Read and send SMS messages using the Qt Messaging API

This code example shows how to use the Qt Messaging API to read and send SMS messages.

This article needs to be updated: If you found this article useful, please fix the problems below then delete the {{ArticleNeedsUpdate}} template from the article to remove this warning.

Reasons: hamishwillee
(16 Feb 2012)
Usage of Qt Mobility API is through QWidget app framework. Should be updated to be accessed through Qt Quick.

Headers required

```
/* This include is not portable and should be avoided */
#include "qtmessaging.h"
```

.pro file

Modify your .pro file as follows.

```
CONFIG += mobility
MOBILITY = messaging
/* To show icon on list box */
RESOURCES += Messages.qrc
symbian:TARGET.CAPABILITY = NetworkServices \
    LocalServices \
    ReadUserData \
    WriteUserData \
    UserEnvironment \
    ReadDeviceData \
    WriteDeviceData
```

Add resource file

Add .qrc file to project to show icon in list widget.

```
<!DOCTYPE RCC><RCC version="1.0">
 <qresource>
     <file>rsc/Inbox.jpg</file>
 </qresource>
 </RCC>
```

Header files

```
/* Messages.h*/
#ifndef MESSAGING_H
#define MESSAGING_H

#include <QWidget>
/* This include is not portable */
#include "qtmessaging.h"

QT_BEGIN_NAMESPACE
class QStackedWidget;
QT_END_NAMESPACE

QTM_USE_NAMESPACE;

/* forward declaration */
class MainPage;
class SendSMS;
class ListBox;

class Messaging : public QWidget
{
    Q_OBJECT

public:
 Messaging(QWidget *parent = 0);
    ~Messaging();

public slots:
    void activateSendSMS();
    void activateMainPage();
    void activateInbox();
    void activateSentItem();
    void activateDraft();
    void activateOutbox();
    void messageServiceStateChanged(QMessageServiceAction::State s);


private:
 /* The stacked widget which will display the pages */
 QStackedWidget *m_stackedWidget;
```


(C) Copyright Nokia 2013. All rights reserved.

```
 /* Widget to show main page/each folder in messaging */
 MainPage *m_mainPage;
 /* widget to create new message */
 SendSMS *m_sendSMS;
 /*Widget to show each message of a folder */
 ListBox *m_listBox;

 QMessageId messageId;
 QMessageServiceAction *messageService;
};

#endif // MESSAGING_H
```

```
/* MainPage.h */
#ifndef MAINPAGE_H
#define MAINPAGE_H

#include <QtGui/QMainWindow>
/* This include is not portable */
#include "qtmessaging.h"

QT_BEGIN_NAMESPACE
class QListWidget;
class QScrollArea;
class QListWidgetItem;
class QAction;
QT_END_NAMESPACE

QTM_USE_NAMESPACE;

class MainPage : public QMainWindow
{
    Q_OBJECT

public:
 MainPage(QWidget *parent = 0);
    ~MainPage();

private:
 void countMessagesInFolders();

signals:
 void createMessage();
 void createInbox();
 void createSentItem();
 void createDraft();
 void createOutbox();

public slots:
 void getSelectedItem(QListWidgetItem* item);
 void getCurrentRow(int a);
 void openItem();


private:
 QListWidget     *listWidget;
 QScrollArea     *scrollArea;
 int selectedRow;
 int previousRow;

    //Actions for menu item
    QAction* rightSoftKeyAction;


 int countInboxSMS;
 int countSentSMS;
 int countDraftSMS;
 int countOutboxSMS;
};

#endif // MAINPAGE_H
```

```
/* ListBox.h */
#ifndef LISTBOX_H
#define LISTBOX_H

#include <QtGui/QMainWindow>

class QListWidget;
class QScrollArea;
class QListWidgetItem;

class ListBox : public QMainWindow
{
    Q_OBJECT

public:
 ListBox(QWidget *parent = 0);
    ~ListBox();

signals:
 void gotoMainPage();

public slots:
 void getSelectedItem(QListWidgetItem* item);
 void getCurrentRow(int a);
    void closeListBox();
    void openItem();
    void setItemText(const QStringList& stringList);

private:
 QAction *rightSoftKeyAction;
 QAction *leftSoftKeyAction;
 QListWidget     *listWidget;
 QScrollArea     *scrollArea;
 int selectedRow;
```


(C) Copyright Nokia 2013. All rights reserved.

```
 int previousRow;
};

#endif // ListBox_H
```

```
/* SendSMS.h */
#ifndef SENDSMS_H
#define SENDSMS_H

#include <QWidget>
/* This include is not portable */
#include "qtmessaging.h"

QT_BEGIN_NAMESPACE
class QAction;
class QTextEdit;
class QLabel;
class QLineEdit;
class QGridLayout;
class QVBoxLayout;
class QListWidget;
QT_END_NAMESPACE

QTM_USE_NAMESPACE;

class SendSMS : public QWidget
{
    Q_OBJECT

public:
 SendSMS(QWidget *parent = 0);
    ~SendSMS();

 void setMessageServiceAction(QMessageServiceAction *messageService);

signals:
 void gotoMainPage();

private slots:
    void sendMessage();
    void closeEditor();


private:
    QAction *softKeyAction;
    QAction *leftSoftKeyAction;
    QLineEdit *mobileNumberInput;
    QTextEdit *messageBodyInput;
    QLabel *labelTo;

    QGridLayout *layout;
    QVBoxLayout *mainLayout;

    QMessageId messageId;
    QMessageServiceAction *messageService;
};

#endif // SENDSMS_H
```

Source files

```
/* Messages.cpp*/
#include "MainPage.h"
#include "SendSMS.h"
#include "ListBox.h"
#include "Messaging.h"
#include <QMessageStore>
#include <QStackedWidget>
#include <QVBoxLayout>
#include <QMessageBox>

Messaging::Messaging(QWidget *parent)
    : QWidget(parent)
{
 setWindowTitle(tr("Messaging"));

 /* initialize main page widget*/
 m_mainPage = new MainPage(this);
 connect(m_mainPage, SIGNAL(createMessage()), this, SLOT(activateSendSMS()));
 connect(m_mainPage, SIGNAL(createInbox()), this, SLOT(activateInbox()));
 connect(m_mainPage, SIGNAL(createSentItem()), this, SLOT(activateSentItem()));
 connect(m_mainPage, SIGNAL(createDraft()), this, SLOT(activateDraft()));
 connect(m_mainPage, SIGNAL(createOutbox()), this, SLOT(activateOutbox()));

 /* initialize widget to send SMS */
 m_sendSMS = new SendSMS(this);
 connect(m_sendSMS, SIGNAL(gotoMainPage()), this, SLOT(activateMainPage()));

 /* initialize list widget  */
 m_listBox = new ListBox(this);

 connect(m_listBox, SIGNAL(gotoMainPage()), this, SLOT(activateMainPage()));

 m_stackedWidget = new QStackedWidget(this);
 m_stackedWidget->addWidget(m_mainPage);
 m_stackedWidget->addWidget(m_sendSMS);
 m_stackedWidget->addWidget(m_listBox);
 m_stackedWidget->setCurrentIndex(0);


 QVBoxLayout *centralLayout = new QVBoxLayout;
 centralLayout->addWidget(m_stackedWidget);

 /* The QWidget will take ownership of centralLayout. So no need to delete centralLayout*/
 setLayout(centralLayout);

 messageService = new QMessageServiceAction(this);
```


(C) Copyright Nokia 2013. All rights reserved.

```
 connect(messageService, SIGNAL(stateChanged(QMessageServiceAction::State)), this, SLOT(messageServiceStateChanged(QMessageServiceAction::State)));

}

/* Activate SensSMS widget */
void Messaging::activateSendSMS()
{
 setWindowTitle(tr("Text message"));
 m_sendSMS->setMessageServiceAction(messageService);
 m_stackedWidget->setCurrentIndex(1);
}

/*Activate main page widget */
void Messaging::activateMainPage()
{
 setWindowTitle(tr("Messaging"));
 m_stackedWidget->setCurrentIndex(0);
}

/* activate widget to show inbox */
void Messaging::activateInbox()
{
 setWindowTitle(tr("Inbox"));

 QMessageFilter messageFilter(QMessageFilter::byStandardFolder (QMessage::InboxFolder));
 QMessageOrdering messageOrdering(QMessageOrdering::byReceptionTimeStamp(Qt::DescendingOrder));

 const QMessageIdList messageIdList(QMessageStore::instance()->queryMessages(messageFilter, messageOrdering));

 QMessage message;

 QStringList itemList;

 for(int i = 0; i < messageIdList.count(); ++i)
 {
  message = QMessageStore::instance()->message(messageIdList.at(i));
  QMessageAddress from = message.from();
  QString number(from.recipient());
  QString text;
  text.append( number + ": " + message.find(message.bodyId()).textContent());
  itemList.append(text);
 }
 m_listBox->setItemText(itemList);
 m_stackedWidget->setCurrentIndex(2);
}

/* activate widget to show sent item */
void Messaging::activateSentItem()
{
 setWindowTitle(tr("Sent"));
 QMessageFilter messageFilter(QMessageFilter::byStandardFolder (QMessage::SentFolder));
 QMessageOrdering messageOrdering(QMessageOrdering::byReceptionTimeStamp(Qt::DescendingOrder));

 const QMessageIdList messageIdList(QMessageStore::instance()->queryMessages(messageFilter, messageOrdering));

 QMessage message;

 QStringList itemList;

 for(int i = 0; i < messageIdList.count(); ++i)
 {
  message = QMessageStore::instance()->message(messageIdList.at(i));
  QMessageAddress from = message.from();
  QString number(from.recipient());
  QString text;
  text.append( number + ": " + message.find(message.bodyId()).textContent());
  itemList.append(text);
 }
 m_listBox->setItemText(itemList);
 m_stackedWidget->setCurrentIndex(2);
}

/* activate widget to show draft */
void Messaging::activateDraft()
{
 setWindowTitle(tr("Draft"));
 QMessageFilter messageFilter(QMessageFilter::byStandardFolder (QMessage::DraftsFolder));
 QMessageOrdering messageOrdering(QMessageOrdering::byReceptionTimeStamp(Qt::DescendingOrder));

 const QMessageIdList messageIdList(QMessageStore::instance()->queryMessages(messageFilter, messageOrdering));

 QMessage message;

 QStringList itemList;

 for(int i = 0; i < messageIdList.count(); ++i)
 {
  message = QMessageStore::instance()->message(messageIdList.at(i));
  QMessageAddress from = message.from();
  QString number(from.recipient());
  QString text;
  text.append( number + ": " + message.find(message.bodyId()).textContent());
  itemList.append(text);
 }
 m_listBox->setItemText(itemList);
 m_stackedWidget->setCurrentIndex(2);
}

/* activate widget to show sent outbox */
void Messaging::activateOutbox()
{
 setWindowTitle(tr("Outbox"));
 QMessageFilter messageFilter(QMessageFilter::byStandardFolder (QMessage::OutboxFolder));
 QMessageOrdering messageOrdering(QMessageOrdering::byReceptionTimeStamp(Qt::DescendingOrder));

 const QMessageIdList messageIdList(QMessageStore::instance()->queryMessages(messageFilter, messageOrdering));

 QMessage message;

 QStringList itemList;

 for(int i = 0; i < messageIdList.count(); ++i)
 {
  message = QMessageStore::instance()->message(messageIdList.at(i));
  QMessageAddress from = message.from();
  QString number(from.recipient());
  QString text;
  text.append( number + ": " + message.find(message.bodyId()).textContent());
  itemList.append(text);
 }
```


(C) Copyright Nokia 2013. All rights reserved.

```
 m_listBox->setItemText(itemList);
 m_stackedWidget->setCurrentIndex(2);
}


void Messaging::messageServiceStateChanged(QMessageServiceAction::State s)
{
    if (s == QMessageServiceAction::Successful)
    {
        QMessageBox::information(0, "Message sent", "Message sent successfully");
    }
    else if (s == QMessageServiceAction::Failed)
    {
        QMessageBox::warning(0, "Failed", "Unable to send message");

        if (!QMessageStore::instance()->removeMessage(messageId))
        {
   QMessageBox::warning(0, "Unable to remove", "Unable to remove message: " + messageId.toString());
        }
        messageId = QMessageId();
    }
}

Messaging::~Messaging()
{
}
```

```
/* MainPage.cpp */
#include "MainPage.h"
#include <QListWidget>
#include <QScrollArea>
#include <QMessageBox>
#include <QItemSelectionModel>
#include <QKeyEvent>
#include <QAction>

MainPage::MainPage(QWidget *parent)
    : QMainWindow(parent)
{
 countMessagesInFolders();

 /* To show items in list view */
 listWidget = new QListWidget();
 listWidget->setObjectName(QString::fromUtf8("listWidget"));

 /* To make UI scrollable */
 scrollArea = new QScrollArea();

 /* scrollArea takes ownership of the listWidget, so no need to delete it */
 scrollArea->setWidget(listWidget);
 scrollArea->setAlignment(Qt::AlignLeft);
 scrollArea->setWidgetResizable(true);

 /* QMainWindow takes ownership of the scrollArea, so no need to delete it */
 setCentralWidget(scrollArea);

 connect(listWidget, SIGNAL(itemClicked(QListWidgetItem*)), this, SLOT(getSelectedItem(QListWidgetItem*)));
 connect(listWidget, SIGNAL(currentRowChanged (int)), this, SLOT(getCurrentRow(int)));

 /* set font and icon size of listWiget */
 QFont listWidgetFont("Courier", 8, QFont::Normal);
 listWidget->setIconSize(QSize(60 ,70));

 /* The listWidget will take ownership of the each QListWidgetItem item, so no need to delete it. */
 QListWidgetItem* listItemNewMessage = new QListWidgetItem;
 listItemNewMessage->setFont(listWidgetFont);
 listItemNewMessage->setText("New message");
 /* set list item icon */
 listItemNewMessage->setIcon(QIcon(":/rsc/Inbox.jpg"));
 listItemNewMessage->setSizeHint(QSize(size().width() - 200 ,70));
 listWidget->addItem(listItemNewMessage);

 QString listItemText;
 QListWidgetItem* listItemInbox = new QListWidgetItem;
 listItemInbox->setFont(listWidgetFont);
 listItemText.setNum(countInboxSMS);
 listItemText.prepend("Inbox ");
 listItemText.append(" messages");
 listItemInbox->setText(listItemText);
 listItemInbox->setIcon(QIcon(":/rsc/Inbox.jpg"));
 listItemInbox->setSizeHint(QSize(size().width() - 200 ,70));
 listWidget->addItem(listItemInbox);

 QListWidgetItem* listItemSent = new QListWidgetItem;
 listItemSent->setFont(listWidgetFont);
 listItemText.clear();
 listItemText.setNum(countSentSMS);
 listItemText.prepend("Sent ");
 listItemText.append(" messages");
 listItemSent->setText(listItemText);
 listItemSent->setIcon(QIcon(":/rsc/Inbox.jpg"));
 listItemSent->setSizeHint(QSize(size().width() - 200 ,70));
 listWidget->addItem(listItemSent);

 QListWidgetItem* listItemDraft = new QListWidgetItem;
 listItemDraft->setFont(listWidgetFont);
 listItemText.clear();
 listItemText.setNum(countDraftSMS);
 listItemText.prepend("Draft ");
 listItemText.append(" messages");
 listItemDraft->setText(listItemText);
 listItemDraft->setIcon(QIcon(":/rsc/Inbox.jpg"));
 listItemDraft->setSizeHint(QSize(size().width() - 200 ,70));
 listWidget->addItem(listItemDraft);

 QListWidgetItem* listItemOutbox = new QListWidgetItem;
 listItemOutbox->setFont(listWidgetFont);
 listItemText.clear();
 listItemText.setNum(countOutboxSMS);
 listItemText.prepend("Outbox ");
 listItemText.append(" messages");
```


(C) Copyright Nokia 2013. All rights reserved.

```
 listItemOutbox->setText(listItemText);
 listItemOutbox->setIcon(QIcon(":/rsc/Inbox.jpg"));
 listItemOutbox->setSizeHint(QSize(size().width() - 200 ,70));
 listWidget->addItem(listItemOutbox);


 listWidget->setCurrentRow(0,QItemSelectionModel::Select);

 selectedRow = previousRow = 0;

 rightSoftKeyAction = new QAction(tr("Open"),this);
 rightSoftKeyAction->setSoftKeyRole(QAction::PositiveSoftKey);
 addAction(rightSoftKeyAction);

 connect(rightSoftKeyAction, SIGNAL(triggered()), this, SLOT(openItem()));
}

void MainPage::getSelectedItem(QListWidgetItem* item)
{
 if(selectedRow == previousRow)
 {
  if(!item->text().compare("New message"))
   createMessage();
  else if(!item->text().left(5).compare("Inbox"))
   createInbox();
  else if(!item->text().left(4).compare("Sent"))
   createSentItem();
  else if(!item->text().left(5).compare("Draft"))
   createDraft();
  else if(!item->text().left(6).compare("Outbox"))
   createOutbox();
 }
 else
 {
  previousRow = selectedRow;
 }
}


void MainPage::getCurrentRow(int row)
{
 previousRow = selectedRow;
 selectedRow = row;
}

void MainPage::openItem()
{
 getSelectedItem(listWidget->currentItem());
}

void MainPage::countMessagesInFolders()
{

 /* Count SMS in inbox */
 QMessageFilter messageFilter(QMessageFilter::byStandardFolder (QMessage::InboxFolder));
 QMessageOrdering messageOrdering(QMessageOrdering::byReceptionTimeStamp(Qt::DescendingOrder));
 QMessageIdList messageIdList(QMessageStore::instance()->queryMessages(messageFilter, messageOrdering));

 countInboxSMS = messageIdList.count();

 /* Count SMS in sent item folder */
 messageFilter = QMessageFilter::byStandardFolder (QMessage::SentFolder);
 messageOrdering = QMessageOrdering::byReceptionTimeStamp(Qt::DescendingOrder);
 messageIdList = QMessageStore::instance()->queryMessages(messageFilter, messageOrdering);

 countSentSMS = messageIdList.count();

 /* Count SMS in draft folder */
 messageFilter = QMessageFilter::byStandardFolder (QMessage::DraftsFolder);
 messageOrdering = QMessageOrdering::byReceptionTimeStamp(Qt::DescendingOrder);
 messageIdList = QMessageStore::instance()->queryMessages(messageFilter, messageOrdering);

 countDraftSMS = messageIdList.count();


 /* Count SMS in outbox folder */
 messageFilter = QMessageFilter::byStandardFolder (QMessage::OutboxFolder);
 messageOrdering = QMessageOrdering::byReceptionTimeStamp(Qt::DescendingOrder);
 messageIdList = QMessageStore::instance()->queryMessages(messageFilter, messageOrdering);

 countOutboxSMS = messageIdList.count();

}

MainPage::~MainPage()
{

}
```

```
/* ListBox.cpp */
#include "ListBox.h"
#include <QListWidget>
#include <QScrollArea>
#include <QMessageBox>
#include <QAction>

ListBox::ListBox(QWidget *parent)
    : QMainWindow(parent)
{
 rightSoftKeyAction = new QAction(tr("Open"),this);
 rightSoftKeyAction->setSoftKeyRole(QAction::PositiveSoftKey);
 addAction(rightSoftKeyAction);

 connect(rightSoftKeyAction, SIGNAL(triggered()), this, SLOT(openItem()));

 leftSoftKeyAction = new QAction(tr("Back"),this);
 leftSoftKeyAction->setSoftKeyRole(QAction::NegativeSoftKey);
 addAction(leftSoftKeyAction);

 connect(leftSoftKeyAction, SIGNAL(triggered()), this, SLOT(closeListBox()));

 /* To show items in list view */
```


(C) Copyright Nokia 2013. All rights reserved.

```
 listWidget = new QListWidget();
 listWidget->setObjectName(QString::fromUtf8("listWidget"));

 /* To make UI scrollable */
 scrollArea = new QScrollArea();

 /* scrollArea takes ownership of the listWidget, so no need to delete it */
 scrollArea->setWidget(listWidget);
 scrollArea->setAlignment(Qt::AlignLeft);
 scrollArea->setWidgetResizable(true);

 /* QMainWindow takes ownership of the scrollArea, so no need to delete it */
 setCentralWidget(scrollArea);

 connect(listWidget, SIGNAL(itemClicked(QListWidgetItem*)), this, SLOT(getSelectedItem(QListWidgetItem*)));
 connect(listWidget, SIGNAL(currentRowChanged (int)), this, SLOT(getCurrentRow(int)));

}

void ListBox::getSelectedItem(QListWidgetItem* item)
{
 if(selectedRow == previousRow)
 {
  QMessageBox::information(0,"Text message", item->text());
 }
 else
 {
  previousRow = selectedRow;
 }
}

void ListBox::closeListBox()
{
 gotoMainPage();
}

void ListBox::openItem()
{
 getSelectedItem(listWidget->currentItem());
}

void ListBox::setItemText(const QStringList& stringList)
{
 listWidget->clear();
 foreach ( QString str, stringList )
 {
  /* set font of listWiget */
  QFont newFont("Courier", 6, QFont::Normal);

  /* The listWidget will take ownership of the each QListWidgetItem item, so no need to delete it. */
  QListWidgetItem* listItem = new QListWidgetItem;
  listItem->setFont(newFont);
  listItem->setText(str);
  listItem->setSizeHint(QSize(size().width() - 200 ,70));
  listWidget->addItem(listItem);
 }

 if(stringList.count() <= 0)
 {
  listWidget->addItem("No messages");
  listWidget->setCurrentRow(0);
 }
 else
  listWidget->setCurrentRow(0,QItemSelectionModel::Select);

 selectedRow = previousRow = 0;
}

void ListBox::getCurrentRow(int row)
{
 selectedRow = row;
}

ListBox::~ListBox()
{
}
```

```
/* SendSMS.cpp */
#include "SendSMS.h"

#include <QAction>
#include <QTextEdit>
#include <QLabel>
#include <QLineEdit>
#include <QGridLayout>
#include <QVBoxLayout>
#include <QListWidget>
#include <QMessageBox>
#include <QDebug>

SendSMS::SendSMS(QWidget *parent)
    : QWidget(parent)
{
 softKeyAction = new QAction(tr("Send"),this);
 softKeyAction->setSoftKeyRole(QAction::PositiveSoftKey);
 addAction(softKeyAction);

 connect(softKeyAction, SIGNAL(triggered()), this, SLOT(sendMessage()));

 leftSoftKeyAction = new QAction(tr("Close"),this);
 leftSoftKeyAction->setSoftKeyRole(QAction::NegativeSoftKey);
 addAction(leftSoftKeyAction);

 connect(leftSoftKeyAction, SIGNAL(triggered()), this, SLOT(closeEditor()));

 /* editor to get mobile number input */
 mobileNumberInput = new QLineEdit(this);

 labelTo = new QLabel(tr("To"),this);
 labelTo->setSizePolicy(QSizePolicy::Fixed, QSizePolicy::Fixed);

 layout = new QGridLayout;
```


(C) Copyright Nokia 2013. All rights reserved.

```
 layout->setContentsMargins(0, 0, 0, 0);
 layout->addWidget(labelTo, 0, 0);
 layout->setAlignment(labelTo, Qt::AlignRight);
 layout->addWidget(mobileNumberInput, 0, 1);

 /* Editor to get text message input */
 messageBodyInput = new QTextEdit(this);

 mainLayout = new QVBoxLayout(this);
 mainLayout->setContentsMargins(0, 0, 0, 0);
 mainLayout->addLayout(layout, 0);
 mainLayout->addWidget(messageBodyInput, 2);
}

SendSMS::~SendSMS()
{
 delete layout;
}

void SendSMS::sendMessage()
{
 //QMessageBox::warning(0, "Message Sent", "Your message is sent successfully");

 QString to(mobileNumberInput->text());

 /* check if receipient number is empty */
 if (to.isEmpty())
 {
  QMessageBox::warning(0, "Missing information", "Please enter a recipient mobile number");
  return;
 }

 /* check if input text is empty */
 QString text(messageBodyInput->toPlainText());
 if (text.isEmpty())
 {
  QMessageBox::warning(0, "Missing information", "Please enter a text message");
  return;
 }

 mobileNumberInput->setText(tr(""));
 messageBodyInput->setText(tr(""));
 gotoMainPage();


 QMessage message;
 message.setType(QMessage::Sms);
 message.setTo(QMessageAddress(to, QMessageAddress::Phone));
 message.setBody(text);

 /* send message */
 messageService->send(message);
 messageId = message.id();

}

void SendSMS::setMessageServiceAction(QMessageServiceAction *messService)
{
 messageService = messService;
}

void SendSMS::closeEditor()
{

 QString text(messageBodyInput->toPlainText());
 int ret;

 /* check if input text is empty */
 if (!text.isEmpty())
 {
  ret = QMessageBox::question(0, "Text message", "Do you want to save message to draft??" , QMessageBox::Ok, QMessageBox::No);
 }

 if(QMessageBox::Ok == ret)
 {
 //save to draft
 }

 mobileNumberInput->setText(tr(""));
 messageBodyInput->setText(tr(""));

 /* go to main page */
 gotoMainPage();
}
```

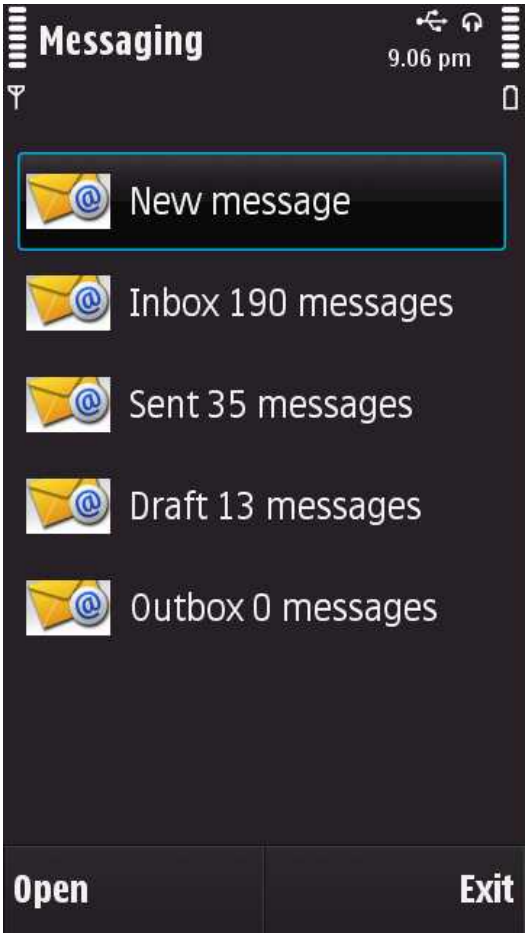
Post-conditions

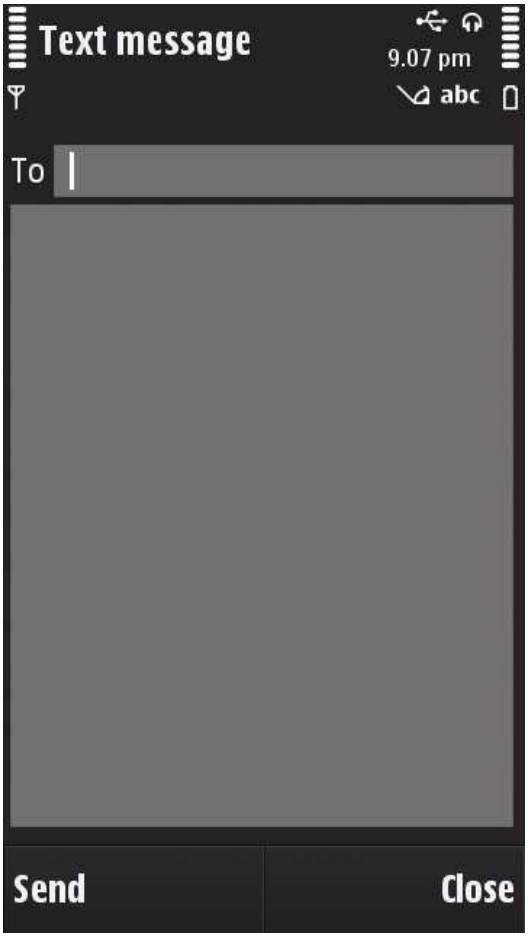
The code snippet is expected to read all messages from selected folder and create & send new message.

Display all messaging folders

Create and send new message

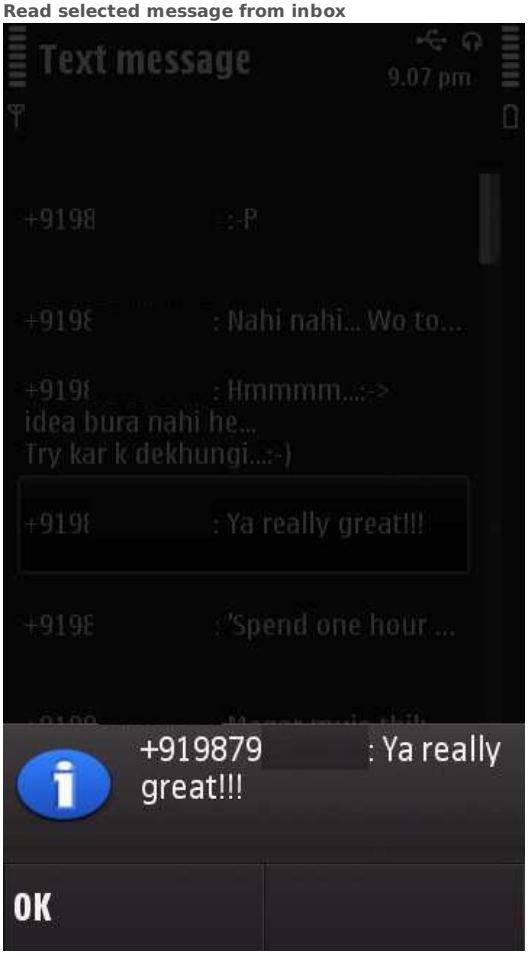
(C) Copyright Nokia 2013. All rights reserved.





Show inbox folder

Read selected message from inbox



Show sent item folder

(C) Copyright Nokia 2013. All rights reserved.

Download Code Example

- Working Code Example is available to download from here.

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.