Qlap - a simple Qt interface for In-App Purchasing

This article explains how to use `QIap`, a class wrapper which provides a simpler interface to the standard Qt In App Purchasing API.

Introduction

In-App Purchasing is a technology that enables end users to purchase and pay for additional content (such as a new level in a game) or services (such as a subscription) from within your application. The standard In App Purchasing API provides a lot of functionality but can be complicated to use. `QIap` introduces a very high level wrapper that makes this technology easier to use.

`QIap` is hosted on Nokia Projects: http://projects.developer.nokia.com/qiap . You can get the project source as a zip (qiap.zip) and there is a quick start here: InAppPurchase_QuickStartForQt.pdf.

Get started with Qlap

Setup

First you have to create your own resource product in OVI Publisher panel. Please refer to this document (page 7).

1. Download Qlap files and extract inside your project directory.
2. Open your .pro file and add the following lines:

```
symbian {
    # Enables In-App Purchase API
    DEFINES += IN_APP_PURCHASE
    #DEFINES += IN_APP_PURCHASE_DEBUG

    # Enables test mode for IAP
    #DEFINES += IA_PURCHASE_TEST_MODE

    contains(DEFINES, IN_APP_PURCHASE) {
        include(./qiap/in-app-purchase.pri)
    }
}
```

3. Create a directory called iap and put inside the following files and directories:
   - <drive>:\myproject\data\ — resource files for paid apps
   - <drive>:\myproject\data\resourceid_XXXXXX\ — The XXXXXX is the six-digit in-app ID provided by Nokia Publish when you identified the in-app purchase items
   - IAP_VARIANTID.TXT — A file containing 000000 (six zeroes), which is required to use the In-App Purchase API. When you submit your app to Nokia Publish, Nokia Publish populates IAP_VARIANTID.TXT with data used to get DRM access.
   - TEST_MODE.TXT — A file used to simulate purchases and bypass real payments. Remove this file for real purchases with the live Nokia Store server.

For details about using this file during testing, see here

QML

Inside main.cpp

```
#ifdef IN_APP_PURCHASE
#include "qiap/qiap.h"
#include <QtDeclarative>
#endif

int main(int argc, char *argv[])
{
    QApplication app(argc, argv);

#ifdef IN_APP_PURCHASE
    qmlRegisterType<QIap>("IAP", 1, 0, "QIap");
#endif
    .....

    return app.exec();
}
```

QML file

```
import IAP 1.0
import Qt 4.7
import QtQuick 1.0

Rectangle {
    id:main
    anchors.fill: parent;

        Button {
            id: photo
```


(C) Copyright Nokia 2013. All rights reserved.

```
            text: "Buy Photo"

            MouseArea {
                anchors.fill: parent
                onClicked: {
                    iap_manager.purchaseProductByName("SBK Photo")
                }
            }
        }

        Text {
            id:text_purchased
        }

    }

    QIap {
        id:iap_manager
        onPurchaseCompleted: {
           console.log("onPurchaseCompleted")
           console.log(">"+status)
           console.log(">"+productID)

            if( status===="OK") {
               if( productID === "813279" ) {
                   text_purchased.text = iap_manager.getDRMFileContent(productID,"video.csv");
               } else
               if( productID === "821073" ) {
                   text_purchased.text = iap_manager.getDRMFileContent(productID,"photo.txt");
               }
           }
       }
    }
}
```

QT

Inside mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QtGui/QMainWindow>

#ifdef IN_APP_PURCHASE
#include "qiap.h"
#endif

namespace Ui {
    class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT
public:
    explicit MainWindow(QWidget *parent = 0);
    virtual ~MainWindow();

private slots:

#ifdef IN_APP_PURCHASE
private:
    QIap* iap_manager;

private slots:
    void getProductsCompleted();
    void restoreProductsCompleted();
    void purchaseFlowFinished(int);
    void purchaseCompleted(QString status, QString productID);
    void itemRestored(QString);
#endif

private:
    Ui::MainWindow *ui;
};

#endif // MAINWINDOW_H
```

mainwindow.cpp

```
MainWindow::MainWindow(QWidget *parent) : QMainWindow(parent), ui(new Ui::MainWindow)
{
    ui->setupUi(this);

#ifdef IN_APP_PURCHASE
    iap_manager = new QIap(this);
    connect(iap_manager, SIGNAL(getProductsCompleted()), this, SLOT(getProductsCompleted()));
    connect(iap_manager, SIGNAL(restoreProductsCompleted()), this, SLOT(restoreProductsCompleted()));
    connect(iap_manager, SIGNAL(itemRestored(QString)), this, SLOT(itemRestored(QString)));
    connect(iap_manager,SIGNAL(purchaseCompleted(QString, QString)),this,SLOT(purchaseCompleted(QString, QString)));
    connect(iap_manager,SIGNAL(purchaseFlowFinished(int)),this,SLOT(purchaseFlowFinished(int)));
#endif

}
```

```
void MainWindow::purchaseProductByID(QString productId)
{
    iap_manager->purchaseProductByID(productId, IAPClient::ForcedAutomaticRestoration);
}
```

```
void MainWindow::purchaseProductByName(QString productName)
{
    if( iap_manager->purchaseProductByName(productName, IAPClient::ForcedAutomaticRestoration) == -1 )
```


(C) Copyright Nokia 2013. All rights reserved.

```
        QMessageBox::critical(this,"IAP Error","In App Purchase Error.\nMay be product name isn't correct.");
}
```

purchaseProductByID is the method to purchase a product by ID.
purchaseProductByName is the method to purchase a product by Name you gave into ovi publisher panel.
It's better to use IAPClient::ForcedAutomaticRestoration parameter.

```
void MainWindow::purchaseCompleted(QString status, QString productID){
    qDebug() << "purchaseCompleted: status :" << status;
    qDebug() << "purchaseCompleted: item ID:" << productID;

    if( status == "OK"){
        QString fileName = "/resourceid_"+productID+"/";

        switch( productID.toInt() ){
            case 813279 : fileName.append("video.csv"); break;
            case 821073 : fileName.append("photo.txt"); break;
            default : break;
        }

       // DRMFile file;
        QString filepath = QApplication::applicationDirPath();
        filepath.append("/drm/data");
        filepath.append(fileName);

        uchar* buffer = NULL;
        int len = iap_manager->getDRMFileContent(filepath, buffer);
        if(len>0) {
            QString content = QString::fromAscii((char*)buffer,len);
            qDebug() << content;
            QMessageBox::information(this,"IAP Example", content);
        } else {
            qDebug() << "Error reading DRM file";
            QMessageBox::critical(this,"IAP Example", "Error reading DRM file");
        }
        delete buffer;

    } else
        QMessageBox::critical(this,"IAP Error", status);
}
```

purchaseCompleted occurs when purchase process is completed retrieving information about transaction status and managed ID product. Status is Cancel if user aborts purchasing.

```
void MainWindow::purchaseFlowFinished(int value){
    qDebug() << "purchaseFlowFinished:"<<QString::number(value);
    qDebug() << "purchaseFlowFinished item ID:"<<iap_manager->currentPruductID();
}
```

purchaseFlowFinished occurs when user closes IAP window and control returns to main GUI.

Restoration Process

Restoration is the process to recover products already purchased in the past

You can use QIap::restoreProducts() to restore all purchased products.
Signal itemRestored(QString) is emitted for each restored item.
Signal restoreProductsCompleted is emitted on restore process terminated.

Qt C++

```
MainWindow::MainWindow(QWidget *parent) : QMainWindow(parent), ui(new Ui::MainWindow)
{
       connect(iap_manager, SIGNAL(itemRestored(QString)), this, SLOT(itemRestored(QString)));
       connect(iap_manager, SIGNAL(restoreProductsCompleted()), this, SLOT(restoreProductsCompleted()));
}
```

```
void MainWindow::itemRestored(QString productID) {
       qDebug() << "Restoring item :" << productID;
}

void MainWindow::restoreProductsCompleted(){
       qDebug() << "Restore: process terminated";
}
```

QML

```
MenuItem {
     text: qsTr("Restore purchased products");
     onClicked: {
          iap_manager.restoreProducts();
     }
}
```

```
onRestoreProductsCompleted: {
     console.log("onRestoredCompleted")
     if( status===="OK") {
          if( productID === "854595" ) {
               fullVersion = true
          }
     }
}
```


(C) Copyright Nokia 2013. All rights reserved.

Source code

Available the full basic example code here

Links

- Restoration didn't work using production server- In-App-Qt-Symbian

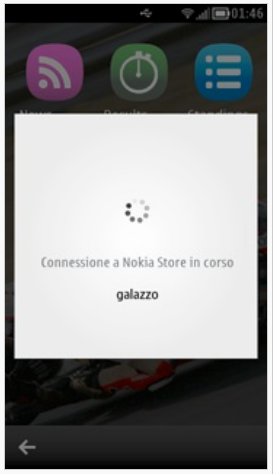
Featured Samples

Superbike

- Superbike. App dedicated to World Superbike Championship. Qlap is used to easily perform the sale of contents.

Start purchasing

Item already purchased, no payment required so will be restored



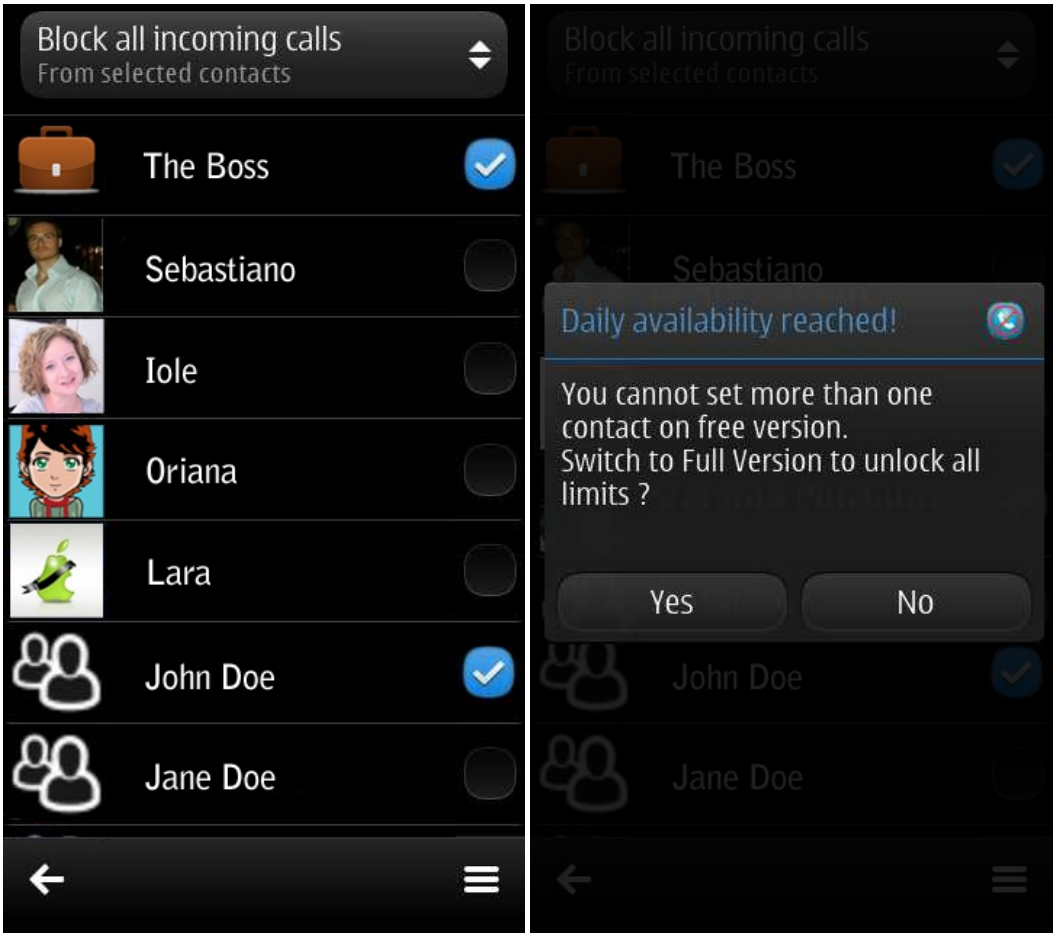
Connecting to server

Last purchasing step using phone credit

Blacklist Manager

- Blacklist Manager. App to blacklist unwanted phone calls. Qlap is used to switch to full version.

(C) Copyright Nokia 2013. All rights reserved.



(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.