Introduction to Self Signed

Self Signed applications are applications that have been digitally signed using a certificate that was created locally by the developer - the information in the certificate has not been independently verified by a certificate authority.

Overview

Self signing has the benefit that its free, fast, and self signed applications are not locked to any particular device:

- You can generate your own free certificate rather than pay for one from a certificate authority and you don't need to pay for testing.
- There's no downtime while waiting for Symbian Signed to sign your application.

However self-signed applications are not *trusted* by Platform security because there is no way of verifying the identity of the signer. On installation the user must acknowledge a warning prompt that the application is untrusted (the warning dialog does not appear when the application is Symbian Signed) and approve any user capabilities requested by the application.

Provided you don't mind the installation warning and you're only using the user capabilities, self signing may be your best distribution option.

Capabilities that can be granted to self signed applications

Self signed applications can be granted the user capabilities on installation:

- Read/WriteUserData: access confidential user data (calendar, contacts and so on)
- NetworkServices: networking, dialing, messaging
- LocalServices: local connectivity – Bluetooth, infrared, messaging
- UserEnvironment: capturing information about the user and his environment (the camera, for example)
- Location: phone location (GPS)

Its possible to self sign an application that requires stronger capabilities, but installation will fail when you attempt to install the application.


Note: If you need more capabilities then you'll need to Symbian Sign the application or use a Developer Certificate.

UID Restrictions

Self signed applications may not use a UID3/SID values in the protected range.

During development you may use a random UID from the `0xE` range. On release you should use a (free) UID in the `0xA` range that has been allocated by Symbian Signed in order to avoid application UID classes.

How to self sign your application

Self signing is largely automated through both the Carbide.c++ and Qt Creator IDEs, and with Qt on the command line. For each of these, the IDE will automatically self sign an application with little or no configuration.

Its also possible to create your own certificate and keys and self sign an application on the command line using Symbian tools directly.

Self signing with Qt Creator

Qt Creator includes building SIS files as part of the Run settings in Project mode (see the Project button on the sidebar). Qt Creator is default pre-configured to build and self-sign sis files for Symbian device targets.

Self signing with Qt Command line

Qt applications can be built and signed on the command line. By default `make SIS` will build a self signed application. `make installer_sis` can then be used to create a self-signed "smart installer" wrapper around the first package.

Both make options take additional parameters that allow you to specify your own certificate (self signed, developer certificate or publisher ID).

Self signing with Carbide.c++

Application SIS file building and signing is handled in the project "Build Configurations" (note that projects are not automatically set up for building SIS files):

1. Open your project in Carbide.c++
2. In project menu do: Project | Properties | Carbide.c++ | Build Configurations | SIS Builder
3. Select a build configuration for a device (usually GCCE debug or release)
4. Select **Add** to add a new SIS build configuration.
5. Select the package file to use.

By default the builder is already configured to self-sign the SIS file (you can change the certificate if you'd prefer to use a Developer Certificate).

The SIS file is then built and self signed when you build your project in a device build configuration.

Self signing using standard Symbian tools

The process to create a new key and certificate is covered in the product reference: Creating a Private Key and Self Signed Certificate

You can then sign the application using Signsis. The normal usage is something like this:

```
signsis ApplicationName.sis Signed_ApplicationName.sisx certificate.cer key.key password
```

What if my self-signed application won't install?

- Some devices may not allow installing self-signed applications by default. This can be enabled by changing the phone settings - see: How to enable installation of self-signed applications
- You may get a certificate warning for a number of reasons. For example if your application requests more capabilities than can be granted by the user, or if your certificate expires, or your PC clock is ahead of your device so that the certificate used appears to be "in the future. See the topic Troubleshooting Installation Errors in order to solve these sorts of issues.