

---

# **S60 Platform: Porting from 2nd to 3rd Edition**

**Version 1.1**  
May 4, 2006

**S60** platform

## Legal notice

Copyright © 2005, 2006 Nokia Corporation. All rights reserved.

Nokia and Nokia Connecting People are registered trademarks of Nokia Corporation. Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. Other product and company names mentioned herein may be trademarks or trade names of their respective owners.

### Disclaimer

The information in this document is provided “as is,” with no warranties whatsoever, including any warranty of merchantability, fitness for any particular purpose, or any warranty otherwise arising out of any proposal, specification, or sample. Furthermore, information provided in this document is preliminary, and may be changed substantially prior to final release. This document is provided for informational purposes only.

Nokia Corporation disclaims all liability, including liability for infringement of any proprietary rights, relating to implementation of information presented in this document. Nokia Corporation does not warrant or represent that such use will not infringe such rights.

Nokia Corporation retains the right to make changes to this specification at any time, without notice.

### License

A license is hereby granted to download and print a copy of this specification for personal use only. No other license to any other intellectual property rights is granted herein.

## Contents

<b>1.</b>	<b>Introduction .....</b>	<b>6</b>
<b>2.</b>	<b>Application file locations.....</b>	<b>7</b>
<b>3.</b>	<b>Application resources.....</b>	<b>8</b>
<b>4.</b>	<b>Icons and bitmaps.....</b>	<b>9</b>
<b>5.</b>	<b>Registration file .....</b>	<b>10</b>
<b>6.</b>	<b>Application security .....</b>	<b>11</b>
<b>7.</b>	<b>Example porting process .....</b>	<b>12</b>
7.1	Example porting iteration .....	12
<b>8.</b>	<b>Application build changes.....</b>	<b>14</b>
8.1	Application code change.....	14
8.1.1	Application ConstructL() .....	14
8.1.2	Application entry point .....	15
8.2	S60 BLD.INF project file changes.....	16
8.3	S60 MMP project file changes .....	16
8.3.1	Target type .....	17
8.3.2	Application UID (UID3).....	17
8.3.3	Security ID .....	18
8.3.4	Stack size .....	18
8.3.5	Resources.....	18
8.3.6	Registration .....	19
8.3.7	Icons .....	19
8.3.8	Bitmaps.....	19
8.3.9	Vendor ID.....	19
8.3.10	Capability .....	19
8.4	Application RSS resource file changes.....	20
8.5	Application registration file addition .....	20
8.6	Icons building .....	21
8.7	PKG file changes .....	21
8.7.1	Application UID.....	21
8.7.2	Vendor names and logos.....	22
8.7.3	Platform productUID .....	22
8.7.4	Registration file paths .....	22
8.8	Signing the installation package .....	22
8.9	Avoiding hard-coded paths .....	23
<b>9.</b>	<b>Terms and abbreviations .....</b>	<b>24</b>
<b>10.</b>	<b>Further reading.....</b>	<b>25</b>

<b>Appendix A.</b>	<b>Code example .....</b>	<b>26</b>
<b>Appendix B.</b>	<b>Commonly used functions that require capabilities .....</b>	<b>37</b>
<b>Appendix C.</b>	<b>Commonly used interfaces that have been changed or removed .....</b>	<b>39</b>
<b>Evaluate this resource.....</b>		<b>42</b>

## Change history

December 12, 2005	Version 1.0	Initial document release
May 4, 2006	Version 1.1	Sections 8.3.2 (Application UID), 8.7 (PKG file changes), 8.9 (PathInfo), and Appendices B and C added.

---

## 1. Introduction

This document provides guidelines and describes how to easily port S60 2nd Edition C++ applications to S60 3rd Edition. The document has been written based on experiences of porting regular S60 2nd Edition applications, such as the [S60 Platform: POP/IMAP Example](#) [4] that can be downloaded from [Forum Nokia](#). Code snippets from the example are shown in Chapter 8, "Application build changes," and in Appendix A, "Code example." In addition, Appendix B, "Commonly used functions that require capabilities," and Appendix C, "Commonly used interfaces that have been changed or removed," provide useful information on some frequently used functions and interfaces in third-party applications.

There are a small number of user APIs that differ between S60 2nd and 3rd Editions. The main changes are in the application base constructor (for skin changes), new application entry point, target type, application resource, registration, and security.

The following changes have to be made in the project files and code files for regular applications:

- BLD.INF project file changes for icon and bitmap building via extension makefiles;
- MMP project specification file changes for target name, secure ID, vendor ID, capabilities, resources, and registration files for localizable resources and application registration;
- Application base constructor and entry point code changes;
- RSS resource file changes;
- The new REG registration file.

Note that in addition to these mandatory changes, further changes to the source code might be needed for more complex applications. The changes can even affect the application logic in a way that S60 2nd and 3rd Editions need separate source code. Because these changes are difficult to generalize, they are beyond the scope of this document.

## 2. Application file locations

The application architecture defines the framework for the policy for file use. The UI libraries and shell programs implement this policy.

Since policy is a property of the GUI, most application programs do not need to create any code to implement it. The following list gives the locations for most common file types in S60 3rd Edition:

File locations		
Emulator: \epoc32\release\ <platform&gt;\udeb\ </platform&gt;\udeb\  Devices: \sys\bin\	.exe, .dll, .ctl, .mdl, .ldd, .pdd...	Location of all binaries.
Emulator and devices: \private\ <processid&gt;\< td=""> <td>.doc, .txt, .xml, .dat, .ini, .mbm, .rsc, .mif</td> <td>Location of all private data files of the application.  The directory name under \private is the application's SecureID or UID3 if the SecureID is not specified.  Other programs without capability AllFiles are not able to see these files.</td> </processid&gt;\<>	.doc, .txt, .xml, .dat, .ini, .mbm, .rsc, .mif	Location of all private data files of the application.  The directory name under \private is the application's SecureID or UID3 if the SecureID is not specified.  Other programs without capability AllFiles are not able to see these files.
Emulator or build in ROM: \private\10003a3f\apps\ Devices installation: \private\10003a3f\import\apps\  \resource\apps\	<appname>_reg.rsc	The private system directory for application registration files.  The directory must always be on the same drive as the application.  Paths are the same for both the emulator and the target device.
	.rsc, .mbm, .mif	Read access is allowed for all applications.
Emulator: \Epc32\data\Z\Resource\Plugins\ Device: \Resource\Plugins\	<plugin_name>.rsc	This is a special folder meant only for ECom plug-in resources.

Note that most of these folders are accessible at run time only if the application has AllFiles capability (which grants access to all files in the system.)

### 3. Application resources

A resource file is available to an application at run time, and it usually has the extension `.rsc` unless localized resource files (`.r01` etc.) are used. A resource file contains resources compiled from an `.rss` source file. The resources are used in defining the application user interface.

In S60 3rd Edition, the resource file is defined inside the `START RESOURCE – END` statement block in an MMP project file.

In S60 3rd Edition, platform security requires that the application's resources must be compiled into the correct path. The target path is `\resource\apps\` for public read-only resources and `\private\<appSecureID>\` for the application's private resources. This is why the keyword `TARGETPATH` must always be used inside the `START RESOURCE – END` statement in an MMP project file in S60 3rd Edition for resource building, although this statement is normally optional.

The optional `HEADER` keyword creates a resource header file `.rsg` to the `\epoc32\include\` directory. The header file is needed when the application or some other program uses resource definitions and structures.

Traditionally, the application framework requires that menu bars and dialogs are defined in a UI resource file. But in the S60 platform, the caption and icon are also defined either in a special resource file called a localizable icon/caption resource file or, more commonly, in the application UI resource file. The localization will be done through the `LOCALISABLE_APP_INFO` resource structure. The header `appinfo.rh` includes the required definitions for both nonlocalizable and localizable resources.

The application UI resource file and/or special icon/caption definition resource file may be used in the following ways:

- If an application has no UI resources but requires icons/caption, then a special icon/caption resource file could be used.
- An application with UI resources that do not need to be localized and that require multiple captions would use a single UI resource file but several language-dependent icon/caption definition files.
- An application with localizable UI resources and different captions for different languages may be better suited to a UI resource file, rather than using a special icon/caption definition resource file.

---

## 4. Icons and bitmaps

Icons are used to represent applications and their associated document files in the system shell or application launcher. If the device UI supports embedding, icons may also be used to represent embedded documents.

S60 3rd Edition provides a new API for the purpose of loading icons. This new API supports loading of both old bitmap icons (from MBM) and scalable icons (from MIF files). The MIF files are located in the same folders as the MBM files.

The source icon and multi-bitmap files may be `.bmp` bitmaps, and vector graphics can be of `.svg` format. If bitmaps are used, they are built into a single `.mbm` (multi-bitmap file) as part of the build process. Correspondingly, the vector graphics sources are built into the `.MIF` (multi-icon file) file.

Some applications may need to localize their icons. Localization is done by first defining them in `.rls` files. After that, the definitions must be added to the `LOCALISABLE_APP_INFO` structure in the application UI resource file or special icon/caption definition resource file. The icon file names should be referred to by their symbolic identifiers rather than as the strings themselves.

The MIFCONV utility is not yet integrated into the build process, that is, it will not be executed from the project's main makefile. Instead, a special *extension makefile* is needed for invoking MIFCONV to build image resources. This makefile is added to `bld.inf` using the *gnumakefile* command. See the [Forum Nokia Technical Library](#) for more details.

## 5. Registration file

The application registration file defines information about an application that is required by the application launcher or system shell. This includes the name of the application executable, its UID, and several (optional) properties defining launch options, supported data types, etc. In addition, the location of a localizable resource structure (for icon and caption data) can be specified here. S60 2nd Edition supports AIF files, but S60 3rd Edition supports only registration files (in S60 2nd Edition, Feature Pack 3, both AIF and registration files are supported).

The name of the registration file is by convention the same as the application's name but with a `_reg.rss` suffix, for example `<appname>_reg.rss`. The location of registration files in the emulator is in the `\private\10003a3f\apps\` directory on the same drive as the application. In the target device, the registration file's location is the `\private\10003a3f\import\apps\` directory.

Every application needs a registration file, even if it has default properties, icons, and a caption. At the very least, the application UID and the executable name of the application (without the extension) must be specified.

The registration file includes both nonlocalizable and localizable information. The nonlocalizable information never varies according to the language settings, for example, embeddable, hidden, or a list of files owned by the application.

Nonlocalizable properties use an `APP_REGISTRATION_INFO` resource structure inside the registration file. The header file `appinfo.rh` includes these properties. The registration file also specifies the location of the localizable definitions if the application needs them, for example, the name and path of a localization resource file without extension and the ID of a `LOCALISABLE_APP_INFO` resource structure. `Localizable_resource_id` is not needed if a separate `_loc` resource file is used, since it contains only one resource.

---

## 6. Application security

Platform security was introduced in S60 3rd Edition (in Symbian OS v9.1), and it is based on the idea of capabilities — that is, every application and process has a set of capabilities that never will be changed.

Capabilities are divided into two categories: system and user capabilities. Regular applications can use only user capabilities. An application's capabilities can be set into the MMP project file with the statement `CAPABILITY <list of capabilities>`, or `CAPABILITY NONE` if they are not needed.

An application's capabilities depend on the kind of OS resources being used. The application's capability checking can be done in the emulator by setting the `PlatSecDiagnostics` flag ON (it is in the `epoc.ini` file in the `\epoc32\data\` directory). The platform security checking writes missing capabilities for every process in the `epocwind.out` file in the user's workstation temp directory.

For more information, see [S60 Platform: Symbian Platform Security FAQ](#) [6] on the Forum Nokia Web site.

## 7. Example porting process

Figure 1 illustrates the porting process from S60 2nd Edition to S60 3rd Edition.

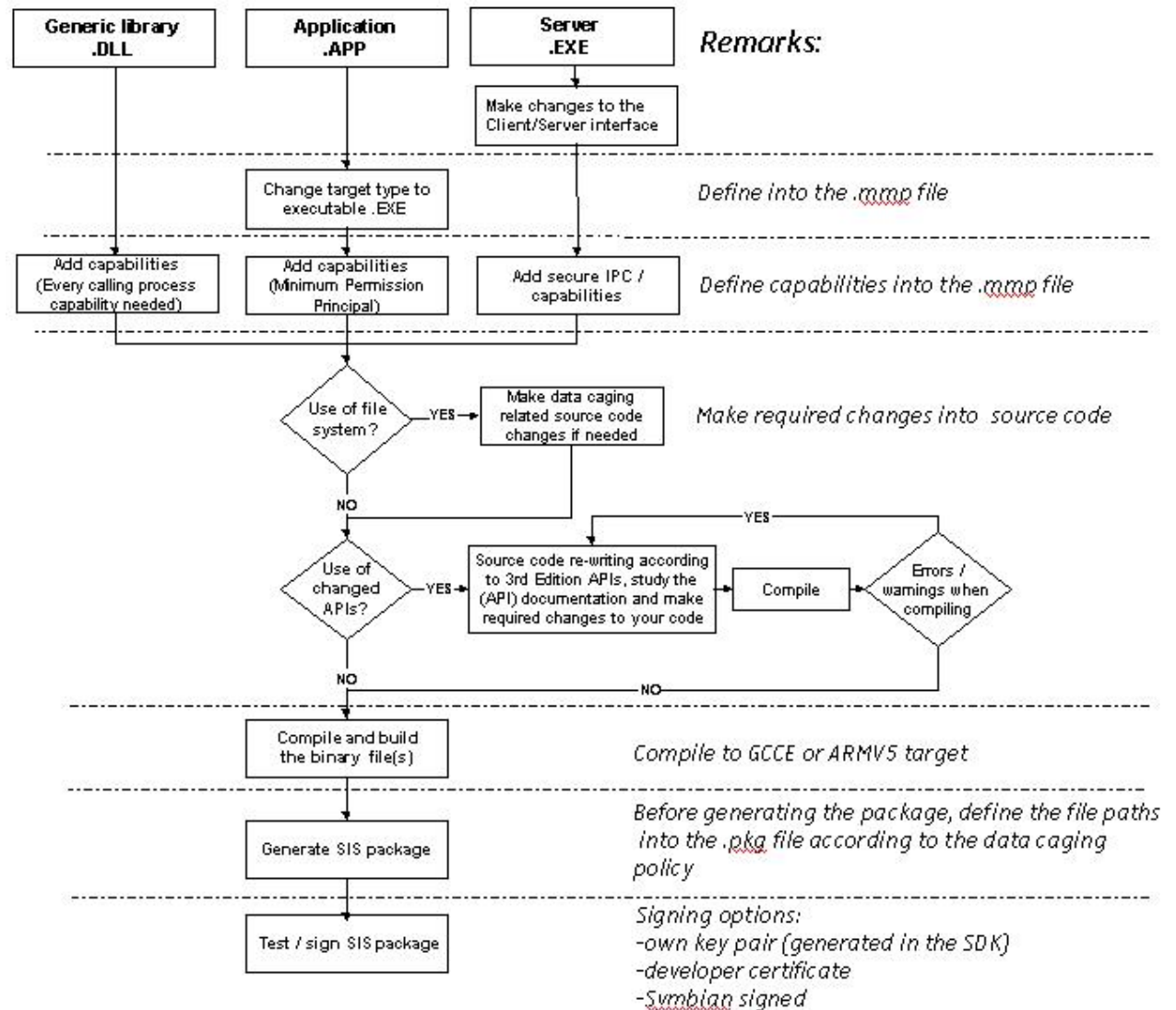


Figure 1: Example porting

### 7.1 Example porting iteration

The following porting iteration process varies depending on the application's complexity:

1. Make necessary known changes in the project files and source code files.
2. Compile project.
3. Study possible errors and correct them one by one.
4. Evaluate if it is possible to use the same code in both S60 2nd and 3rd Editions (multiplatform example).

5. Make changes until the compilation is OK.
6. Build the project to target device.
7. Create the PKG package file.
8. Make the install package.
9. Sign the install package.

In simple cases, it is enough to do iteration steps 1->2-6->7-8-9, but in more complicated multiplatform porting code changes are probably needed and the code flow will need to be controlled by making platform-dependent macro definitions. In these cases, the porting process will probably include all of the iteration steps listed above, for example, 1 -> 2-3-4-5 or 2-3-4-5-6 -> 7-8-9.

One thing to notice is that only those capabilities that the application really requires should be defined. To make it easier to find out the necessary capabilities, the `PlatSecDiagnostics` flag can be set ON in the `epoc.ini` file to receive platform security diagnostics messages to the trace/debug output file and/or an IDE's system log view.

## 8. Application build changes

This chapter briefly describes what kinds of changes must be done in application code files and application project files to port an application from S60 2nd Edition to S60 3rd Edition.

### 8.1 Application code change

The application source code must be changed in two places at minimum: the base constructor and the application entry point.

#### 8.1.1 Application ConstructL()

When using certain Avkon UI components (for example, `CAknForm`), skin support must be enabled in the application source code to ensure correct rendering of the UI components. Skin support is enabled in the application UI class (derived from `CAknAppUi`) method `ConstructL()` by passing an `EAKnEnableSkin` flag to the constructor of the base class.

Though the skin issue was not detected until S60 2nd Edition, Feature Pack 3, this flag can be used without problems from S60 2nd Edition onwards.

The old `BaseConstructL()` call:

```
void <ClassName>::ConstructL()
{
    BaseConstructL();
    ...
}
```

must be changed to:

```
void <ClassName>::ConstructL()
{
    BaseConstructL(EAKnEnableSkin);
    ...
}
```



**Note:** In S60 2nd Edition, Feature Pack 3, the compatibility mode (targeted at 176 x 208 legacy applications that would not scale) can be enabled or disabled with flags. Applications that do not use `ELayoutAwareAppFlag` are run in compatibility mode. The layout aware flag is used by default in the S60 2nd Edition SDK supporting Feature Pack 3, and applications built using this SDK are not run in compatibility mode.

If the application is compiled with an earlier SDK than the S60 2nd Edition SDK supporting Feature Pack 3, flags passed to `BaseConstructL` can be defined manually:

```
#define KEnableSkinFlag 0x1000
#define KLayoutAwareFlag 0x08
```

If compatibility mode is needed, use `KEnableSkinFlag` only. Otherwise, use a combination of the flags, `KEnableSkinFlag | KLayoutAwareFlag`.

These definitions can be used in S60 1st and 2nd Editions without problems. In particular, `KEnableSkinFlag` should be used when using the S60 1st Edition SDK, if 2nd Edition devices are targeted as well.

If manual flag definitions as described above are used, S60 3rd Edition still requires `EAKnEnableSkin`, because the enumeration values have changed. The S60 3rd Edition-specific branch can be defined using the `__SERIES60_30__` macro as follows:

```
#define KEnableSkinFlag 0x1000
#define KLayoutAwareFlag 0x08
void <ClassName>::ConstructL()
{
    #ifdef __SERIES60_30__
        BaseConstructL(EAKnEnableSkin);
    #else
        BaseConstructL(KEnableSkinFlag | KLayoutAwareFlag);
    #endif
    ...
}
```

These examples describe situations where the same application source code is used for multiple S60 platform versions. In these cases the flags defining skin support and layout awareness must be carefully defined.

### 8.1.2 Application entry point

Instead of the S60 2nd Edition application entry point:

```
GLDEF_C TInt E32Dll( TDllReason )
{
    return KErrNone;
}
```

In S60 3rd Edition, EXE applications need an `E32Main()` function as an entry point. The application is started using the `RunApplication` method from `eikstart.h`.

```
#ifdef __SERIES60_30__
#include <eikstart.h>
#endif

...

#ifdef __SERIES60_30__
GLDEF_C TInt E32Main()
{
    return EikStart::RunApplication( NewApplication );
}
# else
GLDEF_C TInt E32Dll( TDllReason )
{
    return KErrNone;
}
#endif
```

## 8.2 S60 BLD.INF project file changes

In S60 3rd Edition, the icon and bitmap building has to be added into the BLD.INF as an extension makefile because the new build tool MIFCONV cannot be started using the MMP project file alone.

Because EKA2 kernel architecture is supported from S60 3rd Edition onwards, the EKA2 macro can be used to separate S60 3rd Edition-specific icon makefile(s) and the MMP file from earlier platforms as follows:

```
PRJ_MMPFILES

#ifdef EKA2          // S60 3rd Ed
gnumakefile <appname>_icons.mk
<appname>_3rd.mmp
#else                // S60 1st and 2nd Ed
<appname>_2nd.mmp
#endif
```

## 8.3 S60 MMP project file changes

This section describes changes in the MMP project file for a typical GUI application. Almost all changes are mandatory, but some, for example, `EPOCSTACKSIZE`, are optional. Note that separate MMP files can be created for S60 2nd and 3rd Editions, and the BLD.INF file can be used to define which one to use in each platform, as shown in Section 8.2, "S60 BLD.INF project file changes."

### 8.3.1 Target type

In S60 3rd Edition, the GUI application target type is `exe`. This is specified with the `TARGETTYPE` and `TARGET` statements.

The old target type

```
TARGETTYPE app
```

and

```
TARGET <appname>.app
```

must be changed to

```
TARGETTYPE exe
```

and

```
TARGET <appname>.exe
```

### 8.3.2 Application UID (UID3)

The ranges for application UIDs (UID3) have changed in Symbian OS v9.1. Earlier-used UID ranges — `0x1000000` to `0x1FFFFFFF` (allocated) and `0x00000000` to `0x0FFFFFFF` (for development use only) — have been replaced with new ranges of UIDs. UIDs have also been split into two ranges: protected and unprotected.

When an application is to be **Symbian Signed** (either during development using a Developer Certificate or via the Symbian Signed program), it must use a UID from the protected range. New UID allocations are made from the **2 range** (`0x20000000` to `0x2FFFFFFF`). Before being submitted for Symbian Signed testing, an application must have a UID allocated from the 2 range.

When an application is **self-signed**, it must use a UID from the unprotected range. New UID allocations are made from the **A range** (`0xA0000000` to `0xAFFFFFFF`). A self-signed application must have a UID allocated from the A range before being released to end users. Alternatively, it is also possible to use your existing allocations for self-signed applications on Symbian OS v9.1 (that is, those applications not requiring access to sensitive APIs); to do this, simply replace the first hex digit (a 1) with F.

For development and testing purposes, E range UIDs (`0xE0000000` to `0xEFFFFFFF`) or allocated A range UIDs from the unprotected range can be used. If a 0 range UID is used, a Developer Certificate is required so it may not be a feasible option.

The old UID3 (for example, `0x101FF1C8`)

```
UID          0x100039CE 0x101FF1C8
```

must be changed to

```
UID          0x100039CE 0x2xxxxxxxx
```

(allocated UID for a Symbian Signed application)

or

```
UID                0x100039CE 0xAxxxxxxxxx
```

(allocated UID for a self-signed application)

or

```
UID                0x100039CE 0xF01FF1C8
```

(Legacy UID for a self-signed application; transposed from the 1 range UID)

For more information on application UIDs, go to [www.symbiansigned.com](http://www.symbiansigned.com).

### 8.3.3 Security ID

It is possible to define a security ID in an MMP project file by adding a `SECUREID` switch to an MMP file. Security ID defines, for example, an application's private data directory. If the security ID is not defined, it defaults to the application UID (UID3).

Add

```
SECUREID           <appID3>
```

or

```
SECUREID           <some proper value>
```

### 8.3.4 Stack size

By default the application has a stack size of 8 kB, which can be insufficient for some applications. The stack size can be increased by using the `EPOCSTACKSIZE` statement.

```
EPOCSTACKSIZE 0x5000
```

The statement above sets the stack size to 20 kB. Any other size can be used as well, and this statement is optional.

### 8.3.5 Resources

In S60 3rd Edition, `START RESOURCE - END` resource statements are used to specify the application's UI resource file.

The resource should be built into the `\resource\apps\` directory. This is specified using the `TARGETPATH` statement.

The `HEADER` line tells the resource compiler to produce a file `\epoc32\include\<appname>.rsg` that defines macro constants from which the application can refer to resource structures.

The old statements

```
TARGETPATH        \system\Apps
```

```
RESOURCE <appname>.rss
```

must be changed to

```
START RESOURCE <appname>.rss
```

```
HEADER
```

```
TARGETPATH    \resource\Apps
```

```
END
```

so that the resource is included inside the `START RESOURCE - END` frame.

### 8.3.6 Registration

The registration file that takes care of the application registration must always be built to the private system directory `\private\10003a3f\apps\`.

The application registration is a special resource file, and it also uses the resource build `START RESOURCE - END` statements frame.

```
START RESOURCE <appname>_reg.rss
```

```
TARGETPATH    \private\10003a3f\apps
```

```
END
```

Note that for security reasons a registration file cannot be installed directly to this location on a target device (phone). When the application software installation package file PKG is created, the registration file must be installed to the `\private\10003a3f\import\apps\` directory.

### 8.3.7 Icons

The icons are no longer built in the S60 platform MMP project file. Instead, an extension makefile is used, which is started from the BLD.INF project file.

### 8.3.8 Bitmaps

Bitmaps are no longer built in the S60 platform MMP project file. Instead, an extension makefile is used, which is started from the BLD.INF project file.

### 8.3.9 Vendor ID

This keyword specifies the vendor of the application. In most cases this VID will be zero, meaning that the source of the example executable is not required for any security checks, for example:

```
VENDORID 0
```

### 8.3.10 Capability

Some applications will require additional capability. It is possible to add new capabilities into the MMP project file by using the `CAPABILITY <capability list>` statement, for example:

```
CAPABILITY ReadUserData
```

or

```
CAPABILITY NONE
```

if capabilities are not needed.

## 8.4 Application RSS resource file changes

Resource building for most S60 3rd Edition applications is similar to building for S60 2nd Edition applications. However, there is one addition to S60 3rd Edition concerning the localization of the application — it will be done by adding a `LOCALISABLE_APP_INFO` <resourceID> statement into the application UI resource file.

The new structure is added:

```
#include <appinfo.rh>

...

RESOURCE LOCALISABLE_APP_INFO r_<appname>_localisable_app_info
{
    short_caption = rls_app_short_caption_string;
    caption_and_icon =
    CAPTION_AND_ICON_INFO
    {
        caption = rls_app_caption_string;
        number_of_icons = 2;
        icon_file = "\\resource\\apps\\<appname>.mbm";
    };
}
```

## 8.5 Application registration file addition

A new registration file must be added into the project because the S60 3rd Edition system launcher needs it to load the application.

Typically the name of the application registration file is the name of the application, plus the `_reg` suffix, for example, `<appname>_reg.rss`.

The registration file includes the `APP_REGISTRATION_INFO` resource structure and the `appinfo.rh` header file in which registration properties are declared.

In the application registration file, `UID2` always has the value `KUidAppRegistrationResourceFile` and `UID3` has the application `UID3` (unique `UID`).

The `app_file` property has the name of the application without an extension. The `newfile` property has the default value `KAppDoesNotSupportNewFile` and `embeddability` has a default value `KAppNotEmbeddable`, for example:

```
<appinfo.rh>

UID2 KUidAppRegistrationResourceFile
UID3 <appUID3>

RESOURCE APP_REGISTRATION_INFO
{
    app_file= "<appname>";
```

```

newfile = KappDoesNotSupportNewFile;
embeddability = KappNotEmbeddable;
localisable_resource_file = "<appResourceName without extension>";
// needed only if localisable_resource_file points to main UI
// resource file
localisable_resource_id = <appResourceInfoID>;
}

```

## 8.6 Icons building

S60 3rd Edition provides a new API for the purpose of loading icons. This new API supports loading of both old bitmap icons (from MBM) and scalable icons (from MIF files). The MIF files are located in the same folders as the MBM files.

In S60 3rd Edition, icons and bitmaps are built with the MIFCONV tool. It cannot be started directly from an MMP project file, but needs a special MK make file, which will be started in the BLD.INF project file.

For example,

BLD.INF project file:

```

PRJ_MMPFILES

gnumakefile <appname>_aif.mk

icon/bitmap makefile:
...
feq (WINS,$(findstring WINS, $(PLATFORM)))
ZDIR=$(EPOCROOT)epoc32\release\$(PLATFORM)\$(CFG)\Z
else
ZDIR=$(EPOCROOT)epoc32\data\z
endif

...

TARGETDIR=$(ZDIR)\RESOURCE\APPS
ICONTARGETFILENAME=$(TARGETDIR)\<appname>.mbm
# ICONTARGETFILENAME=$(TARGETDIR)\<appname>.mif

...

RESOURCE :

    mifconv $(ICONTARGETFILENAME) /c12,1 ..\aif\list_icon.bmp /c12,1
    ..\aif\context_pane_icon.bmp

```

## 8.7 PKG file changes

### 8.7.1 Application UID

Use the correct UID (see Section 8.3.2,"Application UID (UID3)").

### 8.7.2 Vendor names and logos

Add the vendor names.

% specifies the localized vendor names — one per language. At least one name must be provided. The list must correspond to the list of languages specified elsewhere in the PKG file.

: specifies the nonlocalized, globally unique vendor name (mandatory) that is used, in combination with signing, to prevent the unauthorized upgrade of a package by someone other than the rightful vendor.

= specifies optional logo files (corresponding to language):  
 <logfile>,<mimetype>[,<targetname where the file will be installed>]

```
%{"Vendor-EN", "Vendor-FR"}
:"Vendor"
="file\mylogo.jpg","image/jpeg","\public\logos\mylogo.jpg"
```

### 8.7.3 Platform productUID

Use the productUID of S60 3rd Edition:

```
; Supports S60 v 3.0
; Device hardware, [productUID], version-range, {"ProductName"}
[0x101F7961], 0, 0, 0, {"Series60ProductID"}
```

### 8.7.4 Registration file paths

Specify the location of the registration file (it must be installed to the \private\10003a3f\import\apps\ directory on the target device):

```
"..\..\..\epoc32\data\z\private\10003a3f\apps\InternetEmail_reg.rsc" -"!\private\10003a3f\import\apps\InternetEmail_reg.rsc"
```

## 8.8 Signing the installation package

S60 3rd Edition includes mandatory signing of the SISX installation package before it can be installed into a target device. If a SISX package does not need any platform security capabilities or any capabilities other than NetworkServices, LocalServices, ReadUserData, WriteUserData, and UserEnvironment, it can be signed with a self-created private key. In the device, self-signed applications are regarded as untrusted (which means that the installer warns about it but the application will run). Note that with an untrusted application, the application UID must be in the unprotected range (see Section 8.3.2, "Application UID (UID3)"). For more information on application UIDs, refer to [www.symbiansigned.com](http://www.symbiansigned.com).

If a capability that requires a digital signature is used, a certificate is needed to run the application on a target device. To test such an application, a developer needs to acquire a Symbian Developer Certificate; a Symbian Signed certificate is needed before the application can enter the market.

The installation package file can be signed using the installation file signer SignSIS. Any certificates and private keys are specified in the package file or as command line arguments, for example:

```
SignSIS InternetEmail_v30_gcce.sis InternetEmail_v30_gcce.sis  
DevCert.cer Access.key 12345678
```

where:

1. DevCert.cer is a target device installation certificate file.
2. Access.key is private key file of the certificate.
3. 12345678 is a pass phrase of the certificate's private key file.

For more information, see the "How to Sign sis Files" document in the SDK's `S60Doc` folder.

## 8.9 Avoiding hard-coded paths

There are many considerations when writing portable code. One thing is to avoid hard coding, especially with file paths. `PathInfo` has been available since S60 2nd Edition, and it should be used to get the paths for storing different media file types (audio and video clips, installation files, etc.), as well as root paths for phone memory and a memory card. For example, `PathInfo` provides static functions like `MemoryCardRootPath()` and `PhoneMemoryRootPath()`, which should be used instead of hard coding the drive letter (in these cases "C:" or "E:").

Using `PathInfo` is also recommended to avoid possible problems between S60 devices from different manufacturers, because there may be differences in directory naming among the licensees.

## 9. Terms and abbreviations

Term or abbreviation	Meaning
AIF	Application Information File.
API	Application Programming Interface.
BMP	Microsoft Windows bitmap format.
GUI	Graphical User Interface.
MBG	The bitmap header file. It contains an ID for each bitmap within the corresponding .mbm.
MBM	Multi-bitmap. A bitmap (file) containing multiple bitmaps.
MIF	Multi-Icon File. Collection file for scalable icons.
MifConv.EXE	Tool for generating icon files.
MMP	Symbian makmake project file.
PKG	A text file containing instructions that are used by Software Install.
RLS	Resource localization file.
RSC	Resource target file. A resource file containing compiled resources.
RSS	Resource Source file.
RH	Resource Header, for inclusion in a resource file.
RSG	Resource header file. It contains an ID for each resource within the corresponding .rsc.
SVG	Scalable Vector Graphics.
UI	User interface.

---

## 10. Further reading

For more information, see the following resources available at [www.forum.nokia.com](http://www.forum.nokia.com):

- [1] [S60 2nd/3rd Edition: Differences in Features](#)
- [2] [S60 3rd Edition: Tool Chain, IDEs, and Development Process](#)
- [3] [S60 3rd Edition: What's New for Developers](#)
- [4] [S60 Platform: POP/IMAP Example](#)
- [5] [S60 Platform: Source and Binary Compatibility](#)
- [6] [S60 Platform: Symbian Platform Security FAQ](#)
- [7] [Testing and Signing with Symbian Platform Security](#)

---

## Appendix A. Code example

### A.1 BLD.INF project file

```
/*
*
=====
* Name      : bld.inf
* Part of   : InternetEmail
* Created   : 25.10.2005 by Forum Nokia
* Description:
*   This file provides the information required for building the
*   whole of a InternetEmail.
* Version   :
* Copyright: Nokia Corporation 2004
*
=====
*/

PRJ_MMPFILES

// if EKA2 kernel architecture is supported, assume we're running on
// 3rd Ed
#ifdef EKA2
    InternetEmail_3rd.mmp

    gnumakefile InternetEmail_aif.mk
#else
    InternetEmail_2nd.mmp
#endif

/*
End of file
*/
```

## A.2 3rd Edition MMP project file

```

/*
* =====
* Name      : InternetEmail_3rd.mmp
* Part of   : InternetEmail
* Created   : 25.10.2005 by Forum Nokia
* Description:
*   This is the project specification file for InternetEmail.
*   Initial content was generated by S60 AppWizard.
*
* Version   : 1.0
* Copyright: Nokia Corporation 2004
* =====
*/

TARGET      InternetEmail.exe
TARGETTYPE  exe
UID         0x100039CE 0x2xxxxxxx

SECUREID    0x2xxxxxxx
EPOCSTACKSIZE 0x5000

SOURCEPATH  ..\src
SOURCE      InternetEmailApp.cpp
SOURCE      InternetEmailAppUi.cpp
SOURCE      InternetEmailDocument.cpp
SOURCE      InternetEmailContainer.cpp
SOURCE      InternetEmailEngine.cpp

SOURCEPATH  ..\data

START RESOURCE InternetEmail.rss
HEADER
TARGETPATH  \resource\apps
END //RESOURCE

START RESOURCE InternetEmail_reg.rss
TARGETPATH  \private\10003a3f\apps
END //RESOURCE

LANG        SC

USERINCLUDE . ..\inc

SYSTEMINCLUDE . \epoc32\include

LIBRARY euser.lib
LIBRARY apparc.lib
LIBRARY cone.lib
LIBRARY eikcore.lib
LIBRARY eikcoctl.lib
LIBRARY avkon.lib
LIBRARY commonengine.lib // for StringLoader
LIBRARY msgs.lib        // for MsvServer
LIBRARY mtur.lib        // for CMtmUiReg

VENDORID    0
CAPABILITY  ReadUserData WriteUserData ReadDeviceData WriteDeviceData
NetworkServices

// End of File

```

### A.3 Application entry point

```

/*
*
=====
==
* Name      : CInternetEmailApp from InternetEmailApp.cpp
* Part of   : InternetEmail
* Created   : 25.10.2005 by Forum Nokia
* Implementation notes:
*
*   Initial content was generated by S60 AppWizard.
* Version   : 1.0
* Copyright: Nokia Corporation 2004
*
=====
==
*/

// INCLUDE FILES
#ifdef __SERIES60_30__
#include <eikstart.h>
#endif
#include "InternetEmailApp.h"
#include "InternetEmailDocument.h"

// ===== MEMBER FUNCTIONS =====

// -----
// CInternetEmailApp::AppDllUid()
// Returns application UID
// -----
//
TUid CInternetEmailApp::AppDllUid() const
{
    return KUidInternetEmail;
}

// -----
// CDictionaryStore* CInternetEmailApp::OpenIniFileLC(RFs& aFs) const
// overrides CAknApplication::OpenIniFileLC to enable INI file support
// -----
//
CDictionaryStore* CInternetEmailApp::OpenIniFileLC(RFs& aFs) const
{
    return CEikApplication::OpenIniFileLC(aFs);
}

// -----
// CInternetEmailApp::CreateDocumentL()
// Creates CInternetEmailDocument object
// -----
//
CApaDocument* CInternetEmailApp::CreateDocumentL()
{
    return CInternetEmailDocument::NewL( *this );
}

// ===== OTHER EXPORTED FUNCTIONS =====
//
// -----
// NewApplication()
// Constructs CInternetEmailApp
// Returns: created application object

```

```

// -----
//
EXPORT_C CApaApplication* NewApplication()
{
    return new CInternetEmailApp;
}

#ifdef __SERIES60_30__

// -----
// E32Main()
// Entry point function for EPOC Application
// Returns: EikStart::RunApplication: NewApplication
// -----
//
GLDEF_C TInt E32Main()
{
    return EikStart::RunApplication( NewApplication );
}

#else

// -----
// E32D11(TD11Reason)
// Entry point function for EPOC Apps
// Returns: KErrNone: No error
// -----
//
GLDEF_C TInt E32D11( TD11Reason )
{
    return KErrNone;
}

#endif

// End of File

```

## A.4 RSS resource file

```

/*
*
=====
==
* Name      : InternetEmail.rss
* Part of   : InternetEmail
* Created   : 25.10.2005 by Forum Nokia
* Description:
*   This file contains all the resources for the InternetEmail.
*   Initial content was generated by S60 AppWizard.
* Version   : 1.0
* Copyright: Nokia Corporation 2004
*
=====
==
*/

// RESOURCE IDENTIFIER
NAME      INEM // 4 letter ID

// INCLUDES

#include <eikon.rh>
#include <avkon.rh>
#include <avkon.rsg>
#include <avkon.mbg>
#ifdef EKA2
#include <appinfo.rh>
#endif
#include "internetemail.hrh"
#include "internetemail.rls"

// CONSTANTS

// MACROS

// RESOURCE DEFINITIONS
RESOURCE RSS_SIGNATURE { }

RESOURCE TBUF { buf="InternetEmail"; }

// -----
//
//   Define default menu and CBA key.
//
// -----
//
RESOURCE EIK_APP_INFO
{
    hotkeys=r_internetemail_hotkeys;
    menubar=r_internetemail_menubar;
    cba=R_AVKON_SOFTKEYS_OPTIONS_BACK;
}

#ifdef EKA2

//-----
//
//   r_internetemail_localisable_app_info
//
//-----

```

```

//
RESOURCE LOCALISABLE_APP_INFO r_internetemail_localisable_app_info
{
    short_caption = rls_app_short_caption_string;
    caption_and_icon =
    CAPTION_AND_ICON_INFO
    {
        caption = rls_app_caption_string;
        number_of_icons = 2;
        icon_file = rls_app_icon_name_string;
    }
};

#endif

//-----
//
//    r_internetemail_hotkeys
//    ?description
//
//-----
//
RESOURCE HOTKEYS r_internetemail_hotkeys
{
    control=
    {
        HOTKEY { command=EAknCmdExit; key='e'; }
    };
}

//-----
//
//    r_internetemail_menubar
//    ?description
//
//-----
//
RESOURCE MENU_BAR r_internetemail_menubar
{
    titles=
    {
        MENU_TITLE { menu_pane=r_internetemail_menu;
txt=rls_app_menu_file_string; }
    };
}

//-----
//
//    r_internetemail_menu
//    Two custom but common commands
//
//-----
//
RESOURCE MENU_PANE r_internetemail_menu
{
    items=
    {
        MENU_ITEM { command=EAknCmdExit;
txt=rls_app_menu_exit_string; },
        MENU_ITEM { command=EInternetEmailCmdFetch;
txt=rls_app_fetch_string; },
        MENU_ITEM { command=EInternetEmailCmdProtocol;
cascade=r_internetemail_protocol_menu_cascade;txt=rls_app_setprotocol_string; }
    };
}

```

```

//-----
//
//   r_internetemail_app_menu
//   Two basic commands which do not need localization
//
//-----
//
RESOURCE MENU_PANE r_internetemail_protocol_menu_cascade
{
  items=
  {
    MENU_ITEM { command=EInternetEmailCmdSetPop;
txt=rls_app_menu_POP3_string; },
    MENU_ITEM { command=EInternetEmailCmdSetImap;
txt=rls_app_menu_IMAP4_string; }
  };
}

//-----
//
//   resource note
//   Custom dialog definition for wait note dialog
//
//-----
//
RESOURCE DIALOG r_wait_note
{
  flags = EAknWaitNoteFlags;
  buttons = R_AVKON_SOFTKEYS_CANCEL;
  items =
  {
    DLG_LINE
    {
      type = EAknCtNote;
      id = EWaitNote;
      control = AVERELL_NOTE
      {
        layout = EWaitLayout;
        singular_label = rls_app_waitnote;
        imagefile = rls_app_sysresource_name_string;
        imageid = EMbmAvkonQgn_note_copy;
        imagemask = EMbmAvkonQgn_note_copy_mask;
        animation = R_QGN_GRAF_WAIT_BAR_ANIM;
      };
    };
  };
}

//-----
//
//   strings
//   Two localized strings
//
//-----
//
RESOURCE TBUF64 R_NO_POP3_DEFINED
{
  buf=rls_app_nopop3defined_string;
}

RESOURCE TBUF64 R_NO_IMAP4_DEFINED
{
  buf=rls_app_noimap4defined_string;
}

RESOURCE TBUF80 R_RESOURCE_NAME_DEFINED
{
  buf=rls_app_resource_name_string;
}

```

```
    }  
RESOURCE TBUF80 R_RESOURCE_BODY_NAME_DEFINED  
    {  
        buf=rls_app_resource_body_name_string;  
    }  
RESOURCE TBUF80 R_ICON_NAME_DEFINED  
    {  
        buf=rls_app_icon_name_string;  
    }  
// End of File
```

## A.5 Registration file \_reg.RSS for 3rd Edition application

```

/*
*
=====
* Name      : internetemail_reg.rss InterneteMail registration file
* Part of   : internetemail
* Created   : 25.10.2005 by Forum Nokia
* Version   : 1.0
* Copyright: Nokia Corporation
*
=====
*/

#include <appinfo.rh>
#include <internetemail.rsg>

UID2 KUidAppRegistrationResourceFile
UID3 0x2xxxxxxxxx

RESOURCE APP_REGISTRATION_INFO
{
    app_file="InternetEmail";

    localisable_resource_file = "\\resource\\apps\\internetemail";
    localisable_resource_id = R_INTERNETEMAIL_LOCALISABLE_APP_INFO;

    embeddability=KAppNotEmbeddable;
    newfile=KAppDoesNotSupportNewFile;
}
/*
End of file
*/

```

## A.6 Application icon and bitmap

```
# Copyright (c) 2005, Nokia. All rights reserved #

ifeq (WINS,$(findstring WINS, $(PLATFORM)))
ZDIR=$(EPOCROOT)epoc32\release\$(PLATFORM)\$(CFG)\Z
else
ZDIR=$(EPOCROOT)epoc32\data\z
endif

TARGETDIR=$(ZDIR)\RESOURCE\APPS
ICONTARGETFILENAME=$(TARGETDIR)\InternetEmail.mbm

do_nothing :
    @rem do_nothing

MAKMAKE : do_nothing

BLD : do_nothing

CLEAN : do_nothing

LIB : do_nothing

CLEANLIB : do_nothing

RESOURCE :

    mifconv $(ICONTARGETFILENAME) /c12,1 ..\aif\list_icon.bmp
/c12,1 ..\aif\context_pane_icon.bmp

FREEZE : do_nothing

SAVESPACE : do_nothing

RELEASABLES :
    @echo $(ICONTARGETFILENAME)

FINAL : do_nothing
```

## A.7 PKG installation file

```

; InternetEmail v30_gcce.pkg
; Installation file for InternetEmail application

; Languages, &lang-code [(dialect-ID)], lang-code [(dialect-ID)], ...
; e.g. &EN,FR,ZU(1024)
&EN

; Header, {component name language 1,...}, {application
UID},major,minor,build
#{"InternetEmail"},(0x2xxxxxxx),1,0,0

; Localised vendor, {language 1, ...}
%{" My Company"}

; Unique Vendor name
:"My Company"

; Supports S60 v 3.0
; Device hardware, [productUID], version-range, {"ProductName"}
[0x101F7961], 0, 0, 0, {"Series60ProductID"}

;Files to install
"..\\..\\..\\epoc32\\release\\gcce\\urel\\InternetEmail.exe"
-":\\sys\\bin\\InternetEmail.exe"
"..\\..\\..\\epoc32\\data\\z\\resource\\apps\\InternetEmail.rsc"
-":\\resource\\apps\\InternetEmail.rsc"
"..\\..\\..\\epoc32\\data\\z\\private\\10003a3f\\apps\\InternetEmail_reg.rsc"
-":\\private\\10003a3f\\import\\apps\\InternetEmail_reg.rsc"
"..\\..\\..\\epoc32\\data\\z\\resource\\apps\\InternetEmail.mbm"
-":\\resource\\apps\\InternetEmail.mbm"

```

## Appendix B. Commonly used functions that require capabilities

This appendix lists commonly used functions that require setting of capabilities from S60 3rd Edition onwards. Functions in the table are the most commonly used functions in the order of frequency (with the exception that functions from the same class are grouped). Data is based on empiric results from a number of third-party applications.

Class / Function	Capability	Capability type	Description
<b>RwsSession</b>			
RWsSession::SendEventToWindowGroup	<b>SWEvent</b>	Extended	Grants the right to generate and capture software key and pen events.
<b>Rprocess</b>			
RProcess::Kill	<b>PowerMgmt</b>	Extended	Grants the right to kill any process in the system or to switch machine state (turn phone off).
RProcess::Terminate	<b>PowerMgmt</b>	Extended	Grants the right to kill any process in the system or to switch machine state (turn phone off).
<b>RmsvServerSession</b>			
RMsvServerSession::CloseMessageServer	<b>WriteDeviceData</b>	Extended	Grants write access to sensitive system data.
<b>User</b>			
User::SetHomeTime	<b>WriteDeviceData</b>	Extended	Grants write access to sensitive system data.
<b>RwindowGroup</b>			
RWindowGroup::CaptureKey	<b>SWEvent</b>	Extended	Grants the right to generate and capture software key and pen events.
RWindowGroup::CaptureKeyUpAndDowns	<b>SWEvent</b>	Extended	Grants the right to generate and capture software key and pen events.
<b>ClogClient</b>			
CLogClient::AddEventType	<b>WriteDeviceData</b>	Extended	Grants write access to sensitive system data.
<b>Tlocale</b>			
TLocale::Set	<b>WriteDeviceData</b>	Extended	Grants write access to sensitive system data.

<b>CContactDatabase</b>			
CContactDatabase::OpenL	<b>ReadUserData</b>	Basic	Grants read access to user data. System servers and application engines are free to grant this restriction level to their data.
CContactDatabase::OpenContactL	<b>WriteUserData</b>	Basic	Grants write access to user data. Again, system servers and application engines are free to grant this restriction level to their data.
CContactDatabase::MatchPhoneNumberL	<b>ReadUserData</b>	Basic	Grants read access to user data. System servers and application engines are free to grant this restriction level to their data.
CContactDatabase::CommitContactL	<b>ReadUserData &amp; WriteUserData</b>	Basic	Grants read access to user data. System servers and application engines are free to grant this restriction level to their data.  Grants write access to user data. Again, system servers and application engines are free to grant this restriction level to their data.
<b>Rline</b>			
RLine::NotifyIncomingCall	<b>NetworkServices</b>	Basic	This capability is, for example, for dialing a number or sending a text message.
RLine::NotifyIncomingCallCancel	<b>NetworkServices</b>	Basic	This capability is, for example, for dialing a number or sending a text message.
<b>Rcall</b>			
RCall::Dial	<b>NetworkServices</b>	Basic	This capability is, for example, for dialing a number or sending a text message.

## Appendix C. Commonly used interfaces that have been changed or removed

This appendix lists commonly used interfaces that have been changed or removed in S60 3rd Edition. Interfaces are sorted in the order of frequency based on empiric results from a number of third-party applications.

Interface in S60 2nd Edition	Corresponding interface in S60 3rd Edition	Notes
<b>HAL</b>		
HAL::Get(HALData::TAttribute, int &)	HAL::Get(TAttribute aAttribute, TInt& aValue)	Parameter change.
<b>CsettingInfo</b>		
CSettingInfo	Settings are stored in different keys in the Central Repository.	CSettingsInfo class in platformenv.dll is removed.
<b>Base E32</b>		
RSessionBase::DoSendReceive RSubSessionBase::DoCreateSubSession	RSessionBase::SendReceive RSubSessionBase::CreateSubSession	"Do" prefixes have been removed.
<b>System Agent</b>		
RSystemAgent RSAVarChangeNotify	Symbian Publish and Subscribe API	sysagt.dll component has been replaced by ekern.dll component.  Symbian OS SDK v9.1 » Symbian OS guide » System libraries » Using System Agent » MIGRATING to Publish And Subscribe.
<b>PLP Variant</b>		
PlpVariant::GetMachineIdL	GetPhoneId	plpvariant.dll component is replaced by etel3rdparty.dll component to get IMEI code.
<b>Descriptors</b>		
TDes::AppendFormat (TRefByValue<TDesC16 const>,...)	TDes::AppendFormat (TRefByValue<TDesC16 const>, TDes16Overflow*, ...)	New nonoptional parameter added.
<b>Camera</b>		
RCameraServ	CCamera	cameraserver.dll component is replaced with ecam.dll component (ecam.dll is supported from S60 2nd Edition onwards).
<b>Shared Data Client</b>		
RSharedDataClient		Removed.

<b>BIO Message Parser</b>		
CBaseScriptParser	CBaseScriptParser2	Parser has to be derived from the new CBaseScriptParser2 class instead of CBaseScriptParser.
<b>NifMan</b>		
RNif RGenericAgent	RConnection	nifman.dll is replaced by esock.dll.  Symbian OS SDK v9.1 » Symbian OS guide » Comms infrastructure » Using Sockets Server (ESOCK) » Using Sockets Client » Connection Management » Connection Management overview.
<b>Messaging Framework</b>		
CMsvDefaultServices	CBaseMtm	CMsvDefaultServices class is replaced by CBaseMtm.
CBaseMtm::CreateAttachmentL (long & TBuf<256> &) CBaseMtm::DeleteAttachmentL	CBaseMtm::CreateAttachmentL (const TDesC &aFileName, RFile &aAttachmentFile, const TDesC8 &aMimeType, TUint aCharset, TRequestStatus &aStatus)  CBaseMtm::CancelAttachmentOperation	Parameter change, name change.
CMsvEntry::GetFilePath CMsvEntry::HasDirectoryL	MMsvAttachmentManager	CMsvEntry and CMsvServerEntry class methods GetFilePath and HasDirectoryL are replaced by MMsvAttachmentManager methods.
<b>SendAs</b>		
CSendAs MSendAsObserver	RSendAs RSendAsMessage	Send As API is provided by the RSendAs and RSendAsMessage classes.
<b>Favourites Engine</b>		
CFavouritesDb CBookmarkDb	RFavouritesHandle RFavouritesDb	CFavouritesDb and CBookmarkDb classes in favouritesengine.dll has been replaced by RFavouritesDb class
<b>E-mail Client MTMs</b>		
ClmAPPPreferences::StoreL	CEmailAccounts	StoreL function has been removed. Use CEmailAccounts interface.
ClmBaseEmailSettings::SetUserAddressL		Function removed from S60 3rd edition.

<b>Agenda Model</b>		
CAgnEntryModel::RegisterObserverL (TBuf<256> const &, CAgnObserver const *)		Agenda Model replaced with Calendar Interim APIs.
CAgnEntryModel::UnregisterObserverL (TBuf<256> const &)		

---

## Evaluate this resource

Please spare a moment to help us improve documentation quality and recognize the resources you find most valuable, by [rating this resource](#).