Try & Buy App Template

This article explains how to implement Try and Buy functionality by using Nokia In-application Purchase API on Qt

Introduction

The problem

Nokia In-application Purchase API for Symbian Qt does not allow to create complete secure Try and Buy solution because there is no secure storage in IAP client. You cannot collect application use statistic locally on device ensuring there is no way for hacker to extract it. Symbian platform security model allows application data caging but IAP client does not employ that feature. For ordinary application only public folders are available under "c:/data" device root folder. We don't take into account memory card storage for quite obvious reason.

Qt application data persistence

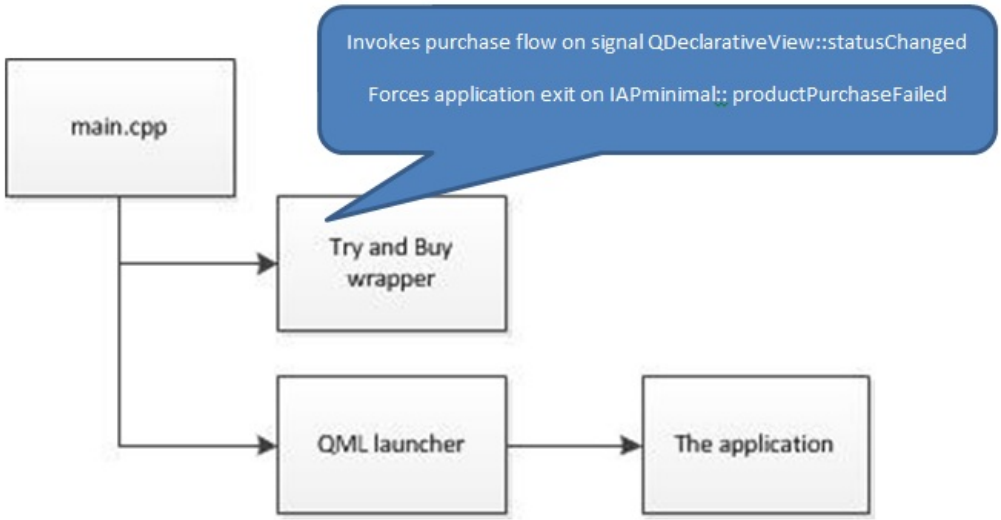
Along others application persistence data mechanisms Qt provides QSettings. QSettings is not data secure. When application settings are created you can explore it with a file browser in "c:/data/.config/<app_UID>" directory on device. Good news is that QSettings creates folder in application run-time that is preserved while application uninstall. That is just a side-effect but that behavior is very helpful to keep application use statistics independently of the application installing and uninstalling. All we need is to implement hacker-proof logic in the application. It will not be 100% unbreakable since public folder is used.

Way of protection application use data statistics

```
* Principles
```

1. Try and Buy implementation logic prevents hacker from application complete unlocking. Hacker can only convert the application to the initial state -- for example "initial 3 tries are available". So after reverting apparently locked app to the initial state, hacker will have to hack it again each time initial 3 tries expiry. That is not for ordinary user -- thus we will build the first line of our application data protection. Please note on production device there is no file browser by default that can delete public directories, only public files can be deleted, and that fact also facilitates for us data protection implementation. Please see "void checkTNB(IAPminimal& iap)" function implementation for further details. The code is available in here
2. Unlocked application state shall not use initial protection logic and anything based on public folders. We are going to use Nokia DRM IAP product protected model. It allows us to use product license checking with reading of DRM encrypted file and automatic license restoration -- that is needed for example after the application re-installing.

```
* Implementation details
```



1. Try and Buy wrapper isolates In-application Purchasing functionality from the application source code and allows you to re-use it on top of you existed application without much of modification.
2. application use statistic protection logic is resides in one function invoked in main.cpp
3. IAP needs a host widget to display pop-up dialogs thus we allow application QML loader to run even in case when the app is apparently locked and user has to buy the app. If purchase flow is needed to show, it will be shown on top of the application screen.
4. IAP cannot be ready immediately after creation and even more , apparently it is impossible to start purchasing flow right after creation, we need to fetch product metadata first, otherwise IAP client will report error "Purchase service is not available". I think that is IAP client implementation flaw. But that "product data fetching -- > purchasing" scenario is acceptable workaround that is encapsulated into Try and Buy wrapper.

Adding Try and Buy functionality step-by-step

1. Download source code from Try & Buy App Template project. The project consists of two parts : original application and instrumented application. You can use a file comparison tool to see what exactly you need to adopt in your application project
2. Instrument main.cpp module that will check for the application license and trigger IAP purchase flow if it is needed
3. Add IAP purchase item to the project (IAP_VARIANTID.txt, recourse_XXXXX, etc...)
4. Add extra program modules that supports Try and Buy functionality: drmfile.*; symbian_drm/drmfile_p.*; iapminimal.*

Summary