

---

# **S60 Platform: Symbian Platform Security FAQ**

**Version 2.0**  
April 13, 2006

**S60** platform

## Legal Notice

Copyright © 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>7</b>
<b>2.</b>	<b>General Information about Symbian OS Platform Security .....</b>	<b>8</b>
2.1	What is capability? .....	8
2.2	What is TCB? .....	8
2.3	What is SID? .....	8
2.4	Is the SID generated by the Software Installer? If not, where do I get the SECUREID value for the MMP file? .....	8
2.5	What is UID? .....	8
2.6	What is the difference between SID and UID? .....	9
2.7	Can the SID and UID3 of an application be different? .....	9
2.8	How do SIDs relate to applications consisting of an EXE and a DLL? Are the SIDs of these the same?.....	9
2.9	How is the SID of the calling EXE checked? .....	9
2.10	What is VID? .....	9
2.11	Why is the registration RSS file required? Where should it be located? What information should be specified in it? .....	10
2.12	What are the locations where files used by the application should be installed? .....	10
2.13	What are the changes regarding application files and their locations in the system? ..	10
2.14	What changes are required to change an application from an .APP to an .EXE?.....	11
2.15	Where did 10003a3f come from in the MMP file for the target resource path? .....	13
2.16	Can I use an SID or a VID instead of capability check? .....	13
2.17	Does the SID of a DLL have any run-time significance? .....	13
2.18	How and when are capabilities put in the DLL or EXE? .....	13
2.19	Is it possible to change capabilities during run time? .....	13
2.20	What certifications do the capabilities require?.....	13
2.21	What certificate is needed in the development phase? .....	14
2.22	What does .sis file signing mean? .....	15
2.23	Does the behavior of self-signed applications differ during execution?.....	15
2.24	How does the UID range correspond to Symbian Signed and Developer Certificates? What range should be used during development time and for the final release?.....	15
2.25	Do I need to re-request legacy UIDs from Symbian Signed before I can use them? ...	16
2.26	How do I obtain a Symbian Signed Developer Certificate? .....	16
2.27	Do I have to certify/sign every update of the application? .....	16
2.28	Who is permitted access to the Symbian Signed Catalogue?.....	16
2.29	What is “passive content”?.....	16
2.30	Why are DLL capabilities needed? .....	17
2.31	Does loading a DLL with a different set of capabilities change the process capabilities?.....	17

2.32	Can a process load a DLL with fewer capabilities than itself?.....	17
2.33	How are the capabilities assigned when a process is spawned by another process?.....	18
2.34	How does the EPOCHHEAPSIZE work? .....	18
2.35	How does the EPOCSTACKSIZE work? .....	19
<b>3.</b>	<b>Data Caging .....</b>	<b>20</b>
3.1	What is the idea behind data caging?.....	20
3.2	What is the idea behind using the import directory?.....	20
3.3	Does the AllFiles capability allow reading from/writing to any directory under \sys, \private, and \resource? .....	21
3.4	Can any program read from/write to other directories not under \sys, \private, and \resource? .....	21
3.5	How do I change SYSTEMRESOURCE in the .mmp file? .....	21
3.6	Is the import folder under \private\<SID-of-process>\ writable or accessible by everyone?.....	21
3.7	Can DLLs have their own private directories? .....	21
3.8	How do polymorphic DLLs work? .....	21
<b>4.</b>	<b>Publish &amp; Subscribe .....</b>	<b>23</b>
4.1	What is Publish & Subscribe? .....	23
4.2	What is the difference between Publish & Subscribe and Central Repository? .....	23
4.3	What is the best way to efficiently share a large amount of data between two processes in a secure way?.....	23
<b>5.</b>	<b>Configuring Platform Security Settings .....</b>	<b>24</b>
5.1	How do I turn on/off platform security diagnostics? .....	24
5.2	How do I turn on/off platform security enforcement? .....	24
5.3	How do I disable a specified set of capabilities and what does it mean to disable them? .....	24
5.4	What is the capability if the CAPABILITY keyword doesn't exist in the MMP file? .....	24
5.5	What does "CAPABILITY ALL -TCB" mean?.....	25
5.6	How do I exclude multiple capabilities? .....	25
<b>6.</b>	<b>SW Installation.....</b>	<b>26</b>
6.1	Where can I find information about signing sis files?.....	26
6.2	What are the new directives used in a .pkg file?.....	26
6.3	Does the S60 3rd Edition SW Installer support old SIS files? .....	26
6.4	Where are ECOM registration resource files installed? .....	26
6.5	Where are application registration resource files installed? .....	26
6.6	What should I do if I receive a "Certificate check failed. Contact application supplier" message?.....	27
6.7	I get a "Cannot grant capabilities" message during installation. What does it mean? ..	27
6.8	Which user capabilities can the user grant for an application not certified by a trusted certificate?.....	27

- 6.9 I get an "Unable to verify supplier. Continue anyway?" message during installation. What does it mean? .....28
- 6.10 I get a "Cannot install. Access denied" message during installation. What is the reason? .....28
- 6.11 How should I use the manufacturer identifier in a condition block in a .pkg file? .....28
- 6.12 Can I launch my application during the install/uninstall procedure? .....29
- 7. Testing..... 30**
  - 7.1 How do I change the capabilities of test applications for testing purposes? .....30
- 8. Evaluate This Resource..... 31**

## Change History

September 5, 2005	Version 1.0	Initial document release in Forum Nokia PRO.
November 2, 2005	Version 1.1	Sections 1.2, 1.15, 5.3, 5.7, and 5.11 updated. Minor terminology update.
April 13, 2006	Version 2.0	Document published in Forum Nokia. Minor title change. Major content update with a number of new or updated questions mainly in Chapter 2 (concerning SID, capabilities, self-signing, application UIDs, memory, and polymorphic dll's).

---

## **1. Introduction**

Symbian OS platform security (introduced in Symbian OS v9) is a fundamental concept addressing the security and integrity of data and applications. Application developers must take this concept into account when developing and certifying Symbian C++ applications. This document answers the most frequently asked questions about platform security implemented from S60 3rd Edition onwards.

## 2. General Information about Symbian OS Platform Security

### 2.1 What is capability?

Capability is an access token that corresponds to permission to access sensitive system resources.

### 2.2 What is TCB?

TCB stands for "Trusted Computing Base." The trusted computing base consists of a number of architectural elements that cannot be subverted and that guarantee the integrity of the device. This trusted core runs with "Tcb" system capability. The components with Tcb capability have full access to file systems including reading/writing to `\sys\bin`.

### 2.3 What is SID?

SID stands for "Secure Identifier." It is a locally unique identifier that is used to determine which private directory a process can access. It is also used to identify the caller applications. The SID can be defined in the Mobile Media Platform (MMP) file as follows:

```
SECUREID <SID>
```

where <SID> should be replaced with your assigned SID. If `SECUREID` is not specified in the MMP file, the value of UID3 is also used as the SID. The usage of UID is as follows:

```
UID <UID2> <UID3>
```

where <UID3> should be replaced with your assigned SID if you don't specify `SECUREID`.

If neither is provided, both the UID3 and the SID will be 0. A Secure ID of zero means "undefined."

### 2.4 Is the SID generated by the Software Installer? If not, where do I get the SECUREID value for the MMP file?

An application's SID is defined within its .mmp file, and is used to uniquely identify the application. SID values are requested from the Symbian Signed Web site ([www.symbiansigned.com](http://www.symbiansigned.com)). The Software Installer (SWI) is responsible for ensuring that SIDs are locally unique, i.e., that no two applications have the same SID value on a particular target device.

### 2.5 What is UID?

UID stands for "Unique Identifier." It can be defined in the MMP file as follows:

```
UID <UID2> <UID3>
```

The first UID is generated automatically, depending on the `TARGETTYPE` of the file, for example, EXE or DLL. UID2 may need to be a specific value for some kinds of DLLs, especially those used by plug-in frameworks. UID3 is used as the Secure ID if `SECUREID`

is not specified in the MMP file. If it is specified, UID3 will be used to identify the applications as its original purpose.

## 2.6 What is the difference between SID and UID?

In the current implementation of platform security, UID3 is used as the SID if `SECUREID` is not specified in the MMP file. But

- The SID is used to determine which private directory a process can access. It is also used to identify the caller applications.
- The UID is used to uniquely identify an application.

## 2.7 Can the SID and UID3 of an application be different?

In the current implementation of platform security, UID3 is used as the SID if `SECUREID` is not specified in the .mmp file. However, the SID is used to determine which private directory a process can access as well as to identify the caller applications, whereas the UID is used to uniquely identify an application. It is theoretically possible that these two identifiers could have different values, but to avoid confusion Symbian recommends that a `SECUREID` not be specified in the application's MMP file; UID3 should always be specified instead.

## 2.8 How do SIDs relate to applications consisting of an EXE and a DLL? Are the SIDs of these the same?

The SID value is not relevant for DLLs because a process' SID will always be that of its EXE.

## 2.9 How is the SID of the calling EXE checked?

Calling `RProcess::SecureId()` returns the SID of a process. When doing interprocess communication (IPC), the `TSecurityPolicy` class can be used to specify a security policy consisting of both capability and SID checks.

## 2.10 What is VID?

VID stands for "Vendor Identifier," and it uniquely identifies the source of the application. The Vendor ID can be defined in the MMP file as follows:

`VENDORID <VendorId>`

- Unsigned applications will have an undefined VID (0).
- Signed applications from the same Vendor should have the same VID.
- If an application needs a VID, it must be certified; noncertified applications must use a VID of 0 (`KNullUid`), which is the value applied by default.
- The VID cannot be modified after building the application.

For example, the Nokia VID is `0x101FB657`.

## 2.11 Why is the registration RSS file required? Where should it be located? What information should be specified in it?

An application registration file defines information about an application that is required by the application launcher or system shell. This includes the application's name, UID, and properties. Other information used by the shell, for instance the icons and captions, is defined separately; the location of the icon/caption definitions is provided in the registration file. Before Symbian OS v8.1, all of this information was provided by aif files. In Symbian OS v8.1, both aif files and registration files are supported, but from Symbian OS v9 (S60 3rd Edition) onwards, only registration files are supported.

A registration file is required by every application, even if it has default properties, icons, and a caption. At minimum, it must specify the application's UID and the name of the application's executable. All registration files are located in `\private\10003a3f\import\apps`, on the same drive as the application.

## 2.12 What are the locations where files used by the application should be installed?

Three completely new protected directories have been introduced because of platform security.

### `\sys`

- All executable native code resides in `\sys\bin`.
- A flat directory — all EXE files, DLLs, etc. must have a different name.
- Tcb capability is required to modify the `\sys` contents.
- AllFiles capability is enough to read `\sys` (file browser, diagnostics tools).

### `\resource`

- Read-only resource files are shared by (potentially) all applications.
- Applications without any capabilities can read these files.
- Tcb capability is required to modify the `\resource` contents.
- The Software Installer adds and updates these files, but nothing else.
- Subdirectories can be utilized to ensure file-name uniqueness.

### `\private`

- `\private\<SID>` is the private directory of the process with the SID.
- Only the process with the same SID has full access to the directory.
- Other processes need the AllFiles capability to be able to access the directory.

These protected directories are located on either fixed or removable drives. All the other (previously used) directories are unprotected, so all applications in a device can access them.

## 2.13 What are the changes regarding application files and their locations in the system?

Type	Old, current location	New location in PlatSec dir. structure
AIF*	\system\apps\ <app>\</app>	None
Application (.app)*	\system\apps\ <app>\</app>	None
Application resource (.rsc)	\system\apps\ <app>\</app>	\resource\apps\
Application bitmap files (.mbm, .mif)	\system\apps\ <app>\</app>	\resource\apps\
EXEs	\system\programs\	\sys\bin\
FEP resources	\system\fep\	\resource\fep\
Help files	\system\help\	\resource\help\
Shared libraries	\system\libs\	\sys\bin\
Executables	\system\programs	\sys\bin\
Temp files	\temp	\private\ <sid executable="" of="" the="">\temp</sid>

\*) AIF and app files are not supported.

## 2.14 What changes are required to change an application from an .APP to an .EXE?

MyApp.MMP:

A new target and target type must be defined in the MMP file. Additionally, every line that starts with RESOURCE or SYSTEMRESOURCE must be changed to a START RESOURCE ... END block so that a TARGETPATH can be defined for the resource to be built. The TARGETPATH keyword will be neglected by the compiler when used outside START RESOURCE ... END blocks. Add the changes indicated in a bold font to the existing MyApp.MMP file:

```
#ifdef APP_TO_EXE
TARGET      myapp.exe
TARGETTYPE  exe
#else
TARGET      MyApp.app
TARGETTYPE  app
#endif
UID        0x100039CE 0x1xxxxxxxxx

VENDORID   XXXX
CAPABILITY XXXX
```

```
// Note that for APP_TO_EXE targetpath need not be defined, it
// is automatically handled by the build tools
#ifdef APP_TO_EXE
TARGETPATH  \System\Apps\MyApp
#endif
```

```
START RESOURCE ..\data\myapp.rsc
HEADER
#ifdef APP_TO_EXE
TARGETPATH  XXXX
#endif
```

```

...
other settings like LANG xx
...
END // RESOURCE

#ifdef SCALABLE_UI // New app framework used

START RESOURCE ..\data\myapp_reg.rss
// Do not change the UID below.
TARGETPATH \private\10003a3f\import\apps
END

#else // Old app framework used (no changes in the definitions below)

START RESOURCE ..\data\myapp_caption.rss
HEADER
#ifdef APP_TO_EXE
TARGETPATH XXXX
#endif

...
other settings like LANG xx
...
END // RESOURCE

AIF myapp.aif ..\aif\ myappaif.rss c8,8
..\..\bitmaps2\qgn_menu_myapp_lst.bmp
..\..\bitmaps2\qgn_menu_myapp_lst_mask_soft.bmp

#endif

```

#### MyApp.cpp:

Add new entry points for EXE type application creation and initialization in the file where `NewApplication()` is defined.

The changes are marked with a bold font.

```

#ifdef APP_TO_EXE

#include <eikstart.h>

LOCAL_C CAppApplication* NewApplication()
{
    return new CMyApplication;
}

GLDEF_C TInt E32Main()
{
    return EikStart::RunApplication(NewApplication);
}

#if defined(__WINS__) && !defined(EKA2)
GLDEF_C TInt E32Dll(TDllReason)
{
    return KErrNone;
}

EXPORT_C TInt WinsMain(TDesC* aCmdLine)
{
    return EikStart::RunApplication(NewApplication, aCmdLine);
}
#endif

#else // !APP_TO_EXE

EXPORT_C CAppApplication* NewApplication()

```

```

    {
    return new CMyApplication;
    }

#if !defined(EKA2)
GLDEF_C TInt E32Dll(TDllReason)
{
    return KErrNone;
}
#endif

#endif // APP_TO_EXE

```

## 2.15 Where did 10003a3f come from in the MMP file for the target resource path?

This is the SID of the application loader. All application registration files should be located in this directory, and located on the same drive as the application.

Note: The project definition (MMP) file will specify the location as `\private\10003a3f\apps`, however the package (PKG) file must specify `\private\10003a3f\import\apps` so that the Software Installer can install the files successfully.

## 2.16 Can I use an SID or a VID instead of capability check?

SID and VID are not capabilities. But you can offer specific services for clients with a specific SID and/or VID.

- The SID is used to determine which private directory a process can access, and it can also be used to identify clients. It is a value that cannot be faked.
- The VID can be used to identify the source of the application.

## 2.17 Does the SID of a DLL have any run-time significance?

The SID of a DLL has no run-time significance.

## 2.18 How and when are capabilities put in the DLL or EXE?

This is done during the build process. Indicate in the project (.mmp) file what capability you want to give to your binary. For example, your application will have `ReadUserData` and `WriteUserData` capabilities by defining the following line in your project file:

```
CAPABILITY ReadUserData WriteUserData
```

## 2.19 Is it possible to change capabilities during run time?

No, it is not possible to change them during run time.

## 2.20 What certifications do the capabilities require?

The capabilities that require Symbian Signed are listed below (for capabilities marked with an asterisk Symbian Signed certification is preferred but not mandatory, as the user can grant the capability at the installation time):

- LocalServices \*
- ReadUserData \*
- WriteUserData \*
- UserEnvironment \*
- NetworkServices \*
- Location

For details, visit [www.symbiansigned.com](http://www.symbiansigned.com).

The capabilities that require Symbian Signed and a Declarative Statement are listed below:

- PowerMgmt
- ProtServ
- SwEvent
- SurroundingsDD
- ReadDeviceData
- WriteDeviceData
- TrustedUI

The capabilities granted after approval of the Symbian Capability Request form (available through [www.symbiansigned.com](http://www.symbiansigned.com)) are listed below:

- Multimedia DD
- NetworkControl
- CommDD
- DiskAdmin
- AllFiles

The capabilities granted only by the manufacturer are listed below:

- DRM
- TCB

Access to these capabilities can be requested through the capability request form at [www.forum.nokia.com/testing](http://www.forum.nokia.com/testing).

## 2.21 What certificate is needed in the development phase?

If your S60 C++ application 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 (see Section 2.22, “What does .sis file signing mean?”). The private key can be created with a command line tool called **makekeys**, and the application can be signed with, for example, **signsis**.

If the application requires access to any other capabilities, at minimum you'll need to get a developer certificate from [www.symbiansigned.com](http://www.symbiansigned.com) in order to run the application on the device.

If ReadDeviceData, WriteDeviceData, and TrustedUI capabilities are used, you also need the ACS Publisher ID provided by Verisign. If DRM, NetworkControl, MultimediaDD, TCB, AllFiles, CommDD, and DiskAdmin capabilities are used, you need support from the manufacturer in addition to the developer certificate and ACS Publisher ID.

## 2.22 What does .sis file signing mean?

The S60 platform includes mandatory .sis file signing. The .sis package must, at minimum, be signed with a self-signed certificate before deployment. See Introduction to S60 3rd Edition | How to Sign .sis Files in the S60 3rd Edition SDK for more information about self-signing. 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 (0x80000000 and up). For more information on application UIDs, refer to [www.symbiansigned.com](http://www.symbiansigned.com).

## 2.23 Does the behavior of self-signed applications differ during execution?

No. The method used to sign an application SIS file is only relevant at installation. The significance of self-signing an application is that it is flagged to the user as an untrusted application at installation time, and thus cannot be granted capabilities beyond the user capability set. Once an application has been successfully installed, its capabilities, SID, and VID (which are stored in the application binary) determine any restrictions to the application's behavior.

## 2.24 How does the UID range correspond to Symbian Signed and Developer Certificates? What range should be used during development time and for the final release?

It is important to remember the two UID ranges used in the context of Symbian OS v9.1 onwards: the protected range and the unprotected range.

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. Specifically, the 0 range (0x00000000 to 0x0FFFFFFF) has been reserved for development purposes, and 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. Specifically, the E range (0xE0000000 to 0xEFFFFFFF) has been reserved for development purposes, and new UID allocations are made from the A range (0xA0000000 to 0xAFFFFFFF). Before being released to end users, a self-signed application must have a UID allocated from the A range. For more information on the application UIDs, go to [www.symbiansigned.com](http://www.symbiansigned.com).

## 2.25 Do I need to re-request legacy UIDs from Symbian Signed before I can use them?

Continue to use your existing allocations for self-signed application usage on Symbian OS v9 (i.e., those applications not requiring access to sensitive APIs). To do this, simply replace the first hex digit (a 1) with F, and leave the remaining digits unaltered. This maps your UID into the Legacy UID compatibility range, where it will not conflict with any other allocations. For example, you have a UID allocation 0x100F55BE that you can transpose to 0xF00F55BE for use in an unsigned Symbian OS v9 application.

If your application is to be Symbian Signed, you need to reapply for your UID at [www.symbiansigned.com](http://www.symbiansigned.com).

## 2.26 How do I obtain a Symbian Signed Developer Certificate?

Developer Certificates (DevCerts) are used only in the context of Symbian OS v9 target devices. These certificates permit you to debug and test applications on target devices before submitting them for Symbian Signing, and limit the target devices and capabilities that the application can access. It is important to note that DevCerts are linked to the developer not the application, and therefore may be used to sign multiple applications.

To request a DevCert, you must be registered with the Symbian Signed Web site ([www.symbiansigned.com](http://www.symbiansigned.com)) and have downloaded the DevCertRequest.exe application (available from [www.symbiansigned.com/app/page/devcertgeneral](http://www.symbiansigned.com/app/page/devcertgeneral)). This application is used to produce a certificate request file (.csr), which can then be submitted via the Web site. Specify the IMEI numbers of the target device(s) you want to test your applications on and the capabilities you require. After the request has been processed, a DevCert will be made available for download from your account.

Depending on the number of target devices you wish to test and the capabilities you require, an ACS Publisher ID may be needed to create the certificate request. See [www.symbiansigned.com/Developer\\_Certificate\\_Request\\_Process\\_v1.1.pdf](http://www.symbiansigned.com/Developer_Certificate_Request_Process_v1.1.pdf) for more information.

## 2.27 Do I have to certify/sign every update of the application?

Updates of an application must be certified in a similar way as the original package. If your original application was signed by a trusted certificate, the updated version must also be signed by a trusted certificate.

If your original application was self-signed, the updated package must be at least self-signed. Note that if the upgraded version of the originally untrusted (self-signed) application is signed by a trusted certificate, the upgraded application is considered a different application because you cannot use the same SID anymore (there are protected and unprotected ranges for SIDs for trusted and untrusted applications, respectively).

## 2.28 Who is permitted access to the Symbian Signed Catalogue?

Catalogue access is limited to major distributors of Symbian software. Please see <http://www.symbiansigned.com/app/page/faq/catalogue> for more information.

## 2.29 What is “passive content”?

In the context of Symbian Signed, passive content is classed as a SIS file that does not contain any executable components. The Symbian Signed test criteria define passive content as “pure data files and/or content that does not reconfigure or cause the

application to change its normal behavior.” Please see <http://www.symbiansigned.com/app/page/faq/content> for more information.

### 2.30 Why are DLL capabilities needed?

DLL capabilities should not be regarded as “what a DLL is capable of doing,” but rather what environment it is trusted to work in. A DLL must have equal or greater capabilities than the loading process. Once loaded, a DLL runs at the capability level of the loading process. Therefore a DLL must have all capabilities required by all its client executables, even if the code within the DLL itself does not require some of these capabilities.

### 2.31 Does loading a DLL with a different set of capabilities change the process capabilities?

Loading a DLL can never change the capabilities of a process; all DLL code runs at the capability level of the loading process.

### 2.32 Can a process load a DLL with fewer capabilities than itself?

No, it cannot. There are two fundamental rules that govern the way the loader must work:

#### **Rule 1: The capabilities of a process never change.**

There is no way of adding capabilities to a process, nor of removing capabilities from a process. All DLL code runs at the capability level of the process that loads it.

#### **Rule 2: A process cannot load a DLL with less capability than itself.**

The need for this constraint follows from the fact that all DLL code runs at the capability level of the loading process and therefore must be at least as trusted as the loading process. DLL capabilities only reflect the level of trust that the loading process can place in them; they do not actually authorize anything.

The following example cases demonstrate the capability relationship between DLLs and a process.

#### **Statically linked DLLs**

The program P.EXE is statically linked to the library L1.DLL.

The library L1.DLL is statically linked to the library L0.DLL.

Case 1:

P.EXE holds Cap1 & Cap2

L1.DLL holds Cap1 & Cap2 & Cap3

L0.DLL holds Cap1 & Cap2

The load fails because L1.DLL cannot load L0.DLL. Since L0.DLL does not have a capability set greater than or equal to L1.DLL, Rule 1 applies.

Case 2:

P.EXE holds Cap1 & Cap2

L1.DLL holds Cap1 & Cap2 & Cap3

L0.DLL holds Cap1 & Cap2 & Cap3 & Cap4

The load succeeds and the new process is assigned Cap1 & Cap2. The capability of the new process is determined by applying Rule 2; L1.DLL cannot acquire the Cap4 capability held by L0.DLL, and P.EXE cannot acquire the Cap3 capability held by L1.DLL.

### **Dynamically loaded DLLs**

The program P.EXE dynamically loads the library L1.DLL.

The library L1.DLL then dynamically loads the library L0.DLL.

Case 1:

P.EXE holds Cap1 & Cap2

L1.DLL holds Cap1 & Cap2 & Cap3

L0.DLL holds Cap1 & Cap2

The load succeeds because P.EXE can load L1.DLL and L0.DLL. Rule 1 does apply, but here the loading executable is the process P.EXE, not the library L1.DLL: the Rlibrary::Load request that the loader processes is sent by the process P.EXE. Here, the fact that the call is within L1.DLL is irrelevant. Rule 2 applies as before and P.EXE does not acquire Cap3 by loading L1.DLL.

Case 2:

P.EXE holds Cap1 & Cap2

L1.DLL holds Cap1&Cap2&Cap3

L0.DLL holds Cap1&Cap2&Cap4

The load succeeds because P.EXE can load L1.DLL and L0.DLL. Once again, Rule 1 does apply with P.EXE as the loading executable rather than L1.DLL, while Rule 2 ensures P.EXE acquires neither Cap3 nor Cap4.

### **2.33 How are the capabilities assigned when a process is spawned by another process?**

When a process is spawned by another process, it runs independently. It doesn't have the capabilities of the creator process. The capabilities of the spawned process are assigned as they are defined during build time (in the .mmp file).

### **2.34 How does the EPOCHEAPSIZE work?**

`epocheapsize minimum maximum`

Use the `epocheapsize` statement to specify the minimum and maximum sizes of the initial heap for a process. The default sizes are 4 KB minimum and 1 MB maximum.

The minimum size specifies the RAM that is initially mapped for the heap's use. The process can then obtain more heap memory on demand until the maximum value is reached.

The sizes can be specified in decimal or hexadecimal format. Memory is allocated in pages, so the minimum and maximum values are rounded up to a multiple of the page size (4 K).

### 2.35 How does the EPOCSTACKSIZE work?

```
epocstacksize stacksize
```

Use the `epocstacksize` statement to specify a stack size for your executable other than the default 8 KB.

The size of the stack, in bytes, can be specified in decimal or hexadecimal format. Use of this statement will have no effect under the WINSCW/WINS platforms.

EPOCSTACKSIZE allows you to specify a stack size for your executable file, therefore enabling you to override the default (8KB). EPOCSTACKSIZE should be included in the MMP file, e.g., EPOCSTACKSIZE 0x5000. Note: The overridden value can be specified in decimal or hexadecimal format.

## 3. Data Caging

### 3.1 What is the idea behind data caging?

Data partitioning involves separating code from data in the file system so that a simple trusted path is created on the platform. The central idea is that by “caging” non-TCB processes into their own part of the file system, system files become hidden to them.

The file system has the following structure:

- `/sys/`  
The `/sys` folder is the restricted system area and is inaccessible to non-TCB programs (TCB capability is required to modify the `/sys` contents but AllFiles capability is enough to read `/sys` (for example, by the file browser application and diagnostics tools)). The `/sys` folder contains a subdirectory `/sys/bin/` that holds all executables (EXEs, DLLs, etc.). Because it is a flat directory, all EXE files, DLLs, etc., must have a different name. The OS will refuse to run code placed in any other location.
- `/private/`  
Non-TCB programs have their own restricted view on the file system consisting of the directory subtree `/private/<SID>/`, where SID is a unique security identifier (SID), taken from the program's UID. Applications, for example, would use their subtree to hold `.ini`, `.mbm`, `.rsc`, and data files. Other processes need AllFiles capability to be able to access the directory.
- `/resource/`  
This directory is public, but is read-only for non-TCB processes. The purpose is to allow read-only files to be publicly shared without compromising their integrity. Applications without any capabilities can read these files.  
  
Tcb capability is required to modify the `\resource` contents. The software installer adds and updates these files, but nothing else. Subdirectories can be utilized to ensure file-name uniqueness.

These protected directories are located on either fixed or removable drives. All the other (previously used) directories are unprotected, so it is not safe to leave any sensitive data in them.

### 3.2 What is the idea behind using the import directory?

Some servers can have their functionality extended by installing associated plug-ins; they discover the existence of any plug-ins by scanning a directory where the plug-ins' resource files are expected to be found.

Since these resource files are of no interest to any executables apart from the relevant server, it is natural to put them in their private directory. But how does Software Install decide whether it is safe for an apparently unrelated SIS package to install a file in the private directory of another process?

The solution is to be found in the concept of the import directory. Software Install will only allow the package to write files into a subdirectory of an unrelated process if it is named “import” (that is, `/private/<process_sid>/import/`) and if such a subdirectory already exists.

An example of this is the directory where third-party application registration files are stored: `\private\10003a3f\import\apps`.

### 3.3 Does the AllFiles capability allow reading from/writing to any directory under `\sys`, `\private`, and `\resource`?

The AllFiles capability allows:

- Read-write access to any directory under `\private`
- Read-only access to any directory under `\sys`.

### 3.4 Can any program read from/write to other directories not under `\sys`, `\private`, and `\resource`?

Yes. Directories not under `/sys`, `/private`, or `/resource` are public and can be read from and written to by any program.

### 3.5 How do I change SYSTEMRESOURCE in the .mmp file?

The SYSTEMRESOURCE keyword automatically places the resource files to the `System\Data` directory, regardless of whether or not TARGETPATH was defined in the MMP file.

If your current MMP file contains the following:

```
SYSTEMRESOURCE myResource.rss
```

after the change it should look like:

```
START RESOURCE myResource.rss
HEADER <- if you need an .rsg file to be generated
TARGETPATH \resource
END
```

### 3.6 Is the import folder under `\private\<SID-of-process>` writable or accessible by everyone?

No, it can only be accessed by the process that owns it or by a process with AllFiles capability.

### 3.7 Can DLLs have their own private directories?

No, they cannot. A private directory is determined according to the loading process' SID.

### 3.8 How do polymorphic DLLs work?

It is no longer possible to look in the `\sys\bin` folder to find the DLLs without having the required access capabilities. Therefore searching for polymorphic DLLs should be done through `TFindLibrary`, which can be used to find DLLs whose full names match a specified pattern. Once the required polymorphic DLL has been found, the `Handle()` method can be used to obtain the find-handle number associated with the kernel object. The polymorphic DLL handle allows a program to load and close a particular polymorphic DLL. It also allows the caller to obtain pointers to functions exported by the

DLL. The system can check that a polymorphic DLL is of the correct type by checking the type UID value.

---

## 4. Publish & Subscribe

### 4.1 What is Publish & Subscribe?

Publish & Subscribe is a new API provided by the real-time kernel (EKA2). It allows publisher processes to define and update a set of properties; other processes, called subscribers, can listen for changes to a property, and get property values. The process that defines a property can specify access rights for both reading and writing. Rights can be defined in terms of either requiring a particular security capability, by a process SID, or by a process VID. Publish & Subscribe replaces System Agent and the usage of temporary Shared Data keys.

### 4.2 What is the difference between Publish & Subscribe and Central Repository?

The main difference is that Publish & Subscribe is meant for nonpersistent settings, whereas Central Repository is used for persistent settings.

### 4.3 What is the best way to efficiently share a large amount of data between two processes in a secure way?

Sharing a large amount of data between two processes can, of course, be done with normal IPC. However, this method is slow, and the following way is more efficient:

- Create an unnamed global chunk by using  

```
RChunk::CreateGlobal(KNullDesC, ...).
```

Creating an anonymous chunk will prevent other processes from finding it using `TFindChunk`.
- Put some data into it.
- Pass the chunk handle by using `client-server IPC` or `RProcess::SetParameter()`.
- Open the chunk in the other process.

Note that no capability is required to use global chunks. If an anonymous chunk is created, then the handle must be passed to the application that will access the chunk.

---

## 5. Configuring Platform Security Settings

### 5.1 How do I turn on/off platform security diagnostics?

When these checks fail, a diagnostic message may be emitted. This feature is enabled as follows:

```
PlatSecDiagnostics On
```

To disable:

```
PlatSecDiagnostics Off
```

If the emulator is in use, you can enable/disable `PlatSecDiagnostics` in `\epoc32\data\epoc.ini`. If `PlatSecDiagnostics` is enabled, diagnostic messages will be sent to `epocwind.out`.

### 5.2 How do I turn on/off platform security enforcement?

When a capability or other platform security policy check fails, and after any diagnostic message has been emitted (if enabled), the final action is dependent on whether platform security enforcement is enabled.

If enforcement is not enabled, the system continues as though the original platform security check in fact passed. If enforcement is enabled, the appropriate action for a failed platform security check happens.

This feature is disabled as follows:

```
PlatSecEnforcement Off
```

To enable enforcement:

```
PlatSecEnforcement On
```

If the emulator is in use, you can enable/disable `PlatSecEnforcement` in `\epoc32\data\epoc.ini`.

### 5.3 How do I disable a specified set of capabilities and what does it mean to disable them?

The keyword `PlatSecDisabledCaps` can be used to disable a specified set of capabilities, as shown in this example:

```
PlatSecDisableCaps ReadDeviceData+ReadUserData
```

Disabling a specific set of capabilities means that these capabilities will always succeed and will not cause diagnostic messages. You can use `PlatSecDisabledCaps` in `\epoc32\data\epoc.ini` if the emulator is in use.

### 5.4 What is the capability if the CAPABILITY keyword doesn't exist in the MMP file?

If the `CAPABILITY` keyword is not present in the MMP file, the default of `CAPABILITY NONE` is used.

### **5.5 What does "CAPABILITY ALL -TCB" mean?**

It means that all capabilities excluding TCB are assigned. Capability names prefixed by "-" indicate which capabilities to exclude.

### **5.6 How do I exclude multiple capabilities?**

The following example shows how to exclude multiple capabilities:

```
CAPABILITY ALL -TCB -AllFiles -DiskAdmin
```

## 6. SW Installation

### 6.1 Where can I find information about signing sis files?

See *Introduction to S60 3rd Edition | How to Sign .sis Files* in the S60 3rd Edition SDK for more information about self-signing.

### 6.2 What are the new directives used in a .pkg file?

There are three new directives and they are specified with the .pkg characters %, :, and =, which indicate the following:

- % specifies localized vendor names (corresponding to language).
- : specifies a nonlocalized (global) vendor name. This is used, in combination with signing, to prevent the unauthorized upgrade of a package by someone other than the rightful vendor.
- = specifies logo files (corresponding to language).

Below is a sample .pkg file with the new directives:

```
; Specify the languages - as previously supported
&EN,FR

; List of localised vendor names - one per language. At least one
; must be provided (English [EN]).
; List must correspond to list of languages specified elsewhere in
; the .pkg
%{"Vendor-EN", "Vendor-FR"}

; The non-localised, globally unique vendor name (mandatory)
:"Forum Nokia"

; Optional Logofile - display logfile of given type mimetype; if
; targetname is supplied, install file to that location
; <logfile>,<mimetype>[,<targetname>]
="file\mylogo.jpg","image/jpeg","\public\logos\mylogo.jpg"
```

### 6.3 Does the S60 3rd Edition SW Installer support old SIS files?

No, it doesn't. The SIS file format has changed in Symbian OS v9, and the S60 SW Installer supports only the new format. Note that the file extension still remains as **.sis**.

### 6.4 Where are ECOM registration resource files installed?

ECom registration resource files for ECom plug-ins should be installed to the `\resource\plugins` directory.

### 6.5 Where are application registration resource files installed?

Application registration resource files should be installed to the `\private\10003a3f\import\apps` directory.

## 6.6 What should I do if I receive a "Certificate check failed. Contact application supplier" message?

Make sure that your .sis file is signed. SIS file signing is mandatory in S60 3rd Edition, and applications must, at minimum, be signed with a self-signed certificate. This means that all applications that are installed via the SW Installer, including test applications and different tools, must be signed. See *Introduction to S60 3rd Edition | How to Sign .sis Files* in the S60 3rd Edition SDK for more information.

This error message may also indicate that certificate validation failed.

## 6.7 I get a "Cannot grant capabilities" message during installation. What does it mean?

The SW Installer performs signature validation and validates the certificate that the .sis file was signed with. If the signature is not valid (for example, it has expired) and your application requires capabilities other than LocalServices, NetworkServices, ReadUserData, WriteUserData, and UserEnvironment, then the SW installer shows "Cannot grant capabilities" and aborts the installation.

Also, make sure that the date is set correctly on the device.

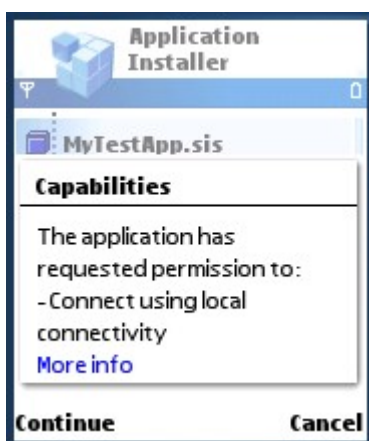
## 6.8 Which user capabilities can the user grant for an application not certified by a trusted certificate?

The following user capabilities can be granted with the user's confirmation when the signature is not valid:

```
LocalServices NetworkServices ReadUserData WriteUserData
UserEnvironment
```

This applies also in the case of self-signed certificates.

For example, the following dialog will be shown to the user to grant the LocalServices capability:



## 6.9 I get an "Unable to verify supplier. Continue anyway?" message during installation. What does it mean?

It means that the certificate that was used to sign the .sis cannot be verified. Make sure to

- Use the correct certificate and key to sign your .sis file
- Set the correct date on the device.

## 6.10 I get a "Cannot install. Access denied" message during installation. What is the reason?

Possible reasons for receiving this message are:

- Installing files into other processes' private directories. If the private directory doesn't belong to an application in the package, the SW Installer will only allow installing files into or under the `\private\<SID of other process>\import` folder. For example, if you are trying to install your application's registration resource file:

Correct:

```
"testapp_reg.rsc"-
"!\private\10003a3f\import\apps\testapp_reg.rsc"
```

Incorrect:

```
"testapp_reg.rsc"- "!\private\10003a3f\apps\testapp_reg.rsc"
```

- `\sys\bin` is the only directory under `\sys` that the SW Installer installs files into. Installing files into any other location than `\sys\bin` is not allowed. For example, the SW Installer rejects installing files into the `\sys\hash` and `\sys\install` directories.

## 6.11 How should I use the manufacturer identifier in a condition block in a .pkg file?

The manufacturer identifier can be used in a condition block in a .pkg file as follows:

```
IF manufacturer = 2 ; (2 is Nokia)
"MyTestApp.exe"- "!\sys\bin\MyTestApp.exe"
ELSE
"othermanufacturer.txt"- "", FILETEXT, TEXTEXIT
ENDIF
```

Manufacturer identifiers are:

```
Ericsson=0,
Motorola=1,
Nokia=2,
Panasonic=3,
Psion=4,
Intel=5,
Cogent=6,
Cirrus=7,
Linkup=8,
Texas Instruments=9
```

## **6.12 Can I launch my application during the install/uninstall procedure?**

Your application must be certified in order to use FILERUN and launch it during install/uninstall. Self-signed applications cannot be launched.

---

## 7. Testing

### 7.1 How do I change the capabilities of test applications for testing purposes?

There are two ways to change the capabilities:

- You can change the capabilities in the MMP file and recompile your test application, or
- You can set the capabilities of your test application by using `SetCap.exe`. This tool creates a copy of an executable file and it gives the specified capabilities. The tool that can be used with the emulator runs under Symbian OS and it is not a native PC utility.

See the SetCap documentation in the SDK Help for further information.

---

## 8. 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](#).