripherals, communications, and consumer markets and Ericsson is one of more than 160 organizations in the IrDA.
More information can be found at http://www.IrDA.org .

IrDA has developed the IrDA-SIR (Infrared Data Association-Serial InfraRed) Data Link Standards for transmitting data via infrared light waves, and it is used for serial communication.

11. IrDA

11.1 IrDA's New Full Range of Digital Information Exchange via Cordless Infrared Connections

Regarding present publications on IrDA features for PC99, IrDA Data is recommended for high speed short range, line of sight, point-to-point cordless data transfer - suitable for HPCs, digital cameras, handheld data collection devices, etc. If IrDA is supported, it must be targeted at the 4 Mb/s components. IrDA Control is recommended for in-room cordless peripherals to hostPC. PC99 for lower speed, full cross range, point-to-point or to-multipoint cordless controller - suitable for keyboards (1way), joysticks (2 way & low latency) etc. IrDA Data and IrDA Control require designer attention to ensure spatial or time sharing techniques (Appendix E) so as to avoid interference.
Since 1994"IrDA DATA" has defined a standard for an interoperable universal two way cordless infrared light transmission data port and now appears in approximately 50 million electronic devices including desktop, notebook and palm PCs, printers, digital cameras, public phones/kiosks, cellular phones, pagers and other mobile devices. Adaptors now include the traditional upgrades to serial and parallel ports, plus Universal Serial Bus (USB) and Ethernet/Token Ring LAN point to point access.
IrDA Data Protocols consist of a mandatory set of protocols and a set of optional protocols. The mandatory protocols are listed below.

- PHY (Physical Signaling Layer)
- IrLAP (Link Access Protocol)
- IrLMP (Link Management Protocol and Information Access Service (IAS))

Characteristics of Physical IrDA Data Signalling

- Range: Continuous operation from contact to at least 1 metre (typically 2 metres can be reached). A low power version relaxes the range objective for operation from contact through at least 20 cm between low power devices and 30 cm between low power and standard power devices. This implementation affords 10 times less power consumption. These parameters are termed the required maximum ranges by certain classes of IrDA featured devices and sets the end user expectation for discovery, recognition and performance.
- Bi-directional communication is the bases of all specifications
- Data transmission from 9600 b/s with primary speed/cost steps of 115 kb/s and maximum speed up to 4 Mb/s
- Data packets are protected using a CRC (CRC-16 for speeds up to 1.152Mb/s and CRC-32 at 4 Mb/s).

Characteristics of IrDA Link Access Protocol (IrLAP)

- Provides a device-to-device connection for the reliable, ordered transfer of data.
- Device discover procedures.
- Handles hidden nodes.

Characteristics of IrDA Link Management Protocol (IrLMP)

- Provides multiplexing of the IrLAP layer. Multiple channels above an IrLAP connection.
- Provides protocol and service discovery via the Information Access Service (IAS).

Optional IrDA Data Protocols

- Tiny TP - provides flow control on IrLMP connections with an optional Segmentation and Reassembly service.
- IrCOMM - provides COM (serial and parallel) port emulation for legacy COM applications, printing and modem devices.
- IrOBEX - provides object exchange services similar to HTTP.

- IrDA Lite - provides methods of reducing the size of IrDA code while maintaining compatibility with full implementations.
- IrTranP - provides image exchange protocol used in Digital Image capture devices/cameras.
- IrMC - specifications on how mobile telephony and communication devices can exchange information. This includes phonebook, calendar, and message data. Also how call control and real-time voice are handled (RTCON) calendar.
- IrLAN - Describes a protocol used to support IR wireless access to local area networks.
- Each of the protocol layers and standards mentioned above can be found on the IrDA FTP site as separate documents.

11.2 **IrDA DATA - Hardware/Protocol Stacks**

IrTran-P	IrObex	IrLan	IrCom	IrMC

LM-IAS	Tiny Transport Protocol - Tiny TP

Ir Link Mgmt - MUX - IrLMP

Ir Link Access Protocol - IrLAP

Async Serial-IR 9600-115.2kb/s	Sync Serial-IR 1.152Mb/s	Sync 4PPM 4Mb/s

11.3 **IrDA CONTROL**

IrDA Control is an infrared communication standard that allows cordless peripherals such as keyboards, mice, game pads, joysticks and pointing devices units to interact with many types of intelligent host devices. Host devices include PCs, home appliances, game machines and television/web set top boxes. IrDA Control is well suited to deal with devices that leverage the USB HID class of device controls and home appliances. It is a sophisticated implementation guideline for bi-directional remote control with MAC enumeration and binding and with LLC transactions. IrDA Control Protocols consist of a mandatory set of protocols.

- PHY (Physical layer)
- MAC (Media Access Control)
- LLC (Logical Link Control)

Characteristics of IrDA Control Physical Signalling:

- Distance and range equivalent current uni-directional infrared remote control units (minimum 5-metre range).
- Bi-directional communication is the basis of all specs.
- Data transmission at 75 kb/s at the top end
- The data is coded using a 16-Pulse Sequence multiplied by a 1.5 MHz subcarrier which is allocated for high speed remote control in IEC 1603-1 although this base band scheme has harmonics which can intrude upon other IEC bands.
- Data packets are protected with a CRC (CRC-8 for short packets and CRC-16 for long packets). The physical layer is optimized for low power usage and can be implemented with low-cost hardware.

Characteristics of IrDA Control MAC:

- Enables a host device to communicate with multiple peripheral devices (1:n) and up to 8 peripherals simultaneously.
- Ensures fast response time (13.8 ms basic polling rate) and low latency.

Asymmetric MAC:

- Provides for dynamic assignment and re-use of peripheral addresses.
- Scheduling of media access is actually buried in the HID LLC.
- Characteristics of the IrDA Control LLC:
- Provides reliability features that provide data sequencing and retransmission when errors are detected.
- Works with an HID-IrDA Control Bridge to enable the link control functions of USB-HID.

All required and optional layers of the IrDA Data and IrDA Control specifications are described in specifications, which you may download at no charge from the IrDA Web site www.irda.org.
Interop product registration is strongly advised on this site.
Infrared Data Assn IrDA specifications are now supported by all divisions of Microsoft (IDG, WinCE, WinNT and Win98) and this universal data port is recommended on PC 99 products (mandated on certain WinCE products - PalmPC, etc.).
IrDA Feature Trademark Licensing Program
IrDA has developed a Feature Trademark Licencing program for products from members and non-members with use of the IrDA "Beaming IR" mark.

IrDA wants to ensure a successful experience for end users using their devices with IrDA infrared features. Need to know how infrared works?
This section includes:

- How it works - Information on how the technology works in end user products
- Interoperability - Information on how products can work together
- FAQ - Frequently asked questions
- Ir Market Statistics - Information regarding market statistics or Ir market updates.

12. GSM Data

12.1 General Features of GSM Data

To put it simply GSM Data is the ability to send data or fax information over the GSM (Global System for Mobile communication) network. Data over GSM offers new opportunities for both GSM network operators and mobile workers alike. By using GSM to send and receive data or faxes, mobile employees have access to a level of geographic mobility previously unobtainable.

The major benefit of using GSM data, is that you can use it from any location at any time. There is no need to rely on being able to gain access to a Public Switching Telephone Network (PSTN) line, or wasting time trying to find one. Using GSM provides a faster response time for mobile employees. This is becoming an increasingly important factor in countries where the PSTN is underdeveloped, but where GSM is widely available, such as Central and Eastern Europe. Using GSM in a foreign country can also be a more cost-effective solution for mobile data than the fixed network. For instance when staying in hotels, calls are charged at a premium rate making data communication excessively expensive. By using GSM users can actually gain in cost advantage by avoiding the premium charged by the hotel.

12.2 Key Benefits

- GSM data can be used from any location at any time.
- Ease of use for international travellers.
- GSM data offers a use anytime, anywhere capability, unmatched by fixed telephone networks.
- The cost of using GSM data is reducing over time

12.3 GSM & GSM Data – An introduction

GSM is the world's most widely deployed digital wireless technology, with networks serving more than 170 million subscribers world wide. There are currently more than 250 networks in over 100 countries/areas around the world. Thanks to the world wide spread of GSM networks, mobile users now have access to a level of geographic mobility previously unobtainable.

GSM offers advanced mobile data capabilities, previously unavailable on analogue cellular networks. Users should select a GSM data solution vendor with developed expertise and experience in GSM data. In that regard Ericsson is an excellent choice. In fact GSM is the ideal communications technology for Ericsson Mobile Companion. Co-operation between the GSM handset manufacturer and data solutions providers is the key to developing reliable, robust products.

12.3.1 Lower power consumption

For Mobile Companion users, battery life is at a premium. Using a software based GSM solution provides substantially improved battery performance for the Mobile Companion compared to using a PC Card.

12.3.2 Security

A company's information is sensitive and hence it is important that the integrity of the information is not compromised. Sending data over the PSTN network in its analogue form is not secure, since the data is not encrypted. Only a very small number of modems are capable of encrypting data, and these carry a significant price premium.

Because GSM is a digital technology, it is inherently more secure. As part of the GSM standard, both data and voice transmission are encrypted for transmission across the network. This has been the key strength for GSM as it provides peace of mind for users, without the need for additional expenditure on expensive hardware and software.

12.3.3 Interworking with ISDN

One of the key advantages that GSM can offer users, is its ability to interwork with ISDN. This is a core feature of the GSM standard, which has been developed to interwork seamlessly with ISDN. Interworking between the ISDN and GSM networks is made possible by using a technique know as rate adaptation. If the ISDN terminal adaptor being called by the GSM terminal is capable of supporting V.110, it can adapt the 9.6 kbps data from the GSM terminal into 64 kbps ISDN data. This is achieved by adding additional bits to the GSM data effectively packing out the data.

12.3.4 Quick call set-up

GSM was developed to interwork with the Integrated Services Digital Network (ISDN). ISDN uses a technique known as rate adaptation to convert the slower GSM data up to 64 kbps ISDN data using the International Telecommunications Union (ITU) V.110 rate adaptation protocol, , which is a standard for ISDN terminals. As a result when a data call is made from GSM handset to an ISDN terminal adapter, the quick call set-up capability of ISDN can be utilised.

GSM Data makes it possible to send data or fax information over the GSM network. When making a data call, the user simply dials the number they wish to send information to. The call is connected to a gateway located at the GSM network operator, and then the gateway takes care of the translation between the different kinds of networks, such as PSTN, ISDN or X.25 networks. Currently the data transmission speed supported is up to 9.6 kbps, but with new GSM technologies being developed this will increase.

12.3.5 Easy to Use

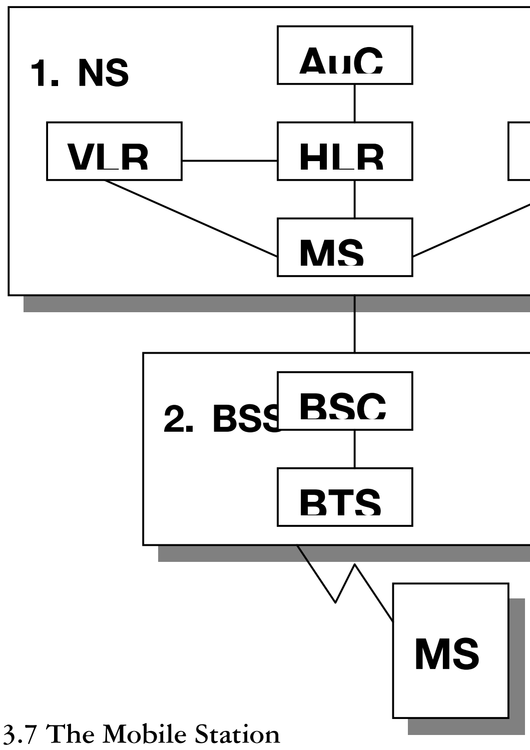
It is easy for international travellers to use GSM Data. The user does not have to worry about selecting the correct telephone connector, ensuring that the modem is approved for connection to the PSTN within that country, or can deal with potential line quality and integrity issues for use over long distances. All this can be side stepped by using GSM, where the user is able to use their equipment in the way they normally would.

Using GSM also provides for a faster response time for mobile employees when travelling. Used in this way, GSM can provide a real competitive advantage to an organization. For example, sales people out in the field can send information to relevant parties immediately. This can allow a company to differentiate itself from its competitors, by its ability to respond to their customers' needs in a time frame that meets (or exceeds) their requirements.

12.3.6 GSM Network

The GSM network can be divided into three broad parts:

- The Mobile Station
- The Base Station Subsystem
- The Network Subsystem



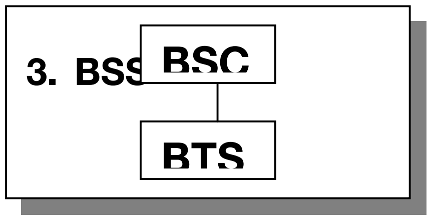
12.3.7 The Mobile Station

The Mobile Station is the Ericsson Mobile Phone, which is carried by the user. To be able to use the phone, the user needs a Subscriber Identification Module, also called a SIM card. The SIM-Card provides personal mobility, letting the user insert his or her SIM card in any GSM phone and still have access to the same subscribed services.

12.3.8 The Base Station Subsystem

The Base Station Subsystem is composed of two parts, the Base Station Controller (BSC), and the Base Transceiver Station (BTS). These two parts communicate across a standardized interface called Abis, and therefore allows components from different suppliers to operate together.

The BTS handles the radio-link protocols with the Mobile Station (the GSM phone). Depending



on the surroundings (urban or country area), several BTSs are required to give good communication coverage for the user.
The BSC controls call set-up, frequency hopping, handovers and the operation of one or more BTS.

12.3.9 The Network System

The Network System (also called the Switching System) consists of five functional entities.

- The Mobile Services Switching Centre (MSC). The MSC switches calls between mobile users and between mobile users and the Public Switching Telecommunication Network (PSTN).
- The Home Location Register (HLR). The HLR contains data on all mobile subscribers registered in its Public Land Mobile Network (PLMN).
- The Visitor Location Register (VLR). The VLR contains data on all subscribers visiting its MSC service area. The HLR and VLR, together with the MSC, provide the call-routing and roaming capabilities of the GSM network.
- The Authentication Centre (AuC). In every subscriber's SIM card, a secret key for authentication and encryption over the radio channel is stored. The AuC is a protected database that stores a copy of the secret key.
- The Equipment Identity Register (EIR). Each Mobile Station is identified by a code called the International Mobile Equipment Identity (IMEI). The EIR contains a list of all valid mobile equipment on the network. If a unit is reported stolen, its IMEI will be marked as invalid and cannot be used on the network anymore.

As the Mobile Station moves between location areas and MSC service areas, data is transferred between the HLR and the VLR. By doing this the GSM network will always know where the Mobile Station is located.

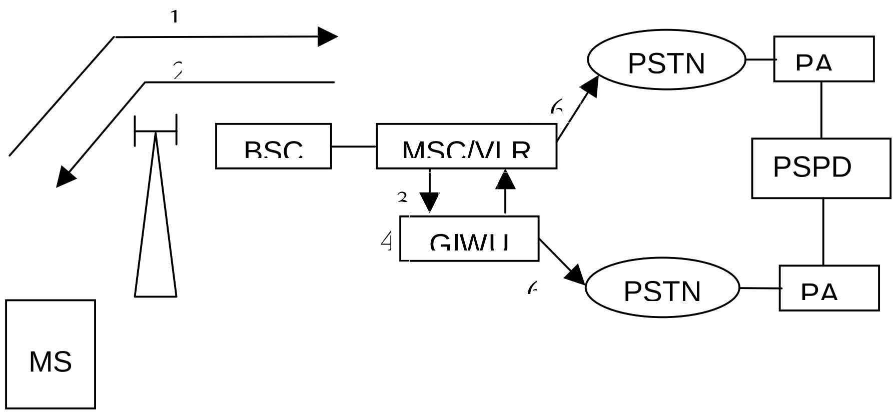
12.4 How does GSM data work?

When a user wishes to make a data call (for more information see paragraph: Data Calls), they simply dial the number they wish to send information to. The GSM network connects the user's call to its GSM Interworking Unit (GIWU), which then completes the call to the remote terminal. In effect the user is dialling the GIWU, and the GIWU is dialling the remote terminal. The GIWU acts as a gateway, translating between the GSM set of protocols, and the protocols used by different types of networks. This allows a GSM network to connect to a range of different network types, such as PSTN, ISDN or X.25 network.

Once a connection between the mobile user and the remote terminal has been established, data can then be transmitted. Data can be sent over the GSM network using one of two modes, these are known as transparent mode and non transparent mode (for more information see paragraphs: Transparent Mode and Non Transparent Mode).

12.4.1 Data calls

To be able to handle data calls, each MSC must have a dedicated Data Transmission Interface (DTI)/GSM Interworking Unit (GIWU). The picture on the next side explains how it works.



1. The mobile station initiates a data call. In the call set-up message Bearer Capability (BC) is included. The BC includes the bearer service type (data, fax) and the requested transmission rate.
2. A connection between the mobile station and the network is set up, as in a normal call, and authentication is performed.
3. The MSC analyses the BC, and then transfers it to the DTI/GIWU.
4. The DTI/GIWU is configured with the rate adaption and the fax/modem service.
5. The DTI/GIWU reroutes the call to the MSC.
6. The MSC routes the call to PSTN or ISDN.

For communication with a Packet Switched Public Data Network (PSPDN), a Packet Assembly Disassembly (PAD) is needed. The PAD transforms the bitstream from an asynchronous terminal to data packages.

12.4.2 Bearer services

The GIWU supports a wide range of bearer services. Synchronous and asynchronous data transmission rates up to 9.6 kbps are supported. Here is a short description of some of the services.

- Traffic to Public Switched Telephony Network (PSTN). Interworking with ISDN or directly to PSTN, a suitable modem is selected in the GIWU.
- Traffic to Integrated Services Digital Network (ISDN). The entire set of data communication services with ISDN terminals is available. Unrestricted digital information is transferred and no modem is necessary.
- Traffic to Packet Switched Public Data Networks (PSPDN). The Packet services supports synchronous data with rates of 1.2, 2.4, 4.8 and 9.6 kbps. Asynchronous data communication between a mobile station and a packet switched network is possible via a PAD facility (see above). Rates between 0.3 and 9.6 kbps are supported.
- Traffic to Circuit Switched Public Data Networks (CSPDN). A CSPDN is reachable via PSTN or ISDN, but depends on the interface between CSPDN and the transit networks

12.4.3 Transparent Mode

In transparent mode, a non-error-corrected connection is established between the user and the remote terminal. Data is then sent synchronously or asynchronously. Because no error correction is taking place, there is little delay (latency) in transmitting data over the link.

12.4.4 Non Transparent Mode

In non-transparent mode, a secure error corrected link is established between the mobile terminal and the remote modem. The connection between the mobile terminal and the GSM network uses the Radio Link Protocol (RLP) for error correction. The GIWU then establishes a connection to the remote modem, implementing the V.42 protocol, the International Telecommunications Union (ITU) standard for error correction.

12.4.5 Fax

Sending a fax is very similar to sending data. To send a fax, an initial call is made to the network. The call is then routed through the network to the GIWU. The GIWU then establishes a connection to the remote fax machine. The fax protocols are than passed end to end between the mobile terminal and the remote fax machine.

12.4.6 SMS Services

The Ericsson Mobile Companion is capable of sending and receiving SMS messages, using the Inbox software. With the Short Message Service, a user can send text messages containing up to 160 characters to and from GSM mobile stations. A Service Centre (SC) acts as a store and forward centre.
SMS consists of two basic services:

- Mobile Originated SMS (from a Mobile Station to a SMS-C)
- Mobile Terminated SMS (from a SMS-C to a Mobile Station)

For Mobile Orginated SMS, an SMS message is sent from a Mobile Station to the SMS-C where it is forwarded to its destination. This can be another Mobile Station, or a terminal in the fixed network.

A Mobile Terminated SMS is when an SMS message is forwarded from the SMS-C to a Mobile Station. When the Mobile Station receives the message, it returns a delivery report saying the transfer was successful.

13. Serial Terminal Support, Comms

Basic serial terminal support is provided by the EPOC Comms application. It is split into an engine and an application.
The application uses EIKON's console support to display the emulator.
The engine supports login scripting, XMODEM and YMODEM file transfer, and emulation of TTY and VT100 terminals.

The language used in scripts is made up of commands, which are very similar to ordinary English words. This section provides an introduction to some important aspects of the scripting language. The following section then guides you through all the script commands.

13.1 How script commands are used

A short script might look like this:

```
STATUS "Uploading test file"
UPLOAD "C:\Documents\anyfile", "Xmodem"
EXIT
```

This script uses three commands: **STATUS**, which displays an information message in the bar at the bottom of the screen; **UPLOAD**, which sends a file to the remote machine, and **EXIT**, which stops the script running, returning control to the Terminal emulation screen.

- Commands can be entered in upper or lower case, or a mixture of both.
- To combine more than one command on a line, separate them with a colon, e.g.

  INFO "Receiving test transmission" : DOWNLOAD "C:\Temp\trashfile" , "XModem" : EXIT
- The total length of a line must not exceed 255 characters.

Note: In general, scripts should finish with a carriage return (i.e. ensure that there is an extra blank line at the end).

13.1.1Labels

Scripts are split up into sections marked by "labels". A label, e.g. 'mylabel:' denotes the start of a section of script. Use labels to split up scripts into manageable sections, and to enable the script to jump directly to a given area by using the **GOTO** command. E.g., in the script

```
GOTO mylabel
        STATUS "Didn't jump to mylabel"

mylabel:
        STATUS "Jumped to mylabel"
```

The first status message is never displayed, since the **GOTO** command jumps straight to 'mylabel:'
Note: To learn how to repeat the instructions given in a label a specific number of times, see the sections on the **SET** and **REPEAT…UNTIL** commands in the glossary of commands later.

13.1.2 End-commands

Some commands, e.g. **SETUP**, **IF** and **QUERY**, require an "end-command" to tell the machine that the instructions they refer to are finished. E.g.

```
SETUP
        BAUD=57600
        PARITY=Even
        HANDSHAKE=XONXOFF
ENDSETUP
```

- Those commands which require ending are indicated in the glossary of script commands later on in this document.

13.1.3 Text strings

Strings are sequences of characters between quotes. The **SEND** command sends a string through the serial port. E.g.

SEND "ATZ"

This sends the characters **A**, **T** and **Z**, a modem initialisation string. You can insert control characters in a string by putting them in angled brackets, e.g.

SEND "guest<013>"

sends a carriage return character after the text "**guest**". Control character codes can also be entered in hexadecimal form, prefixed with a $ sign. E.g.

SEND "bye<$1B>"

Puts the control character code 27 (the 'Esc' control) at the end of the **SEND** string.

13.1.4 Filenames and folders

When receiving a file using the XModem protocol or ASCII, you must specify a name for the file to have on the MC 218. To do this, either

- Simply state the filename, eg.

DOWNLOAD "LetterIV","XModem"

This will place the downloaded file in the Documents folder. Or,

- Specify the full "path name", including the folder and disk, e.g.

DOWNLOAD "C:\Documents\Letters\LetterIV","XModem"

Use this method to download files directly to a memory disk, e.g.

DOWNLOAD "D:\Publications\Frontcover","XModem"

Note: Make sure the name you give to the file you are downloading is not the same as that of another file within the same directory, or you may overwrite the existing files.

When receiving files using the YModem (batch) protocol, the sending machine transmits the filename, so there is no need to specify one. Simply use a "blank text" (i.e. no characters between the quote marks) string in the command, e.g.

DOWNLOAD "","YModem (batch)"

The file will be saved in the Documents folder.

13.1.5 Names

Comms can store information you use frequently, such as passwords or telephone numbers, as "names" in a file. You can then insert a name in a **SEND** or **SENDWAIT** command which Comms will substitute with the stored information.

13.2 To store information as a name

Select '**Set up names**' on the '**Names**' menu in the Terminal emulation screen, and tap '**Add**'. Enter the name, e.g. "pass", then enter the value, e.g. a password.

- The text represented by the name can be up to 64 characters long.
- Names can be up to sixteen characters long, and must start with a letter.

Once you have created a set of names, save them using the '**Save names as**' command on the '**Names**' menu. You can then use '**Load names**' to retrieve the specific set of names that you need at any one time. In general, you should only need to use one names file, even if it contains more names than you will ever use at one time.

- To change the information for a name, use the '**Set up names**' command, and edit the '**Value**' line for the relevant name.

13.2.1Using names in a script

To use named information in a script, you need to instruct the machine to load the correct names file using the loadnames command, e.g.

```
LOADNAMES "D:\Comms\Comms.nam"
```

You can then use the information in this file in either a send or sendwait command. To do this, add a $ sign to the name. E.g., to send the information named "pass1", use the line:

```
SEND pass1$
```

You can send more than one name at once - or mix different kinds of information - by using the & operator. E.g.

```
SEND username$ & pass$ & "mail"
```

- If you alter named information in the course of a connection, or create new named information using, say, the **QUERY** command, you can then use the **SAVENAMES** command to preserve the changes, e.g.

```
SAVENAMES "C:\Comms\Names\BBSnames.nam"
```

13.2.2 Storing name information using the query command

You can create names while the script is running, using the **QUERY** command. E.g.

```
QUERY "Dial-up details"
        "BBS telephone number" , no$
        "Modem initialisition string" , init$
        "Username" , name$
        "Password" , pass$
ENDQUERY
```

prompts the user with a dialog asking for information which is then stored under the names "no", "init", "name" and "pass".

13.2.3 Reserved names

There are three "reserved" (i.e. not free to use for any information) names, which can only be used for special kinds of information. They are:

```
sys_echo$
sys_incoming_crlf$
sys_outgoing_crlf$
```

These are essentially the scripting language equivalents of the '**Translate codes**' options on the '**Tools**' menu in the Terminal emulation screen. They toggle the "local echo" feature, and control whether or not a line feed instruction is added to each incoming and outgoing carriage return.
The line

```
sys_echo$=1
```

switches the local echo function on, which allows you to see the characters you are sending on the home screen. Setting a value of 0 switches it off.
The line

```
sys_incoming_crlf$=1
```

tells the MC 218 to start a new line each time the remote machine sends a carriage return. Setting a value of 0 switches it off.
The line

```
sys_outgoing_crlf$=1
```

tells the remote machine to start a new line each time you send a carriage return from the MC 218. Setting a value of 0 switches it off.

13.2.4 Password protecting names

You may want to protect confidential information stored as a name by giving the names file a password. To do this:

1. Select '**Password**' on the '**Names**' menu, then enter and confirm your password.
2. Select '**Save names**' on the '**Names**' menu. You will now have to enter the password to access the '**Names**' dialog.

- **To clear a password:** select '**Password**', enter the current password, then enter a "zero-length" password in the '**Set password**' dialog.

Note: You can still run scripts that use names without entering the password. If you want to be prompted every time a script that uses password protected names runs, use the **LOADNAMES** command. E.g.

```
LOADNAMES "C:\BBSstuff\Comms.nam"
SEND init$ & phone$
```

will prompt for a password if the file being loaded is password-protected.

13.2.5 Variables

A variable is a name which can be used to store and represent different information. E.g. **username$** could be used to represent the information "Anton", "Sam" or "5" etc, depending on what has been stored under that name. There are two types of variables in the scripting language:

- **Non-persistent variables:** these have no $ after them, e.g. "**phoneno**". Information stored in this kind of variable is lost when the script stops running.

- **Persistent variables:** these are followed by a $. Information stored in this kind of variable can still be edited and saved once the script has stopped running, using the '**Set up names**' and '**Save names as**' commands on the '**Names**' menu.

13.3 Script commands

This section provides a summary of the script commands according to type, followed by a detailed glossary of commands and their usage.

13.3.1 Summary of the script commands

13.3.1.1Port and handshaking control

CONNECT
HANGUP
LINE
REPLY
RESET
SETUP

13.3.1.2 Sending and testing for character strings

SEND
SENDBREAK
SENDWAIT
WAIT

13.3.1.3 String operations

&
ASC
COLLATE
FOLD
LEFT

LOWER
MID
RIGHT
UPPER

13.3.1.4 File handling
APPEND
AS
CAPTURE
CATCH
CLOSE
COPY
DELETE
DOWNLOAD
EOF
EXCLUDE
EXISTS
INCLUDE
LOADNAMES
MOVE
OPEN
READ
RENAME
SAVENAMES
UPLOAD
WRITE

13.3.1.5 User information & interaction
ALERT
BEEP
CLS
INFO
MENU
QUERY
QUERYOK
STATUS

13.3.1.6 Program control
BREAK
CALL
CHAIN
CONTINUE
ELSE
ELSEIF
EXIT
FORGET
GOTO
IF
ON
THEN
WHILE
DO
REPEAT
UNTIL
SET

13.3.1.7 Logical operators
AND
NOT
OR

13.3.1.8 Expression operators
= - / * =
<>
<
>
<=
>=
%

13.3.1.9 Other
//
DRAIN
VERSION

13.4 Glossary of commands

This section contains an alphabetical list of all the commands available when creating scripts. Examples are given for each command. You may also find it useful to look at the example script supplied with the machine.

Note: Some commands, like **SETUP** and **MENU**, must be ended by a corresponding command, e.g. **ENDSETUP** or **ENDMENU**. It is indicated below where this is necessary.

13.4.1 How glossary entries are arranged

Glossary entries on script commands consist of three parts:

- **The name of the command itself.** This is on the left side of the page. Related commands and other commands explained in the same glossary entry are listed underneath.
- **The command syntax**. This appears to the right of the command name. The syntax indicates the other types of information that are needed for the command to operate, and how the components should be arranged.
- **The command definition.** This is found below the command name and syntax. The definition explains exactly what the command does, and gives examples of its uses.

13.4.2 Syntax abbreviations

The explanations of command syntax use a number of terms to denote different types of information:

- **<value>**: indicates a number.
- **<string>**: indicates a text string within quotation marks.
- **<string exp>**: indicates either text within quotation marks or a variable which contains text (e.g. **user$** if the information it represents is, say, “Adam”).
- **<variable>**: indicates either a persistent or non-persistent variable, e.g. **mailpass$** or **modeminit**.
- **<filename>**: indicates a filename within quotation marks. It is usually better to include the full path in filenames, e.g “**C:\Documents\Logs\Log1**” rather than “**Log1**”.

- **<handle>:** indicates a numeric label you have assigned to a file you are opening. You can open up to 9 files, using the handles 1-9.
- **<label>:** indicates a script label.
- **<command>:** indicates a scripting language command.
- **<exp>:** indicates an expression, e.g. a condition in an **IF...THEN** command such as '**day$="Tuesday"**'.

Note: If the syntax explanation just contains the command on its own, e.g. **DRAIN**, then the command does not require any additional information when used in a script.

- Where the command syntax contains square brackets, the elements within brackets are optional.

13.5 Commands

&	*see* SEND

//	//

Precedes a comment you have included to explain what part of a script does. Everything from the '//' to the end of the line is ignored. You can also put a comment after another command, e.g.

```
SEND "Fred","XModem" // Sends password
```

(There is no need to insert a colon.)

ALERT	**ALERT <string exp>[,<string exp>]**

Prompts the user with information (indicated by the text string) and waits for acknowledgement before continuing. E.g.

```
ALERT "No carrier - try again later."
```

You can add a second line to the alert dialog using an optional second string of text. E.g.

```
ALERT "No carrier.","Check modem is connected correctly
and try again."
```

AND
NOT
OR

AND
NOT
OR

Use the **AND**, **NOT** and **OR** logical operators to link conditions in an expression. E.g.

```
IF c>1 AND c<10 THEN
    INFO "Value is between 1 and 10"
ELSE
    INFO "Value is not between 1 and 10"
ENDIF
```

only carries out the first **INFO** command if *both* conditions are met. The following section of script

```
IF usern$="Jerry" OR usern$="Sam" THEN
    pass$="disk14"
ELSE
    GOTO getpass
ENDIF
SEND pass$
```

defines **pass$** and sends **pass$** if *either* of the conditions are met. You can use **NOT** to specify conditions which must *not* be met for a given action to be carried out.

```
IF username$="" AND NOT superuser$="Anton99"
THEN
    GOTO getuserinfo
ELSEIF superuser$="Anton99" THEN
    GOTO superlog
ELSE
    GOTO getpass
ENDIF
```

Here the script will jump to '**getuserinfo**' only the variable '**username$**' contains no information, and the variable '**superuser**' does *not* contain the information '**Anton99**'.

As

see **Open**

Asc

ASC (<string exp>)

Returns the **ASCII** code of a character. If a whole string is entered between the brackets, **ASC** returns the code for the first character unless you specify a particular character using **LEFT**, **MID** or **RIGHT**. E.g.

```
INFO ASC ("H")
```

returns the code 72.

Beep

BEEP <value>,<value>

Activates the MC 218's buzzer. The first value is the length of the note in 1/32s of a second, and the second indicates the pitch. E.g.

```
BEEP 16,300
ALERT "Connection dropped"
```

Break

BREAK

Breaks out of a loop.

Call

CALL <filename>

Stops running the current script and transfers control to a new one, without clearing any variables. E.g.

```
CALL "C:\Documents\Scripts\script2.scr"
```

starts 'script2.scr' running. When the called script ends, the Terminal emulation screen is shown. The program does not return to the calling script.

CAPTURE

CAPTURE <filename>,[APPEND[,DEBUG]]
CAPTURE OFF

Begins saving received characters to a specified file. Use **CAPTURE OFF** to stop the capture and save the file. E.g.

```
CAPTURE "C:\Documents\Logs\log15"
...
CAPTURE OFF
```

To add more captured information to a file that exists already, use the **APPEND** modifier. E.g.

```
CAPTURE "C:\Documents\Logs\log", APPEND
```

This simply adds the newly captured information onto the end of the old. You may find **CAPTURE** useful when reading your e-mail messages on a remote system. E.g.

```
CAPTURE "C:\Email\messages"
SENDWAIT 600 "readmail" , "ready:" GOTO fail5min
CAPTURE OFF
```

Sends "**readmail**" to the remote system - a command to 'read my messages' - and captures the resulting text. In this system, the remote machine signals the end of the mail by a return to the "**ready:**" prompt. If this happens within 5 minutes, the capture is terminated, if not, the script jumps to the '**fail5min**' label.

Note: To store control characters in the captured file, e.g. <13> or <8>, use the **DEBUG** modifier. E.g.

```
CAPTURE "C:\Scripting\log" , DEBUG
```

You can combine the two modifiers together by separating them with a comma. E.g.

```
CAPTURE "C:\Captures\log2" , APPEND , DEBUG
```

CATCH

CATCH {INCLUDE/EXCLUDE} ,<string>, <variable>, <timeout>, <maxlength> GOTO <label>

Stores incoming information to the specified variable. E.g.

```
CATCH INCLUDE, "OK" , R$ , 60 , 20 GOTO anerror
```

This captures incoming data to the variable **R$**. The capturing will stop after a time of **<timeout>** (in this case 60 half-seconds), when the string "OK" is received, or when twenty characters have been received. The script jumps to the label (i.e. '**anerror**') if the given string is not received before either the **<timeout>** value or the maximum number of characters (**<maxlength>**) is reached. Use **INCLUDE** if you want **<string>** to be included as part of the variable, or **EXCLUDE** if you do not.

CHAIN

CHAIN <filename>

Starts a new script, clearing all variables first. E.g

```
CHAIN "C:\Documents\Scripts\script3.scr"
```

starts 'script3.scr' running, whilst clearing all the variables set up in any previous scripts.

CLOSE *see* **OPEN**

CLS **CLS**

Clears the terminal screen.

COLLATE **COLLATE (<string exp>)**

Collates a string, by removing the differences between the characters that are deemed unimportant for the purposes of sorting them.

CONNECT **CONNECT <value> GOTO <label>**

Waits for the current handshaking method to allow transmission. The period to wait is specified in half-seconds. E.g.

```
CONNECT 20 GOTO again
SEND “Hello”
```

waits 10 seconds for the connection to be established. If it happens within this time, the script jumps to the next command (“**SEND**”), otherwise it moves to the ‘**again**’ label.

CONTINUE **CONTINUE**

Makes the script jump to the start of a **REPEAT…UNTIL** or **WHILE…DO…ENDWHILE** loop.

COPY **COPY <filename>,<filename>**

Copies a file, from the location specified in the first string expression to the location specified in the second. E.g.

```
COPY “C:\Logs\Logfile3”,“D:\Documents\logfile3”
```

will create a copy of the file ‘Logfile3’ in the specified directory on the D: drive.

DELETE **DELETE <filename>**

Deletes a specified file. E.g.

```
DELETE “C:\Logs\Logfile3”
```

DIR **DIR (<string exp>)**

Returns a directory listing of files. To list all the files in a directory, use the following sequence of commands:

```
X$=DIR(“C:\Documents\scripts\*.*”)
WHILE X$<>“” DO
    INFO X$
    X$=DIR
ENDWHILE
```

You can list specific files, say script files, by limiting the string expression. E.g.

DIR ("C:\Documents\*.scr")

which will only list files with the filename extension '.scr'.

DO

see **WHILE**

DOWNLOAD

DOWNLOAD <filename>,<string>

Prepares the MC 218 to receive a file from the remote machine, using a protocol indicated by the second string expression. E.g.

DOWNLOAD "D:\Documents\letter1" , "XModem"

The protocol is indicated by one of the following labels: ASCII; XModem; YModem (batch). Both XModem and ASCII protocols require you to specify a filename in the first string expression, as above. When receiving files using the YModem (batch) protocol, the filename is sent by the other machine, so a "blank text" string ought to be entered. E.g.

DOWNLOAD "","Ymodem (batch)"

DRAIN

DRAIN

Empties the "receive buffer" (the part of the memory where received characters are stored before being displayed). Any received characters which have yet to be processed will be discarded.

EOF

see **OPEN**

EXISTS

EXISTS <filename>

Returns a value of 1 or 0, depending on whether or not a file exists. If the file exists, the script carries out the commands specified by an **IF** command. E.g.

```
IF EXISTS "C:\Files\Log"=1 THEN
    GOTO gotlog
ENDIF
GOTO nolog
```

checks to see if there is a file called 'Log' in the 'Files' folder. If there is (i.e. if **EXISTS** returns a value of 1), the script jumps to the '**gotlog**' label. If there is no such file (and **EXISTS** returns 0), the script jumps to the '**nolog**' label. **EXISTS** can also be used to assign a value to a variable. E.g.

SEND EXISTS "C:\Files\Log"

will send 0 or 1 depending on whether or not the file exists, while

gotlogfile$=EXISTS "C:\Files\Log"

sets the value of the string according to the value returned by **EXISTS**.

EXIT

EXIT

Stops running the script and returns to the Terminal emulation screen. E.g.

```
endsession:
    INFO "End of session. Now hanging up."
    HANGUP
    EXIT
```

FALSE — *see* TRUE

FOLD

FOLD <string exp>

"Folds" a string: i.e. converts it to upper case, and removes any accents. E.g.

```
SEND FOLD Username$
```

FOR — *see* OPEN

FORGET — **FORGET**

Clears all variables.

GOTO

GOTO <label>

Jumps to the command following a specified label. E.g.

```
GOTO prompt
...
prompt:
    WAIT 60
    "Hello" GOTO gothello
    ...
    ENDWAIT
```

The **GOTO** command makes the script jump straight to the **WAIT...ENDWAIT** command. The specified label can be anywhere in the script.

HANGUP — **HANGUP**

Drops the DTR line for three seconds. If you are currently using a telephone line connection, your modem will hang up and drop the DCD line to the Psion. This command has the same effect as '**Hangup**' on the '**Transfer**' menu.

IF

IF <exp>THEN<command>{[ELSE]<command>[ELSEIF<exp>THEN<command>]} ENDIF

Allows you to specify a choice of instructions according to the conditions. **IF** requires both a **THEN** command, and an **ENDIF** command. E.g.

```
IF Username$="" THEN
    GOTO getuserinfo
ELSEIF Username$="Bob" THEN
    GOTO boblogon
ELSE GOTO getpass
ENDIF
```

checks to see if there is any information stored in the '**Username**' name. If there is no such information, the script jumps to the '**getuserinfo**' label. If the name contains the information '**Bob**', the script jumps to the '**boblogon**' label. If the name contains something other than '**Bob**', it jumps to the '**getpass**' label. You can link more than one condition to an **IF** command by using the **OR** and **AND** logical operators. E.g.

```
IF Username$="" OR Password$="" THEN
    GOTO getuserinfo
```

jumps to the '**getuserinfo**' label if *either* of the names has no information associated with it. On the other hand,

```
IF Username$="" AND Address$="" THEN
    GOTO getuserinfo
```

jumps to the '**getuserinfo**' label only if *both* of the names have no associated information.

INFO

INFO <string exp>

Displays a message in the terminal screen, e.g. about the action currently being performed. E.g.

```
INFO "Initialising modem and dialling out..."
```

You can use **INFO** messages to keep you informed about the progress of your connection.

INPUT

see OPEN

LEFT
RIGHT

LEFT (<string exp>,<value>)
RIGHT (<string exp>,<value>)

Extracts a specified number of characters from the furthest left or right part of string. E.g.

```
SEND LEFT (logname$,3)
```

will send "**sec**" if the information stored as **logname$** is "**secondlog**". Similarly,

```
SEND RIGHT (logname$,3)
```

would send "**log**".

LEN

LEN <string exp>

Returns the length of a string. E.g.

```
namelen=LEN (name$)
```

will assign a value of 5 to '**namelen**' if the string "**Anton**" is stored as **name$**.

LOADNAMES
SAVENAMES

LOADNAMES <filename>
SAVENAMES <filename>

Loads a given names file, so that you can use its information in the current script. E.g.

LOADNAMES "C:\Documents\Comms\BBSnames"

SAVENAMES saves all the current names information in a specified file. E.g.

SAVENAMES "C:\Documents\Names\Newnames"

LOWER — *see* UPPER

MENU

```
MENU <string exp>
   <string exp> GOTO <label>
   <string exp> GOTO <label>
   ...
ENDMENU
```

Offers a menu of options to the user. Each option jumps to a particular label when it is chosen. E.g.

```
MENU "Choose action"
      "Connect to server" GOTO logon
      "Upload file" GOTO sendfile
      "Download file" GOTO downl
      "Exit" EXIT
ENDMENU
```

creates a menu with the heading "Choose action" from which the user can select one of four actions (e.g. "Upload file" etc). The script then carries out the command next to the menu item, e.g. jumping to "sendfile" for the "Upload file" menu item. Menu items can only be followed by **GOTO** or **EXIT** commands. **MENU** requires a matching **ENDMENU** command.

MID

MID (<string exp>,<value>[,<value>])

Extracts part of a string or variable, beginning a specified number of characters (indicated by the first value) from the left. E.g.

```
logfilename$="C:\files\log1"
INFO "..." & MID (logfilename$,9)
```

will display "...log1" in the terminal screen. The optional second value allows you to specify the number of letters to be extracted. E.g.

SEND MID ("password15",4,4)

will just send the characters '**word**'.

MOVE

MOVE <filename>,<string exp>

Moves a file, from the location specified in the first string expression to the location specified in the second. E.g.

MOVE "C:\Logs\Capture","C:\Email\"

NOT — *see* AND

ON ERROR GOTO
ON ERROR OFF

ON ERROR GOTO <label>
ON ERROR OFF

Makes the script jump to the specified label if an error occurs. The line

```
ON ERROR OFF
```

returns Comms to normal error handling.

OPEN

```
OPEN <filename> FOR {INPUT/OUTPUT/APPEND} AS <handle>
   READ <length>,<variable>,<handle>
   WRITE <handle>,<string>
   EOF <handle>
CLOSE <handle>
```

Opens files and enables reading from ("input") and writing to ("output") them. When you open a file, you assign it a "handle" (a number from 1 to 9), which you then use to refer to that file. You can have up to 9 files open at a time. E.g., to write a text string to a file:

```
OPEN "C:\Files\Connectlog" FOR OUTPUT AS 1
      WRITE 1,"Connect failed"
CLOSE 1
```

opens the file "Connectlog" to write to it, then writes "Connect failed" and closes it again. You can use **OPEN** to read information from one file and write it to another. When reading from a file, use **EOF** to look for the end of the file, so that Comms knows when to stop. E.g.

```
OPEN "C:\sourcefile" FOR INPUT AS 1
OPEN "C:\destination" FOR OUTPUT AS 2
WHILE NOT EOF(1) DO
      READ 16,a$,1
      WRITE 2,a$
ENDWHILE
CLOSE 1
CLOSE 2
```

This script opens the two files, then instructs Comms to read 16 characters into **a$** from the first file and write the contents of **a$** to the second file. These instructions are held within a **WHILE...DO...ENDWHILE** loop, so they are repeated until the end of the first file is reached.

OR

see **AND**

OUTPUT

see **OPEN**

QUERY

```
QUERY <string exp>
   <string exp>,<variable>
   <string exp>,<variable>
ENDQUERY
```

Prompts the user for information which is then stored as a name. E.g.

```
QUERY "Enter details"
      "Username:" , myname$
      "City" , city$
ENDQUERY
```

creates a dialog where the user can enter information such as "Username" etc. This information is then stored with the name that follows, e.g.

'**myname**'. Up to six items can be prompted for in a **QUERY** dialog. The string which follows **QUERY** is the title of the dialog. **QUERY** requires a matching **ENDQUERY** command. **QUERYOK** allows the script to act differently depending on whether the **QUERY** dialog was exited by tapping '**OK**', or by pressing Esc. E.g.

```
QUERY "Enter details"
      "Username:" , myname$
      "City" , city$
      "Password" , pass$
ENDQUERY
IF QUERY OK THEN
      GOTO logon
ELSE
      GOTO cancelled
ENDIF
```

jumps to the '**logon**' label if '**OK**' is pressed, but the '**cancelled**' label if the **QUERY** dialog is exited using Esc.

READ

see **OPEN**

RENAME

RENAME <filename>,<filename>

Renames the file indicated in the first string expression to the name specified in the second. E.g.

```
RENAME "C:\Documents\Names","Names2"
```

REPEAT
UNTIL

```
REPEAT
   <command>
UNTIL <expression>
```

Repeats a command until a certain condition is met. E.g.

```
SET a=5
REPEAT
      SEND "<013>"
      a=a-1
UNTIL a=0
```

Sends carriage returns and deducts 1 from the value of **a** until the value of **a** reaches 0.

Note: The important difference between **REPEAT...UNTIL** loops and **WHILE...DO...ENDWHILE** loops is that **REPEAT...UNTIL** always executes the specified action at least once, and ceases to do so when a condition is met. By contrast, **WHILE...DO...ENDWHILE** checks for a condition *before* ever carrying out an action, so if the condition is not met, the action will not be executed at all.

RESET

RESET

Returns all communications settings back to their standard values. You may want to use this before a **SETUP** command, so that you can be sure of the complete configuration. E.g.

```
RESET
SETUP
    Baud=57600
    Parity=Even
ENDSETUP
```

RIGHT	*see* **LEFT**
SAVENAMES	*see* **LOADNAMES**

SEND
&

SEND <string exp> [;] [& <string exp>]

Sends data to a remote machine, usually for controlling a modem or for conveying information to a remote system. E.g.

```
SEND "atz"
```

sends a modem initialisation string. You can add the character code for the Enter key to the **SEND** command by putting a semi-colon after the string, e.g.

```
SEND "mypassword";
```

To include control characters, such as carriage returns or line feeds, in a **SEND** string, use angled brackets. E.g.

```
SEND "mail<13><10>"
```

sends a carriage return (control character number 13) and then a line feed (control character number 10) after '**mail**'. Control character codes can also be entered in hexadecimal form, prefixed with a $ sign. E.g.

```
SEND "bye<$1B>"
```

puts the control character code 27 (the 'Esc' control) at the end of the **SEND** string. You can send more than one string at a time by using the **&** operator. E.g.

```
SEND "john" & "password29"
```

SENDBREAK

SENDBREAK

Sends a "break" (i.e. transmits nothing for a short space of time) to the remote machine. The effect of this depends on the type of remote machine and the software it uses.

SENDWAIT

SENDWAIT <value> <string exp>,<string exp> GOTO <label>

Combines a **SEND** command with a **WAIT** command. E.g.

```
SENDWAIT 60 "Anton" , "Password:" GOTO nopass
GOTO pass
```

does exactly the same as

```
SEND "Anton"
WAIT 60
    "Password:" GOTO pass
```

```
GOTO nopass
...
```

I.e. the MC 218 sends "**Anton**", then waits for 30 seconds (60 half-seconds) for the remote machine to send "Password:". If "Password:" is received, the script jumps to the '**pass**' label. If "Password:" is not received, the script moves on to the next command ('**GOTO nopass**'). The script only follows the **GOTO** command after "Password:" if "Password:" *isn't* received.

SET

SET <variable>=<value>

Set assigns a variable or name a given value. E.g.

```
SET me$="Albert"
SET c1=5
```

Assigns the information "**Albert**" to the name '**me**' It also gives the variable **c1** a value of 5. You can then decrement this value to control the number of times an action is performed. E.g.

```
setcount:
    SET c1=5 sendcr:
    SEND "<13>"
    c1=c1-1
    IF c1=0 THEN
        GOTO next
    ELSE
        GOTO sendcr
    ENDIF
next:
...
```

first gives **c1** a value of 5. The script then sends a carriage return character ("<13>") and deducts 1 from the value of **c1**. If **c1** still hasn't reached 0, it repeats this step. Once it has sent 5 carriage returns, it will move on to the '**next:**' label.

SETUP	**SETUP**
BAUD	**BAUD {rate}**
PORT	**PORT {COMM::0}{IRCOMM::0}**
DATA	**DATA {5}{6}{7}{8}**
FAIL	**FAIL {DSR}{DCD}{PARITY}**
HANDSHAKE	**HANDSHAKE {DCD}{RTSCTS}{DSR}{XONXOFF}{NONE}**
PARITY	**PARITY {NONE}{ODD}{EVEN}**
STOP	**STOP {1}{2}**
TIMEOUT	**TIMEOUT {timeout}**
ENDSETUP	**ENDSETUP**

Sets any or all of the communications parameters. All of the above parameters can be set, to any of the options within square brackets. You can separate each of the parameters either with a colon, or by putting them on a new line. E.g.

```
SETUP
    BAUD=19200 : STOP 1
    DATA=8
    PORT="COMM::0"
ENDSETUP
```

Note: For both **HANDSHAKE** and **FAIL**, you can set more than one of the choices by separating them with a comma. E.g.

```
HANDSHAKE=XONXOFF,DCD
FAIL=DCD,DSR
```

Parameters you do not specifically set with **SETUP** remain unchanged. **SETUP** requires a matching **ENDSETUP** command.

STATUS

STATUS <string exp>

Displays an information message in the status bar at the bottom of the Terminal emulation screen. Use **STATUS** messages to keep you informed about the progress and state of your connection, e.g.

```
STATUS "Connected to BBS."
```

TRUE
FALSE

TRUE
FALSE

Return 1 if a specified condition is true, and 0 if it is false.

```
IF user$="bob" THEN
    f$=TRUE
ELSE
    f$=FALSE
ENDIF
SEND f$
```

Will send 1 if the information stored as **user$** is "**bob**", and **0** if it is not.

UNTIL

see **REPEAT**

UPLOAD

UPLOAD <filename>,<string>

Prepares the MC 218 to send a file to a remote machine, using a protocol indicated by the second string. E.g.

```
UPLOAD "D:\Documents\letter1" , "XModem"
```

The protocol is indicated by one of the following labels: ASCII; XModem; YModem (batch).

UPPER
LOWER

UPPER <string exp>
LOWER <string exp>

Convert text to upper or lower case. E.g.

```
SEND UPPER username$
```

will send "**CLIVE**" even if the information stored as "**username$**" is "**Clive**". Similarly,

```
INFO LOWER prompt$
```

will display the information in lower case. You can combine **UPPER** with other text control commands such as **MID**, **LEFT**, **RIGHT** etc. E.g.

```
STATUS UPPER LEFT (L$,1) & LOWER MID (mid(L$,1))
```

will display **L$** with a capital letter at the start and the rest in lower case.

VERSION

VERSION

Returns the version number of Comms.

WAIT
ENDWAIT

```
WAIT <value>
   <string exp> GOTO <label>
   <string exp> GOTO <label>
   ...
ENDWAIT
```

Pauses the script for a given number of half-seconds, and waits for one of several strings. E.g.

```
WAIT 10
      "No carrier" GOTO tryagain
      "Hangup" GOTO closeconn
ENDWAIT
SEND "ATDT91011992220"
```

pauses for 5 seconds before proceeding with the next command in the script ('**SEND**'). If, during that time, the MC 218 receives "**No carrier**" from the remote machine, it will carry out the '**GOTO tryagain**' command. Similarly, if the MC 218 receives "**Hangup**" during the 5 seconds, the script will jump to the '**closeconn**' label. **WAIT** requires a matching **ENDWAIT** command.

WHILE
DO
ENDWHILE

```
WHILE <exp> DO
   ...
ENDWHILE
```

Carries out a given command while a certain condition is met. E.g.

```
WHILE R$<>"OK" DO
      SEND "AT"
ENDWHILE
```

sends the "**AT**" string as long as **R$** is not equal to "**OK**".

Note: The important difference between **REPEAT...UNTIL** loops and **WHILE...DO...ENDWHILE** loops is that **REPEAT...UNTIL** always executes the specified action at least once, and ceases to do so when a condition is met. By contrast, **WHILE...DO...ENDWHILE** checks for a condition *before* ever carrying out an action, so if the condition is not met, the action will not be executed at all.

14. FAQ

14.1 General

Q: What is included in the MC 218 kit, and what kits/ language versions will there be?
The MC 218 kit contains;

- MC 218 main unit running EPOC, with all programs you need pre-installed
- Batteries
- Backup battery
- Ericsson Infrared Modem
- RS-232 cable
- Ericsson User's Guide
- Ericsson Quick Guide.
- CD-ROM with SW and manuals
- Carrying case
- Warranty card

The MC 218 will be available in International English, American English, German, French and Spanish language versions. There will be another two keyboard versions (Portuguese and Nordic), both with an English operating system. Finally there will be a version of the International English kit made for Central and Eastern Europe with manuals for these countries.

Q: What cameras are supported?
Cameras using IrTran-P v1.0 or Compact flash memory disks and supporting the Jpeg file format. If a camera meets these basic requirements, the MC 218 should be interoperable with it.
The MC 218 is tested with but not limited to:
JVC GC-S1
Sharp VE-LC25
Casio GV 7000

Q: What is EPOC?
EPOC is the operating system from Symbian. Symbian is a company owned by the major players in the Mobile Telecommunications world: Ericsson, Nokia, Matsushita, Motorola and Psion. EPOC has several characteristics that make it the most appropriate solution for manufacturers of wireless information devices.
Real time - designed from scratch to be a really 'real time' operating system
User friendly user-interface, with for instance zoom functionality
Power efficient
Very fast, which is necessary for a mobile device. The user requires instant response.
A large community of 3rd part developers.

Q: What about battery lifetime?
Depending on usage (how much typing, screen update, backlight activated or not, communication etc.), the operational time is approx. 60 hours with high capacity AA batteries (Based on a usage model of 10% active use, 90% in wait-mode with the screen on). An advantage is that you can always buy standard AA batteries at the nearest grocery store, and instantly be up and running again. The battery time is so long that a normal user will not need a mains adaptor for external power supply.

14.2 Applications

Q: What applications are included?

- EPOC operating system
- My Phone
- Postcard
- Message (e-mail, SMS, fax)
- Mobile Internet (WAP browser)
- Instant Ericsson Mobile Internet
- Ericsson Infrared Modem
- Ericsson manuals
- Contacts
- Calendar / support for iCalendar
- Internet (HTML web browser)
- Word
- Sheet
- Record (with voice note functionality)
- Jotter
- Sketch
- Data (information card index system)
- Calculator
- World clock
- Spell check
- Alarms
- Games

Q: What does the My Phone program do?
The My Phone program allows you to make and end calls from the MC 218, manage your phone's address book from the mobile companion and modify the phone's settings including an easy to use keyboard for ring melody handling. My Phone is a unique Ericsson program, which is pre-installed at delivery.

Q: What does the Postcard program do?
The Postcard program allows you to insert a digital image from a digital camera into an electronic postcard. The image can be transferred via IR directly from the camera or on the Compact Flash. The image is then combined with text and personal drawings in a user-friendly interface and the Postcard is sent as e-mail. Postcard is a unique Ericsson program, which is pre-installed at delivery.

Q: What does the Mobile Internet (WAP Browser) program do?
"Mobile Internet" - one of the first commercially available WAP browsers in the world. This application will give you access to information and services which are optimized for mobile use. Being a leader in the mobile communications business Ericsson will also provide the user with access to the site with the same name. It is one of the first sites to deliver optimized content. Over time more or less all mobile devices will include a Mobile Internet program.

Q: What is Contacts?
Contacts is a database optimized for storing contacts data, such as phone numbers, e-mail addresses, mail addresses.

Q: I have created a graph but I want to change it and I can't.
Move to Sheet view and amend the data or the highlighted region as required. Select Graph view. Select 'Replace ranges' from the Ranges menu or tap the Ranges button on the top toolbar.

Q: Can I password-protect a Sheet file?
Yes. Select 'Password...' from the File menu.

Q: I've forgotten my spreadsheet's password, what can I do?
Delete the file and start again.

Q: Sheet Produces "Information in use" when I try to open it.
Create a sheet file and password protect it. Close the file and return to the system screen. Now open the file but press ESC before the File opens. Now if you try to open the file you will get "Information in use" WHY ? Answer Tap on system above the control panel button and find the 'Sheet' process in the list close this and you will once again be able to open the file.

Q: How can I make an X\Y Graph?
To set the X range you need to set the graph type to X/Y scatter. The change range dialogue then includes an entry for "Scatter range (x values)". Note: you should not select the X range if you are doing an x/y plot since each Y range is associated to an X range.

Q: How many levels of Undo does Sketch have?
Five.

Q: How big can a Sketch file be?
Dimensions: 968 by 480 pixels or 8 by 3.7 inches or 20.33 by 10.08 cm

Q: My clock keeps losing/gaining time or my alarms don't go off.
Check the time on the PC they have been connecting to. The MC 218 can synchronize clocks with your PC when you connect so if your PC is 2 hours out your MC 218 will be after you have connected.

Q: Can I set a snooze alarm and then carry on working?
Yes. When the alarm rings, press the SPACEBAR to select the snooze time, then press CTRL+I to go back to the foreground program. If you just press CTRL+I, then the snooze is activated for 5 minutes. You can see the re-queued alarm in Time in the "Next alarm" list.

Q: How many changes does Time (map) database support?
There is no limit in principle to the number of changes.

Q: Why does the machine tell me 'Access denied' whenever I try to test the sound for the alarm I am currently trying to set?
If you press 'Enter' you will be given a more meaningful message -- you will be told 'All sounds currently disabled'. Basically, if you get 'Access denied' when you tap the 'Test sound' option, check the control panel and make sure that sounds are enabled.

Q: I have entered a lot of entries in World and want to delete all of them but can't remember the things I entered ?
There does not seem to be anyway to delete this, Where does it hide the file ?
\system\data\wldserv.dat

Q: I have cancelled an alarm and now I can't remember what it was for...
Use 'View past alarms' from the 'Edit' menu.

Q: Can I add a new country?
Yes. 'Add country...' from the 'Edit' menu of Time's Map view.

Q: The Time program isn’t running - does that mean alarms won’t go off?
The Time program does not have to be running for normal alarm and clock operations.

Q: What does a flashing red light mean on the outside of the case?
The light flashes when the alarm goes off. If you hit SNOOZE, then the light continues to flash until you finally turn off the alarm.

Q: I set an alarm with an associated sound but it goes off silently and displays "Alarm sound is off".
There is an option with Time to turn off alarm sounds only. This can be confusing because ’testing’ the alarm makes it sound off but the when the alarm goes off at its set time, it is silent. To fix this, choose the ’Sound’ option from the ’Tools’ menu of Time. Change the ’Alarm sound’ from ’Off’ to ’On’.

Q: Can I use templates in Word
Yes.

Q: How do I produce the EURO Symbol?
The EURO Symbol is a special character and therefore can be produced by pressing SHIFT+CTRL+C and selecting the symbol.

Q: How can I create a table in Word?
Insert a spreadsheet object and you can get most of the effects tables are used for.

Q: How can I get an overview of which styles I’ve used in which paragraphs?
Menu | Paragraph | Style gallery
This shows all the paragraph styles in the document.
The MC 218 does not create a new style every time you change the paragraph settings for the current paragraph.

Q: Footnotes
Footnotes are not available in MC 218’s Word Processor.

Q: Can I change my default template for Word files?
Yes. Make the necessary changes to a Word file and move it to Then choose "Menu | More | Save as template..." and give it the name "Normal".

Q: There is no Thesaurus button in Word.
The thesaurus cannot be accessed from within Word.

Q: Is there a quick way to append country codes to an entry in my database?
No.

Q: How do I display all data records again after doing a find?
Press Esc

Q: Can I change the length of my fields in the data application after I’ve defined them?
No. You can change field order though. There are options in both the card and the list view to move things around onscreen. Doing a ’save as’ should write these to disk in a new file.

Q: I have a Data file that won’t let me add or edit entries...
If the ’Add entry’ and ’Edit entry’ options are greyed-out it indicates that the file is read-only ie the user has set this. Close the file and select ’Properties...’ from the ’File’ menu of the system screen. Remove the tick from the ’Read-only’ box. The file will now behave normally.

Q: I don’t like the default labels. Is there a way of creating my own default labels for new databases?

Yes. Start a new database and edit the labels to suit. Whilst still in this database, choose the 'Create new file...' option from the 'File' menu. The new database will inherit the labels from the originating database. The new database will also retain any preferences set in the original database.

Q: How do I undelete records from a Data file? I have accidentally deleted some records from my database. Can I get them back?
The "Undo delete" option in the "Edit" menu is the only hope of recovering a single, recently deleted record. Nothing more can be done to undelete numerous records. As no one has any knowledge of the low-level data format, it is not possible to undelete entries using another program either.

Q: How can I change the type of a field in an MC 218 database?
Open the database you want to alter Select "Create New File", which makes an empty database with the same field names Delete the field(s) you want to restructure - in my case the "Notes:" field Add the new field(s) with exactly the same names as the old ones Under "File", "More", select "Merge In..." and select the original database

Q: When I try to sort by more than one label, I get "Labels too long for sorting".
Problem: One of your fields has more than 200 characters. Solution: Use different fields to sort by or change the field information to use fewer than 200 characters.

Q: Why is an (sound recording for example) embedded object "read only"?
If you select the object from the main view, the object is read only. To modify the object, you must first select "edit entry" and then select the object.

Q: "Setup has detected a previous installation on your machine..."
For Information Purposes: If a user receives the following message, it is likely that they have tried to uninstall EPOC Connect 2.x manually and got it in a bit of a mess, and then tried to re-install. "Setup has detected a previous installation on your machine. Some of the application files are still in use. Please close all Explorer windows and reboot your computer or restart Windows. " The message occurs as a result of a check during the installation for any existing files that are open (and consequently cannot be overwritten). It will cause the installation to quit. This is usually the result of still having the connection server running. To get round this, the user should close the connection server and delete all the files in the installation directory.

Q: Can you convert files without an MC 218 connected?
Yes using the stand alone converter application. You can find a shortcut to it in "Start Menu | Programs | Ericsson EPOC Connect | Convert Files."

Q: Can I have more than one version of EPOC Connect installed on my PC?
No.

Q: When I try to synchronize, I get “unable to initialise API, must supply valid account details”. What does it mean?
Their mailbox login is set incorrectly.

Q: When I try to synchronize to Lotus, I get “cannot find Calendar”.
This is because they have renamed their section for appointments / day entries in Lotus. Just change the specified section in Day entries / Appointments under Entry Types under Properties.

Q: Are third party developers creating plug-ins for EPOC Connect?
Yes, check the site:
http://www.advansyscorp.com.

14.3 Communication & Synchronisation

Q: What desktop computers are supported, and how does PC synch work?
EPOC facilitates complete synchronization with all leading Windows® 95/98 and NT 3.51 and NT4 PC desktop software including programs from Microsoft, Lotus and Corel. In addition, it supports both embedded graphics and sound files. All calendar, contacts and ToDo data can be synchronized automatically. In addition to this files can be dragged and dropped in an easy way between the MC 218 and the desktop PC in any direction. The Ericsson EPOC Connect also includes a backup and restore facility.
Ericsson EPOC Connect is an open program, which allows plug-ins. This further strengthens the range of possible PC programs that the user can synchronize with. Several 3rd party synchronization plug-ins already exist.

Q: When I try to synchronize, I get "unable to initialise API, Network logon validation error". What does it mean?
Their mailbox password is wrong.

Q: "Unable to open ..." dialog when synchronizing with Lotus Organiser 2.1
Problem: I set up and ran an Organiser 2.1 contacts synchronization successfully. Seeing the option in the property pages to change which section is used ('Contacts to synchronize') I changed my Address section in Organiser 2.1 to be called Contacts. Synchronizing now gives me 'Unable to open Contacts.' Answer: Apparently this is a bug in Organiser 2.1 whereby if a tab was renamed, Organiser displays one thing whilst still thinking it is the other. Workaround 1: Change the tab back to what it was originally called Workaround 2: Create a new .or2 file and import the contact information into to, making sure you do not rename any of the section tabs

Q: Is MC 218 supporting the e-mail protocol IMAP 4?
Yes.
Q: What is Ericsson Mobile Internet?
Ericsson Mobile Internet is a dedicated service for the travelling business person, using Ericsson Mobile Data products. You'll find fast access to useful information; User's guides, latest updates on software, links to 3rd party software, which is tested and recommended by Ericsson. The MC 218 contains a wizard that step-by-step guides you through a set-up process. Ericsson mobile Internet can both be accessed through the traditional WEB browseer as well as the new Mobile Internet browser (based on WAP). The wizard will help with the set-up in both cases. Ericsson Mobile Internet will be your portal to the new WAP world.

Q: What version of WAP is supported?
The version shipped with the product will be based on the 1.1 version of the specifications.

Q: Why an InfraRed modem? Will it fit my laptop also?
The combination of your choice of Ericsson phone, MC 218 and the Infrared modem really solves the need for cable free connection; no hassle, just snap on the modem to the phone, align the IR eyes and you are immediately connected. The Infrared modem contains both data and fax modems, implying:

- You don't need any expensive, power consuming fax/ data modem card for connecting to your phone.
- Small devices like MC 218 have a sufficient, but limited computing power. One solution could have been to implement the modems in software on the PDA, but especially when it comes to fax modems with its inherent critical timing issues, the computing power simply is not sufficient to make fax modems stable and reliable. Ericsson is a telecommunication company, and is very knowledgeable in the modem business.

Windows drivers for the Infrared modem is available on the Ericsson Mobile Internet, for any user wanting to connect the modem to a laptop or IR enabled desktop.

Q: What is an IrDA infrared data connection?
A cordless data connection using infrared light. It is a low-cost transceiver signalling technology for two way data exchange. It provides high-speed digital exchange through the typical PC UART/serial port at 9600-115200 bits/s, and in some units compatible high-speed extensions up to 1Mb/s and 4Mb/s speeds.

Q: What are the distance limitations for IrDA-compliant infrared connections?
Although the IrDA standard only specifies a connection from zero to one metre, many IrDA-compliant products can connect at distances greater than this.

Q: What is IrDA?
The initials IrDA stand for the Infrared Data Association. The IrDA is a non-profit trade association with a membership of over 160 companies representing computer and telecommunications hardware, software, components and adaptors.

Q: What is the difference between diffuse infrared, directed infrared, and radio frequency?
Diffuse infrared allows many-to-many connections, does not require direct line of sight and can be uni- or bi-directional. Since it is based on visible light, it is a secure form within a room. Financial trading floors are an example of diffuse infrared.
Direct infrared is point-to-point, typically one-to-one communications, is not subject to regulations, requires line of sight and is a secure form of data transmission and reception. IrDA is an example of directed infrared.
Radio frequency is not secure in that it can penetrate walls, is subject to uncontrolled interference, is typically higher in power than directed infrared and requires FCC certification.

Q: How secure is infrared?
Very secure. Using infrared connection to access the LAN is as secure as using a cable at any other access point on the network. You need to be an authorized user on the subnet.
Q: How reliable is infrared?
Often more reliable than wired solutions. When was the last time your TV remote control broke? In fact, we believe that the IR port will prove more reliable than wired connections because we will have eliminated wear-and-tear. No pins to bend, no plugs to jam.

Q: Does the length of the infrared connection affect the speed of the network?I.e., if the portable is farther away from the network access point, will the connection be slower?
To be IrDA-compliant a product must be capable of maintaining a constant connection speed.

Q: What do FIR and SIR stand for?
FIR stands for Fast Infrared, which is the capability to transfer data up to 4Mbps. SIR stands for Serial Infrared which is the capability to transfer data at 115Kbps.

Q: What is the maximum baud rate the MC 218 can handle?
115200 bps.

Q: Can I beam a file to another MC 218?
Yes! Just mark the file and send it by using infrared. Please note that that the infrared modems must point towards each other.

Q: Can I beam a Contact or Calender entry to another MC 218?
You should be able to send any Contact or Calender entry to another palmtop that allows the beaming of vCard or vCalendar objects over IrDA.

14.4 Hardware

Q: Can I use the Psion Dacom Travel Modem with my MC 218?
Yes

Q: What do I need to connect to a parallel printer?
The Psion Parallel Printer Link.

Q: Can I print to any Macintosh Printers, even ones with PostScript fonts?
No. There are no PostScript drivers in the MC 218.

Q: Can the MC 218 print to printers that do not have built-in fonts?
Yes, however, the built-in fonts would come out looking extremely small when viewed on a modern high-resolution printer. This is because MC 218 screen fonts are bitmap fonts not scaleable ones. Printer fonts are utilized in order to ensure the highest quality of typeface reproduction. Of course, the MC 218 can print to any printer via PC.

Q: Will there be a PC drive for Memory Disks?
You can plug a compact flash card into an adaptor that makes it pin compatible with a PC Card, so it will plug into any laptop with a PC Card (PCMCIA) slot or any PC Card Drive (e.g. Psion Dacom's Gold Drive.) The interface is a standard ATA interface, so the card should look like a hard disk.

Q: Will the machine be affected by airport Xray/security machines?
There are no known problems with passing the MC 218 through airport security machines and other X-Rays

Q: How can I clean the screen without randomly using the menus?
Ensure 'Switch on when screen is tapped' is not ticked in the Switch on/off control panel of the system screen.

Q: My MC 218 has been repaired and now some of my bespoke/third party software won't run.
Each MC 218 ROM contains a unique ID that can be read from software. It is possible that bespoke applications especially those aimed at vertical markets (corporates) might use this as a copy-protection scheme. If the unique ID changes because the ROM has been replaced at a service centre such software might then refuse to run. Contact the author of the software for advice.

Q: What kind of power supply do I need?
A 6v 1A centre pin +ve

14.5 Hints & Tips

Q: How can I easily add accented characters/symbols/fraction signs etc?
SHIFT+CTRL+C will display an 'Insert special character' dialog in word. This doesn't work in Sheet. In Data it's only usable from within the edit/add dialog (not the find bar) etc. To get special characters into Sheet you can put them into a Word file, copy them to the clipboard (CTRL+C) and the paste (CTRL+V) them into the Data/Sheet file as required.

Q: How can I free up more memory?
1. Go to the system screen and close all files (File | Close | All Files). If an app is busy, skip it. 2. Repeat step 1 until all apps are and nothing is on the task list apart from "System" 3. Empty clipboard (from Edit|Clear clipboard, on the system screen)

Q: Limited options embedding graphs in word
If you need to add information to a graph such as call out boxes extra text or just want to tidy it up before inserting it into Word, you can use the capture screen option, hold down “shift”, “Ctrl”, “Fn” and s, to take it into sketch and then insert an appended sketch object.

Q: Is there a simple way to increase/decrease numbers in dialog boxes e.g Time and Date?
The 'M' and '?' keys decrement by 1 and increment by 1 numbers respectively. Shift-M decrements by 10 whereas Shift-? increments by 10.

Q: Sketch tips
Use Rotate or Flip on the Transform menu to get further clipart variations. Press the Ctrl key while drawing an ellipse to produce a circle or while drawing a rectangle to produce a square. Drag a corner "handle" to resize a selection and keep the "aspect ratio" (i.e. the ratio between the height and the width). Use the pen to position the selection in roughly the right place, then use the arrow keys to line it up exactly. You can trace over a paper drawing in Sketch.

Q: When you go to an application, how can you find out exactly where the currently open file lives?
Whilst inside the application, tap on the filename on the toolbar and you get a list of all open files. The current file is the one highlighted and shows its full location including disk & folders.

Q: In the system help how do I see a list of help topics relating to any one application?
Add an exclamation mark after the application name in the Find box, e.g. calender!. This will then give you a list of the topics for that one application.

Q: Is there any way to display a background picture other than the Ericsson image?
Yes, Press CTRL+K on the System screen to get the System preferences, which allow you to change the wallpaper. You can create your own wallpaper via the sketch program

Q: How can I tell exactly which version of the OS I am running?
From the System screen you can press: Menu|Information|Machine SHIFT+CTRL+Q

Q: How do I control MC 218 from the keyboard alone?
Menu Key to call up menus, arrow keys to navigate, Enter to select an option, Tab to open a pop-list, hotkeys listed on all menu items. In addition: In Tabbed dialogs, to move between tabs - use the Up arrow to highlight the tab's name and then use left and right to switch tabs. Menu cascades operate the same as other menus, try it and see In System arrow keys highlight files, Enter opens, Fn-Enter opens & leaves any other files of that type open. Preference option to control this behaviour. Omission (these only really matter if the digitizer breaks). No way to multi-task from the keyboard alone (app buttons are part of the screen's digitizer). You can list open files from System, but if you're in an app you can't task out without using one of the application buttons or by touching the screen. No way to run apps, or open the 'Extras' bar from the keyboard.

Q: EPOCWorld Information
"EPOC World is Symbian's Developer organisation, supporting a growing world-wide community of developers creating software solutions for our EPOC mobile computing platform." See http://www.epocworld.com/

Q: How do I take a screenshot?
Press Ctrl-Shift-Fn-S to generate a screenshot in MBM format. You can view MBM files in Sketch by using 'Merge in...' (on the More cascade of the File menu). (MBM files are referred to as the EPOC picture File type.) This won't work with MBM files in the ROM These are ROM image files - not file store files.

14.6 Record

Q: Is there a way to disable the dictaphone buttons?
No.
Q: How much sound can I realistically record on a 16 Mb machine?
4 minutes per Mb is the current maximum.

Q: My external record buttons are not working.
Check the following: If they are trying to use the External record button and a password or owner info has been set on the machine then they will need to open a file called "voice notes" and leave it running. The machine is unable to start the application. Note if the file you create is on a CF/memory disk it will still not work it must be on drive C.

Q: When I use the dictaphone record button it doesn't always record.
You have to press the button once to start the program, then press and hold to record. Otherwise the app starts and nothing else happens. It's also worth opening up the machine to check if you have a dialog telling you your batteries are low.

Q: Can I manipulate sound files to add, for example, an underwater effect?
Not without some third party software.

Q: Can I record directly onto CompactFlash?
Yes, but if you use the dictaphone buttons a file will be created in the default folder called 'Voice notes', and, as the default folder can only exist on the 'C' drive this means that all sound recorder using the dictaphone buttons can only be recored on 'C'. Of course, you can still copy and paste the sound onto another drive if you wish.

14.7 System

Q: Can I password-protect all my files?
No, not all programs allow you to password protect your files. However you can do this with a third party product called "Encrypt it" available at http://www.palmscape.com

Q: Whenever I close the case my MC 218 turns off.
This is controlled by a preference in the Switch on/off control panel in the system screen.

Q: What is a 'bookmark'?
A bookmark points to a folder that you use often. If you set a bookmark to a favourite folder then you can use it to jump to that folder quickly. You can only set one bookmark at a time. The last one you set (with 'Set bookmark' from the 'Edit' menu) is the only one that is remembered.

Q: I have accidentally deleted a file from my internal disk - can it be recovered?
No.

Q: I can't rename a folder if it contains files that are open.
That's correct. You can't. Close the file(s) and then rename the folder.

Q: Can I move files?
Yes.
Use 'Cut (Move)' from the 'Edit' menu and 'Paste' into the new location.
The shortcut keys for these options are:
CTRL+V - Paste
CTRL+X - Cut

Q: Can I customize the Extras bar?
Yes. Bring up the bar and tap on the word 'Extras' (or press Enter) to reveal the preferences dialog. Also accessible through the 'Extras bar' control panel on the system screen.

Q: Whenever I open the case my MC 218 is on!
This is the correct behaviour of the machine; it turns on as you open it. This is a preference in the Switch on/off control panel in the system screen.

15. Index

Compatibility and PC Synchronization, 9

Compatible phones, 9
Digital Camera Compatibility, 9
EPOC Compatibility, 10
PC Software Compatibility, 9
PC Synchronization, 10

EPOC, 20

Applications, 33
Applications - EPOC Connect conversions and synchronization, 40
Applications - Introduction, 33
Applications - Technology, 36
Applications - User perspective, 33

Communications, 40
Communications - Components, 51
Communications - Design, 49
Communications - Dial-up networking, 54
Communications - Introduction, 41
Communications - Messaging, 45
Communications - Serial communications, 41
Communications - Sockets and TCP/IP, 43
Communications - Telephony, 42
Communications - WEB, 48

Core, 21
Core - Base, 22
Core - Engine Support, 26
Core - Graphic, 28
Core - GUI and System, 29
Core - Introduction, 22

EPOC R5 New Features, 57
EPOC R5 New Features - Changes by component, 58
EPOC R5 New Features - Introduction, 57
EPOC R5 New Features - Pervasive changes, 57

Introduction, 21

EPOC Software Development, 67

OPL SDK, 67
EPOC SoftwareDevelopment
SDK, 67

Ericsson Mobile Internet, 17
Free access, 17

FAQ, 100

Applications, 101
Communication & Synchronisation, 104
General, 100
Hardware, 106
Hints & Tips, 107
Record, 108
System, 109

GSM Data, 74

General Features of GSM Data, 74
GSM & GSM Data, 74
How does GSM data work?, 77
Key Benefits, 74

Hardware and technical details, 6

Design, 7
Interfaces, 7
Power consumption, 8
Technical Data, 6

IMAP4 functionality, 18

Automatic resynchronisation, 19
Basic functionality, 18
Connect, 18
Glossary, 20
Limitations and omissions, 19
MIME & MTM, 19
Simultaneous server access, 19

Infrared communication, 68

Communicationsolution, 68
Connect, 69
Connection beetween
companion and phone, 68
Ericsson Infrared Modem, 69
IrDA, 70
Use, 68

IrDA, 71

Asymmetric MAC, 73
Characteristics of IrDA, 71
Characteristics of IrDA Control, 72
Download specifications, 73
IrDA CONTROL, 72
IrDA DATA -
Hardware/Protocol Stacks, 72
IrDA Data Protocols, 71
IrDA Infrared features, 73
Optional IrDA Data Protocols, 72
Recommended use, 71

Mobile Internet -
WAP Browser, 15

Access information, 15
Features, 15
Use, 15

Purpose, 5

Serial Terminal Support Comms, 79

Commands, 86
Glossary of commands, 85
How script commands are used, 80
Script commands, 83
To store information as a name, 81

Third Party Software, 16

Find EPOC applications, 16