How to manage phone calls with QS60Telephony

This article explains how to manage phone calls using QS60Telephony a Qt/Qt Quick wrapper to native Symbian Telephony API (CTelephony).

Introduction

Qt Mobility 1.2 does not currently (13/04/2012) support Telephony operations. QS60Telephony is a wrapper around the Symbian Public Telephony API that allows you to manage phone calls on Symbian devices using either Qt C++ or QML. This article documents the API and shows how to install the package and the libraries it uses. It then provides an example of how to create an app answer or close a phone call and retrieve **IMEI** code, in both QML and Qt C++.

The API allows us to intercept incoming and outgoing calls. The article also shows a possible implementation of a *blacklist management app* which uses this feature to automatically close "unwanted" incoming calls.

API Reference

QML

The call manager is exposed to QML as the `QS60Telephony` element. This element provides the following signals and slots which can be called/handled in JavaScript.

Slots

- `answerIncomingCall()` - answer incoming call
- `terminateCall()` - terminate a connected call
- `call(string)` - call the specified number
- `string IMEI()` - returns the IMEI of the device

Signals (handlers)

- `onAnswered()` - handler for the `answered()` signal. Invoked when the phone call is answered.
- `onTerminated()` - handler for the `answered()` signal. Invoked when the phone call is answered.
- `onLineStatusChanged(int)` - handler for the `lineStatusChanged()` signal. Invoked whenever the line status changes with one of the #Line Status Codes.
- `onError(int)` - handler for the `error()` signal. Invoked whenever an error occurs with one of the #Error Codes.

Qt C++

The call manager class is `QS60Telephony`. It has the following slots and signals **Slots**

- `void rejectCall();` - reject incoming call
- `void terminateCall();` - terminate connected call
- `void answerIncomingCall();` - answer incoming call
- `call(QString)` - call specified phone number
- `QString IMEI();` - get IMEI

Signals

- `void lineStatusChanged(QS60Telephony::LineStatus, QString)` - called when the line status changes with one of the #Line Status Codes.
- `error(QS60Telephony::Error)` - called whenever an error occurs with one of the #Error Codes.
- `void answered()` - invoked when phone call connected.
- `void terminated()` - invoked when phone call terminated.

Line Status Codes

These line status values are signalled by `lineStatusChanged` (handled by `onLineStatusChanged` in QML)

```
enum LineStatus {
        StatusUnknown,
        StatusIdle,
        StatusDialling,
        StatusRinging,
        StatusAnswering,
        StatusConnecting,
        StatusConnected,
        StatusReconnectPending,
        StatusDisconnecting,
        StatusHold,
        StatusTransferring,
        StatusTransferAlerting
};
```


(C) Copyright Nokia 2013. All rights reserved.

Error Codes

```
enum Error {
        NoError = 0,
        OutOfMemoryError,
        AccessDeniedError,
        TimedOutError,
        UnknownError = -1
};
```

Installation

QS60Telephony

1. Download source code from here
2. Include **QS60Telephony** directory into your project

Note: Becouse of capabilities required you need your own developer certificate to run application on your device for testing.

.pro

```
symbian {
    equals(QT_MAJOR_VERSION, 4): greaterThan(QT_MINOR_VERSION, 6) {
         QT+=network
    } else {
         MOBILITY+=bearer
    }
    DEFINES += TELEPHONE_API
    contains(DEFINES, TELEPHONE_API) {
        include(./QS60Telephony/telephony.pri)
    }
}
```

Using QS60Telephony in QML

main.cpp

```
#ifdef TELEPHONE_API
#include <QtDeclarative>
#include "QS60Telephony/qs60telephony.h"
#endif

Q_DECL_EXPORT int main(int argc, char *argv[])
{
    QScopedPointer<QApplication> app(createApplication(argc, argv));

#ifdef TELEPHONE_API
    qmlRegisterType<QS60Telephony>("Telephony", 1, 0, "QS60Telephony");
#endif

    QmlApplicationViewer viewer;
    viewer.setMainQmlFile(QLatin1String("qml/QS60TelephonySample/main.qml"));
    viewer.showExpanded();

    return app->exec();
}
```

QML

```
import QtQuick 1.1
import com.nokia.symbian 1.1
import Telephony 1.0

Page {
    id: mainPage

    QS60Telephony {
        id: callManager
        onAnswered: {
        }

        onTerminated: {
        }

        onLineStatusChanged: {
            console.log(status)
            console.log(number)
            switch(status) {
            case QS60Telephony.StatusRinging : {
                console.log("Ringing");
                callManager.showOnTop();
            }
            break;

            case QS60Telephony.StatusIdle: {
                console.log("Idle");
                callManager.toBackground()
            }
            break;

            case QS60Telephony.StatusDialing: {
                console.log("Dialing:"+number);
```


(C) Copyright Nokia 2013. All rights reserved.

```
            }
            break;

            case QS60Telephony.StatusUnknown: {
                console.log("StatusUnknown");
            }
            break;

            default : break;
            }
        }

        onError: {
        }
    }

    Component.onCompleted: {
        callManager.toBackground();
    }
}
```

Answer Incoming Calls

```
 Rectangle {
            id:answer
            width: 100
            height: 60
            color:"green"
            radius: 5
            Text {
                text: qsTr("Answer")
                color: platformStyle.colorNormalLight
                font.pixelSize: 20
            }

            MouseArea {
                anchors.fill: parent
                onClicked: {
                    callManager.answerIncomingCall();
                }
            }
}
```

Terminate a Call

```
 Rectangle {
    id:terminate
    width: 100; height: 60; color:"red"; radius: 5;

    Text {
        text: qsTr("Close")
        color: platformStyle.colorNormalLight
        font.pixelSize: 20
    }

    MouseArea {
        anchors.fill: parent
        onClicked: {
            callManager.terminateCall();
        }
    }
}
```

Making a Call

```
TextInput {
     id:call_number
}

Rectangle {
    id:call
    width: 100; height: 60;   radius: 5

    MouseArea {
        anchors.fill: parent
        onClicked: {
            callManager.call(call_number.text);
        }
    }
}
```

Getting the phone's IMEI

The **International Mobile Equipment Identity** (IMEI) is a 15 digit code that uniquely identifies your mobile phone. It is a very useful code, for example, the wireless operator will use your **IMEI** code to relate to your account and identify stolen phones, locking them from using the network.

A way to find out your phone's **IMEI** number is to type *#06# on your keyboard. Immediately after you type in these numbers, the phone will display its **IMEI number**. Now let's see how to retrieve **IMEI code** using **QS60Telephony**.

```
Rectangle {
    width: 100; height: 60; radius: 5;

    Text {
        id:imei
        color: platformStyle.colorNormalLight
        font.pixelSize: 20
    }
}
```


(C) Copyright Nokia 2013. All rights reserved.

```
Component.onCompleted: {
    imei.text = callManager.IMEI()
}
```

Using QS60Telephony in Qt C++

In Qt C++ the concept is similar to QML. Assuming your main class is named mainwindow:

mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H
    ....
#include "qs60telephony.h"
    ....

class MainWindow : public QMainWindow
{
    Q_OBJECT
public:
    explicit MainWindow(QWidget *parent = 0);
    virtual ~MainWindow();
    ....
private:
    QS60Telephony* m_telephony;
    QS60Telephony::LineStatus currentStatus;
    ....

private slots:
    void handleStatusChange(QS60Telephony::LineStatus, QString);
    void error(QS60Telephony::Error);
    void answered();
    void terminated();
    void call(QString);
    void dialing(const QString);
    void toBackground();

};
```

mainwindow.cpp

```
MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent),
      ui(new Ui::MainWindow)
{
    m_telephony = new QS60Telephony(this);
    connect(m_telephony, SIGNAL(lineStatusChanged(QS60Telephony::LineStatus, QString)), this, SLOT(handleStatusChange(QS60Telephony:
    connect(m_telephony, SIGNAL(error(QS60Telephony::Error)), this, SLOT(error(QS60Telephony::Error)));
    connect(m_telephony, SIGNAL(answered()), this, SLOT(answered()));
    connect(m_telephony, SIGNAL(terminated()), this, SLOT(terminated()));

   QTimer::singleShot(0, m_telephony, SLOT(startMonitoringLine()));
}

void MainWindow::handleStatusChange(QS60Telephony::LineStatus status, QString number) {

    switch(status) {
    case QS60Telephony::StatusUnknown : break;
    case QS60Telephony::StatusIdle : QTimer::singleShot(1200, m_telephony, SLOT(showNormal())); break;
    case QS60Telephony::StatusDialling : break;
    case QS60Telephony::StatusRinging : QTimer::singleShot(1200, m_telephony, SLOT(showOnTop())); break;
    case QS60Telephony::StatusAnswering :  break;
    case QS60Telephony::StatusConnecting : break;
    case QS60Telephony::StatusConnected : break;
    case QS60Telephony::StatusReconnectPending : break;
    case QS60Telephony::StatusDisconnecting :  QTimer::singleShot(1200, m_telephony, SLOT(showNormal()));  break;
    case QS60Telephony::StatusHold :  break;
    case QS60Telephony::StatusTransferring : break;
    case QS60Telephony::StatusTransferAlerting : break;
    default: qDebug() << "Default status ("<<QString::number(status)<<")"; QTimer::singleShot(1200, m_telephony, SLOT(showNormal()))
    }

}

void MainWindow::error(QS60Telephony::Error) {
     // Here manage errors
}
```

Answer Incoming Calls

```
void MainWindow::on_b_answer_clicked()
{
    m_telephony->answerIncomingCall();
}
```

Terminate a Call

```
void MainWindow::on_b_hangup_clicked()
{
    if(currentStatus==QS60Telephony::StatusRinging){
        m_telephony->rejectCall();
    } else
       m_telephony->terminateCall();
}
```


(C) Copyright Nokia 2013. All rights reserved.

Making a Call

```
void MainWindow::call(QString number) {
    this->m_telephony->call(number);
}
```

Getting the phone's IMEI

The **International Mobile Equipment Identity** (IMEI) is a 15 digit code that uniquely identifies your mobile phone. This is discussed in more detail in the QML section of the same name.

```
void MainWindow::on_b_imei_clicked()
{
    qDebug() << m_telephony->IMEI();
}
```

How to blacklist a phone number

A possible implementation to close immediately an unwanted phone call.

```
void MainWindow::dialing(QString number) {
    if( number.contains("3385482XXX") )
        m_telephony->terminateCall();
}
```

A possible implementation to close after 3 sec an unwanted phone call.

```
void MainWindow::dialing(QString number) {
    if( number.contains("3385482XXX") )
        QTimer::singleShot(3000, m_telephony, SLOT(terminateCall()));
}
```

Sample

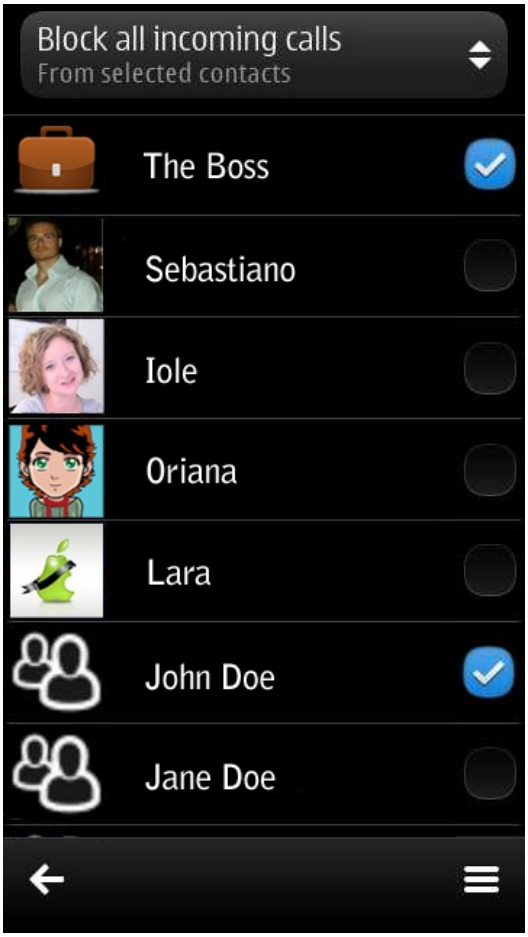
- A sample code can be downloaded for tests here

Note: Becouse of capability type you need your own developer certificate to run application on your device also for tests.

Featured Sample

- Blacklist Manager. App to blacklist unwanted phone calls. QIap - a simple Qt interface for In-App Purchasing 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.