Connection state aware applications for Maemo 5 with Qt Mobility

Archived: This article is **archived** because it is not considered relevant for third-party developers creating commercial solutions today. If you think this article is still relevant, let us know by adding the template {{ReviewForRemovalFromArchive|user=~~~~|*write your reason here*}}.

The article is believed to be still valid for the original topic scope.

This article provides workarounds to the issue that Maemo 5 displays the connection dialog on every attempt to connect to the network. Qt 4.7 should fix these issues, but the code given may still be useful for developers that want to track network connection state changes.

Introduction

Currently, at the moment Maemo 5 Select connection dialog appears on every attempt to connect to the network. User decisions are not remembered thus application easily may become unusable. There are bugs filed to Qt on this issues QTBUG-10343 QTBUG-10503

Before Qt 4.7 is released Qt applications have to workaround this bug.

Solution

One of the possible solutions is to use Qt Mobility Bearer Management. It's available in PR 1.2 release. You can install it with a commands

```
sudo gainroot
apt-get update
apt-get install libqtm-bearer
```

in XTerminal on N900 and inside scratchbox (skip sudo command there). Nokia Qt SDK contains it ready for use.

Project file should enable mobility

```
TEMPLATE = app
TARGET =
DEPENDPATH += .
INCLUDEPATH += .

QT += network

CONFIG += mobility
MOBILITY = bearer

unix:!symbian {
  CONFIG += link_pkgconfig
  PKGCONFIG += gq-gconf
}

# Input
HEADERS += networkapplication.h netmanager.h
SOURCES += main.cpp networkapplication.cpp netmanager.cpp
```

A very basic main.cpp

(C) Copyright Nokia 2014. All rights reserved.

```
#include "networkapplication.h"
#include "netmanager.h"

int main(int argc, char *argv[])
{
    NetworkApplication a(argc, argv);

    NetManager netManager;
    netManager.show();

    return a.exec();
}
```

NetworkApplication header file:

```
#ifndef NETWORK_APPLICATION_H
#define NETWORK_APPLICATION_H

#include <QApplication>
#include <QNetworkConfigurationManager>
#include <QNetworkSession>
#ifdef Q_WS_MAEMO_5
#include <GConfItem>
#endif
using namespace QtMobility;

class NetworkApplication : public QApplication
{
    Q_OBJECT

public:
    NetworkApplication(int &argc, char **argv);

private slots:
    void started();
    void onlineStateChanged(bool isOnline);
#ifdef Q_WS_MAEMO_5
    void valueChanged();
#endif

private:
    QNetworkConfigurationManager *configManager;
    QNetworkSession *session;
#ifdef Q_WS_MAEMO_5
    GConfItem *autoConnect;
    bool autoConnectValue;
#endif
};

#endif //NETWORK_APPLICATION_H
```

```
#include <QMessageBox>
#include <QTimer>
#include <QNetworkConfiguration>
```


(C) Copyright Nokia 2014. All rights reserved.

```
#include "networkapplication.h"

NetworkApplication::NetworkApplication(int &argc, char **argv) :
    QApplication(argc, argv), session(0)
{
#ifdef Q_WS_MAEMO_5
    autoConnect = new GConfItem("/system/osso/connectivity/network_type/auto_connect",
                                this);
    valueChanged();
    connect(autoConnect, SIGNAL(valueChanged()), this, SLOT(valueChanged()));
#endif
    configManager = new QNetworkConfigurationManager(this);
    QTimer::singleShot(0, this, SLOT(started()));
}
```

QTimer::singleShot is used to make sure that slot *started()* is invoked when event loop is already running.

```
#ifdef Q_WS_MAEMO_5
void NetworkApplication::valueChanged()
{
    autoConnectValue = !(autoConnect->value().toString().isEmpty());
}
#endif
```

valueChanged slot is used in monitoring Internet Connection settings, when it changes to/from "Always ask".

```
void NetworkApplication::started()
{
    QCoreApplication::processEvents();
    onlineStateChanged(configManager->isOnline());
    connect(configManager, SIGNAL(onlineStateChanged(bool)),
            this, SLOT(onlineStateChanged(bool)));
}
```

We have to wait until all events are processed, to make sure that QNetworkConfiguationManager is properly initialised, then we test for connection.

```
void NetworkApplication::onlineStateChanged(bool isOnline)
{
    if (!isOnline) {
        QMessageBox::StandardButton reply;
#ifdef Q_WS_MAEMO_5
        if (autoConnectValue)
            reply = QMessageBox::Yes;
        else
#endif
            reply = QMessageBox::question(0,
            "Connection",
            "This application requires connection to the Internet. "
            "Do you want to connect or quit an application?",
            QMessageBox::Yes | QMessageBox::No, QMessageBox::Yes);
        if (reply == QMessageBox::No)
            QCoreApplication::quit();
```


(C) Copyright Nokia 2014. All rights reserved.

```
        else {
            if(session) {
                delete session;
                session = 0;
            }
            QNetworkConfiguration config = configManager->defaultConfiguration();
            session = new QNetworkSession(config, this);
            session->open();
            session->waitForOpened(-1);
            if(!autoConnectValue)
                onlineStateChanged(configManager->isOnline());
        }
    }
}
```

In case connection is not available user is proposed to quit application or choose a valid connection.

NetManager header file:

```
#ifndef NET_MANAGER_H
#define NET_MANAGER_H

#include <QNetworkAccessManager>
#include <QNetworkReply>
#include <QLabel>
#include <QPushButton>

class NetManager : public QWidget
{
    Q_OBJECT

public:
    NetManager(QWidget *parent=0);

private slots:
    void clicked(bool checked = false);
    void finished(QNetworkReply* reply);

private:
    QPushButton *button;
    QLabel *label;
    QNetworkAccessManager *nam;
};

#endif //NET_MANAGER_H
```

Implementation of NetManager, which is used to initiate network request and process a network reply.

```
#include <QDebug>
#include <QVBoxLayout>

#include "netmanager.h"

NetManager::NetManager(QWidget *parent)
    : QWidget(parent), nam(0)
{
```


(C) Copyright Nokia 2014. All rights reserved.

```
    QVBoxLayout *layout = new QVBoxLayout;
    button = new QPushButton("Get page");
    connect(button, SIGNAL(clicked(bool)), this, SLOT(clicked(bool)));
    layout->addWidget(button);
    label = new QLabel("");
    layout->addWidget(label);
    setLayout(layout);
}

void NetManager::clicked(bool checked)
{
    Q_UNUSED(checked);

    if(!nam) {
        nam = new QNetworkAccessManager(this);
        connect(nam, SIGNAL(finished(QNetworkReply*)),
                this, SLOT(finished(QNetworkReply*)));
    }
    nam->get(QNetworkRequest(QUrl("http://tycho.usno.navy.mil/cgi-bin/timer.pl")));
}

void NetManager::finished(QNetworkReply *reply)
{
    if (reply->error() == QNetworkReply::NoError) {
        label->setText(reply->readAll());
    } else {
        label->setText(reply->errorString());
    }

    reply->deleteLater();
}
```

Package

There is a bearer-qt-sample package available in extas-devel repository.

bearer-qt-sample.install file:

```
[install]
catalogues = extras-devel
package   = bearer-qt-sample

[extras-devel]
name = Extras-Devel
uri = http://repository.maemo.org/extras-devel/
dist = fremantle
components = free non-free
```


(C) Copyright Nokia 2014. All rights reserved.

 (C) Copyright Nokia 2014. All rights reserved.