Migrating from Avkon to QWidget: UI component comparison

21 Nov 2010

Introduction

This document is intended to be a reference guide for Symbian developers who are implementing user interfaces with Qt. The document lists the most-often-used Symbian Avkon UI components and their Qt counterparts (QWidgets). Source code will sometimes need to be customised for an optimal Symbian look and feel.

The illustrations used in this document depict Qt 4.6.3 on a Nokia N8 device based on Symbian^3.

CCoeControl vs. QWidget

Symbian and Qt have their own base classes for the UI: QWidget in Qt and CCoeControl in Symbian. Both classes can work as a container control or parent class for the other child components, and they have fairly similar features, for example

- Handling mouse and key events, catching events, handling event flow between parent and child components.
- When a component is created or window size is changed, the same kind of method is called to inform about the event: `CCoeControl::SizeChanged()` in Symbian and `QWidget::resizeEvent()` in Qt. In these methods it is necessary to define component position and size: `CCoeControl::SetRect()` in Symbian and `QWidget::setGeometry()` in Qt. There is, of course, a chance to use layouts in Qt, and it is even advisable.

Additional similarities include:

- Dialogues have their own base classes: CAknDialog in Symbian and QDialog in Qt.
- Applications have their own class that creates and sets the geometry of the main window: CAknApplication in Symbian and QApplication in Qt.

Switching between views

Qt does not support Symbian-style application view switching, where different applications can open each other's views. Symbian views are described further in the View Server Tutorial and in Switching views on Symbian.

A Qt mobile application is usually built on QMainWindow, and it has one QWidget as the main central widget. Switching between different widgets can be implemented by, for example:

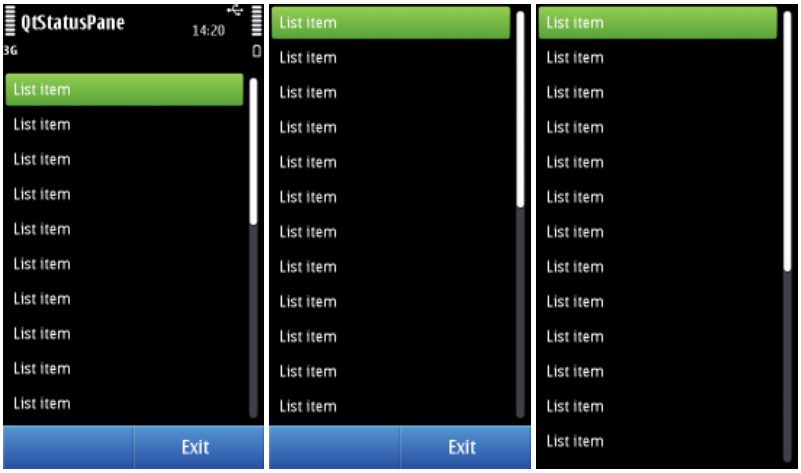
- Using QStackedWidget as the central widget, which allows you to stack different QWidgets;
- Opening QDialog on top of the central widget;
- Using multiple QMainWindow instances where you open another QMainWindow on top of the first one. Define `QMainWindow::setAttribute(Qt::WA_DeleteOnClose)` for the opened windows, which are then deleted after close.

UI Scalability

All Symbian and Qt components scale dynamically. For a Qt application with multiple widgets in one view, use layouts to support automatic scaling. Sometimes Qt components need more space than is available in a mobile device. In that case, try using the `SetNoConstraint` flag in QLayout; for more information, see QLayout::setSizeConstraint(). This will allow components to be smaller than what has been defined as the Qt minimum.

Status pane and CBA buttons

CBA buttons and the status pane are visible automatically when running a Qt application on its maximised mode, `QWidget::showMaximized()`.



(C) Copyright Nokia 2013. All rights reserved.

If you want to run a Qt application on full screen mode, `QWidget::showFullScreen()`, there is no status pane or CBA buttons visible.

On full screen mode it is possible to make CBA buttons visible by enabling the `Qt::WindowSoftkeysVisibleHint` flag into your window:

```
QWidget::setWindowFlags(windowFlags() | Qt::WindowSoftkeysVisibleHint);
```

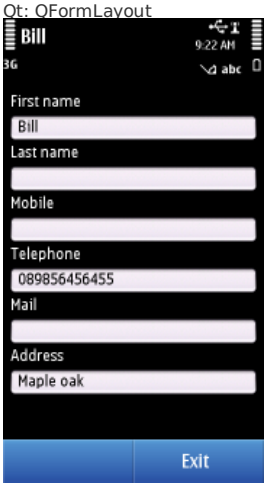
Note that there is a slight difference in the placement of softkeys in Symbian^3 - they are in the bottom of the Avkon component. In QWidget they are in the CBA pane. This makes no difference when using the application, though.

Component Quick Reference

Qt UI component	**Symbian UI component**
QWidget	CCoeControl
QMenuBar, QMenu	RESOURCE MENU_BAR, RESOURCE MENU_PANE
QAction	RESOURCE MENU_ITEM
QDialog	CAknDialog
QFormLayout	CAknForm
QMessageBox	CAknInformationNote, CAknConfirmationNote, CAknWarningNote, CAknErrorNote
QProgressDialog	CAknProgressDialog
QProgressBar	CEikProgressInfo
QColorDialog	CAknColourSelectionGrid
QFileDialog	AknCommonDialogsDynMem
QInputDialog	CAknTextQueryDialog
QListWidget	CAknSingleStyleListBox, CAknSingleGraphicStyleListBox, CAknDoubleStyleListBox
QGridLayout	CAknGrid
QGroupBox	
QSlider	CAknSlider
QDial	
QLineEdit	CEikEdwin
QPushButton	CAknButton
QRadioButton	CAknRadioButtonSettingPage
QCheckBox	CAknCheckBoxSettingPage
QLabel	CEikLabel
QTextEdit	CEikGlobalTextEditor
QTextBrowser	CEikRichTextEditor
QTabWidget	CAknTabGroup
QComboBox	CAknChoiceList
QScrollBar	CEikScrollBarFrame
QDateEdit	CEikDateEditor
QTimeEdit	CEikTimeEditor
QToolBar	CAknToolbar, CAknButton

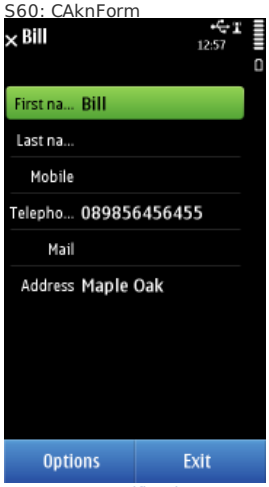
Layouts

Qt: QFormLayout



Layout Management, QFormLayout

S60: CAknForm



Form API Specification

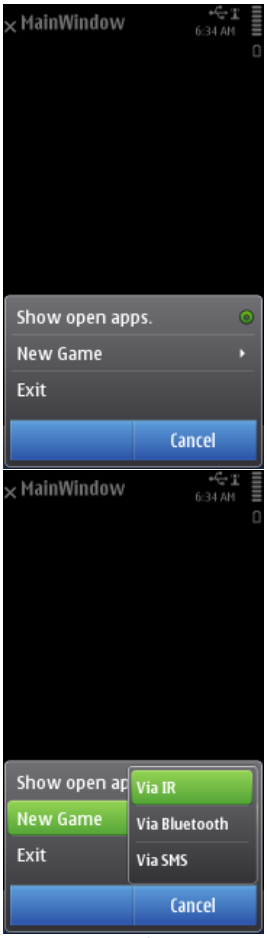
Qt has its own form layout class for building forms. Symbian has a different solution, where the form line is DLG_LINE in the DIALOG item in the application resource file.

Menu Bar

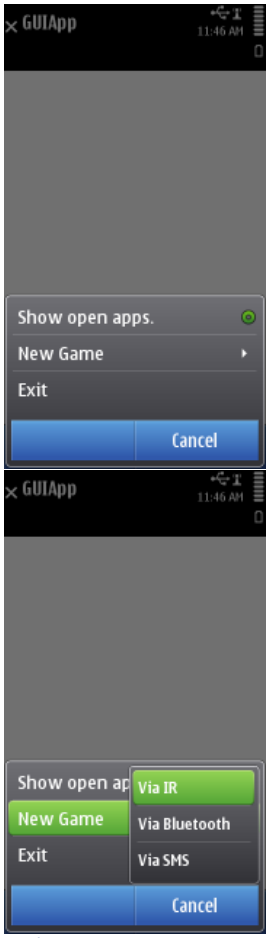
Qt: QMenuBar, QMenu, QAction

S60: RESOURCE MENU_BAR, RESOURCE MENU_PANE, MENU_ITEM

(C) Copyright Nokia 2013. All rights reserved.



QMenuBar, QAction



Options Menu API

Symbian defines menus in application resource files, whereas QMenuBar and actions are created in the source code in Qt. In Qt, the Options menu can be defined in QMainWindow or QDialog.

In QMainWindow, the menu can be created in one line, such as:

```
menuBar()->addAction("Menu item", this, SLOT(someSlot()));
```

The menu in QDialog can be implemented using QMenu:

```
menu = new QMenu(this);
menu->addAction("Menu item", this, SLOT(someSlot()));
```

In QDialog, you have to define CBA buttons, too:

```
QAction *optionsAction = new QAction("Options", this);
optionsAction->setMenu(menu);
optionsAction->setSoftKeyRole(QAction::PositiveSoftKey);
addAction(optionsAction);

QAction *backSoftKeyAction = new QAction(QString("Back"), this);
backSoftKeyAction->setSoftKeyRole(QAction::NegativeSoftKey);
connect(backSoftKeyAction, SIGNAL(triggered()),this, SLOT(close()));
addAction(backSoftKeyAction);
```

Basic Components

Label, Date Editor, Time Editor, Text Editor

Qt: QLabel, QDateEdit, QTimeEdit, QTextEdit

S60: CEikLabel, CEikDateEditor, CEikTimeEditor, CEikEdwin, CEikGlobalTextEditor

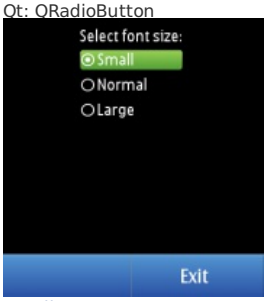
(C) Copyright Nokia 2013. All rights reserved.

QLabel, QDateEdit

Editors API

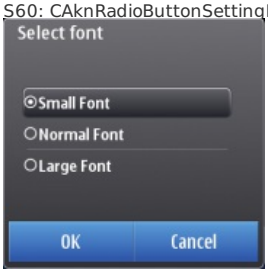
Radio Button

Qt: QRadioButton



QRadioButton

S60: CAknRadioButtonSettingPage

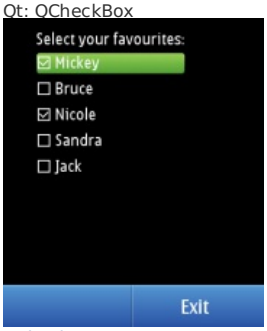


Setting Pages API

Symbian radio buttons can be used only in the setting page and are derived from CAknSettingPage. Qt QRadioButton can be used anywhere you want.

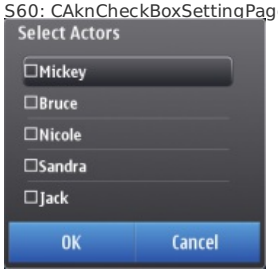
Check Box

Qt: QCheckBox



QCheckBox

S60: CAknCheckBoxSettingPage

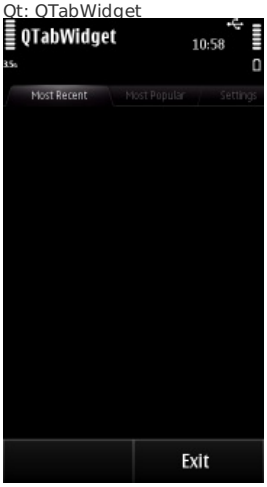


Setting Pages API

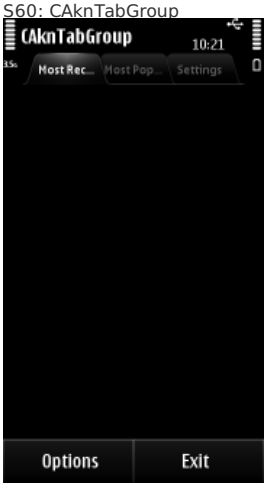
Symbian check boxes can be used only in the setting page and are derived from CAknSettingPage. Qt QCheckBox can be used anywhere you want.

Tabs

Qt: QTabWidget



S60: CAknTabGroup



(C) Copyright Nokia 2013. All rights reserved.

QTabWidget

Tabs API

Users can define whether or not the scroll buttons are visible in tabs using the `QTabWidget ::setUsesScrollButtons()` method.

In Qt applications, tabs reside in the content area instead of being embedded into the status pane, as in Symbian. If you want to maximise the application window area when using QTabWidget as the main window, define the application to run this window in full screen mode:

```
int main(int argc, char *argv[]){
    QApplication a(argc, argv);
    MyTabWidget tabWidget;
    tabWidget.showFullScreen();
    return a.exec();
}
```

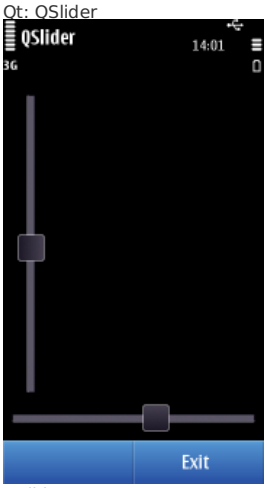
You may want to show CBA buttons also in full-screen mode. You can make them visible by setting the `Qt::WindowSoftkeysVisibleHint` flag into your window:

```
QWidget::setWindowFlags(windowFlags() | Qt::WindowSoftkeysVisibleHint);
```

Note that there are some usage problems when using tabs with a keyboard: when the focus is inside a tab, you cannot easily move to another tab but you first need to press 'Done' to move the focus to the tab row (unlike in Symbian, where pressing the left/right navigation key will always switch to another view).

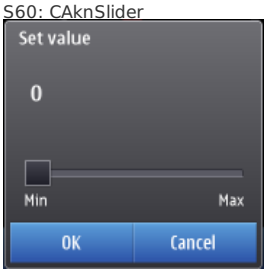
Slider

Qt: QSlider



QSlider

S60: CAknSlider

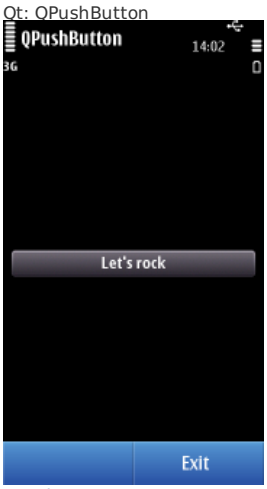


CAknSliderSettingItem

The Symbian slider can be used only in the setting page and is derived from CAknSettingPage. Qt QSlider can be used anywhere you want.

Button

Qt: QPushButton



QPushButton

S60: CAknButton

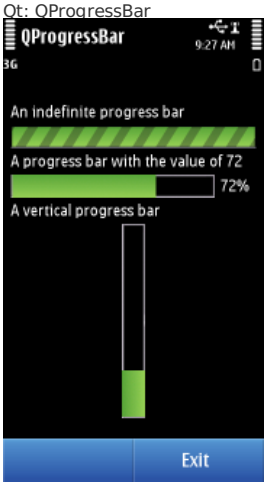
Generic button API

Note: The default size of QPushButton is currently too narrow for easy finger touch use. The height of the button can, however, be defined.

(C) Copyright Nokia 2013. All rights reserved.

Progress Indicator

Qt: QProgressBar



QProgressBar

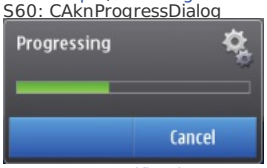
S60: CEikProgressInfo

Progress Bar Control Example, CEikProgressInfo

Qt: QProgressDialog

QProgressDialog

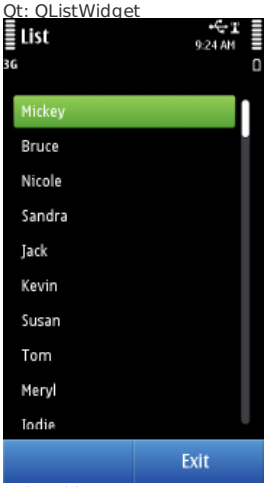
S60: CAknProgressDialog



Notes API Specification

List

Qt: QListWidget



QListWidget

S60: CAknSingleStyleListBox

Lists API

Symbian has many different UI classes for lists; Qt has only one, QListWidget. When using keyboard devices, there are some usage problems for selecting a row or moving between rows in the list in Qt.

Kinetic scrolling is not yet supported in Qt 4.6.3 in Symbian. It is possible to enable it by implementing the solution provided at labs.qt.nokia.com.

Enabling single tap

To use a Symbian^3 style single-tap interaction, listen to the `QListWidget:: itemClicked(QListWidgetItem*)` slot.

Enabling context menu

With single-tap interaction in use, item-specific commands cannot be used via the Options menu but rather by using a context menu that is launched with a long tap.

First, enable the context menu for the list:

```
listWidget->setContextMenuPolicy(Qt::CustomContextMenu);
```

Then start listening to a context menu signal from the list:

```
QObject::connect(listWidget, SIGNAL(customContextMenuRequested(const QPoint&)), this, SLOT(showContextMenu(const QPoint&)));
```

Now define the context menu:

```
m_contextMenu = new QMenu(this);
```


(C) Copyright Nokia 2013. All rights reserved.

```
QAction* aMenuItem1 = menuBar()->addAction("Some item ", this, SLOT(someSlot()));
m_contextMenu->addAction(aMenuItem1);
```

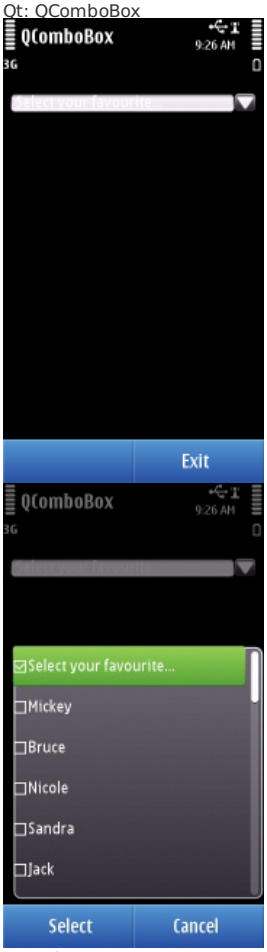
Finally, implement a slot for the `customContextMenuRequested()` signal to execute the context menu that has been defined:

```
void QMyWidget::showContextMenu(const QPoint& pos) {
m_contextMenu->exec(pos);
}
```

Qt 4.6.3 in Symbian has a bug: When a widget has defined a context menu, an 'Actions' menu command appears in the Options menu that launches the context menu.

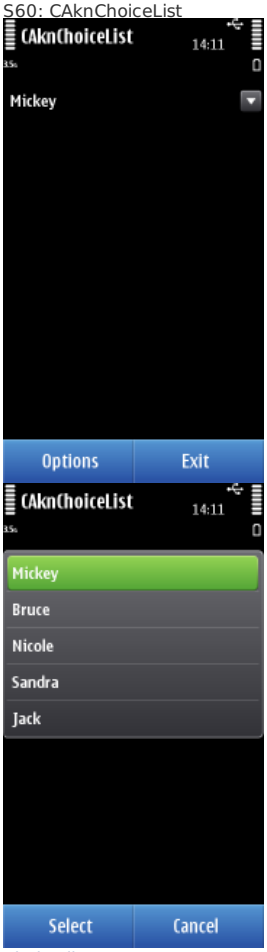
Combo Box

Qt: QComboBox



QComboBox

S60: CAknChoiceList

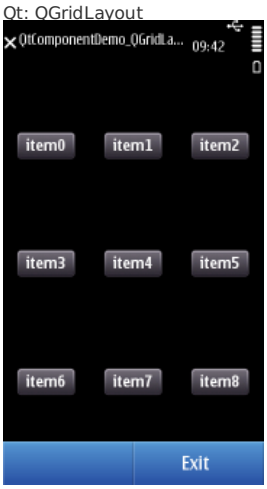


Choice list API

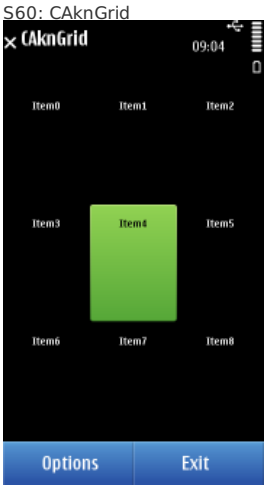
Qt 4.6.3 has invalid text colour in the QComboBox - it has white text on a white background. Open QComboBox has some kind of check box icons on each row in the list.

Grid

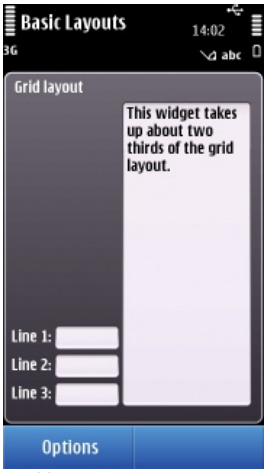
Qt: QGridLayout



S60: CAknGrid



(C) Copyright Nokia 2013. All rights reserved.

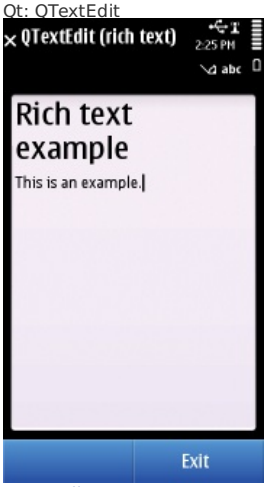


QGridLayout Grids API

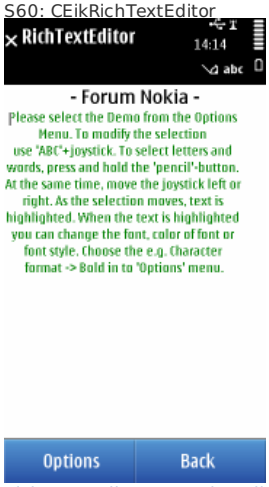
Qt QGridLayout is only a layout for other UI components, whereas CAknGrid is a more versatile grid component (on the other hand, CAknGrid is more complicated than QGridLayout).

Rich Text Editor

Qt: QTextEdit



S60: CEikRichTextEditor



QTextEdit Rich Text Editor Example, Editors API

QTextBrowser is derived from QTextEdit and can show HTML pages in Qt.

Toolbar

Qt: QToolBar

S60: CAknToolbar

QToolBar ToolBar API

Note that custom look and feel has been applied to the buttons in the example picture of the CAknToolbar.

By default QToolBar seems to be too big for mobile devices in Qt. Define a suitable size for the toolbar with this code:

```
QToolBar *tool = new QToolBar(this);
tool->setGeometry(0,0,200,20);
```

To learn more, see How to use QToolBar and QToolButton in Qt and Symbian C++ and Qt toolbar comparison.

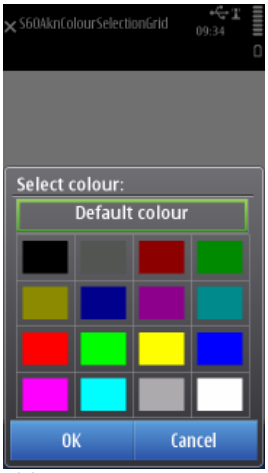
Dialogs

Color Dialog

Qt: QColorDialog S60: CAknColourSelectionGrid

(C) Copyright Nokia 2013. All rights reserved.

Application Windows and Dialogs, QColorDialog

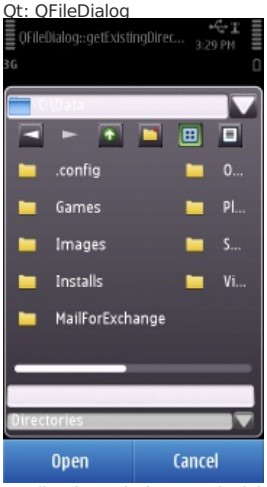


Dialogs API

File Dialog

List Directories

Qt: QFileDialog



Application Windows and Dialogs, QFileDialog
`QFileDialog::getExistingDirectory()`

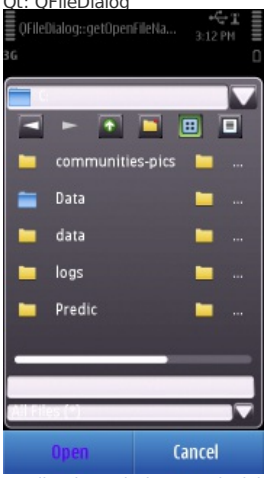
S60: AknCommonDialogsDynMem

Common File Dialogs API, AknCommonDialogsDynMem

AknCommonDialogsDynMem does not have directory browsing functionality.

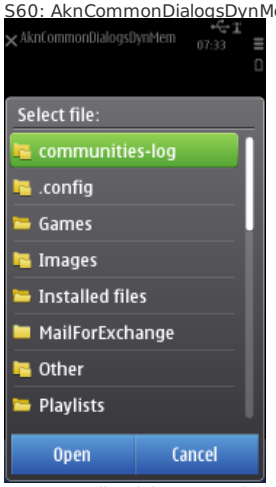
Open File

Qt: QFileDialog



Application Windows and Dialogs, QFileDialog
`QFileDialog::getOpenFileName()`

S60: AknCommonDialogsDynMem



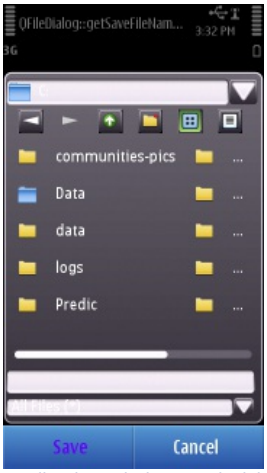
Common File Dialogs API, AknCommonDialogsDynMem

Save File

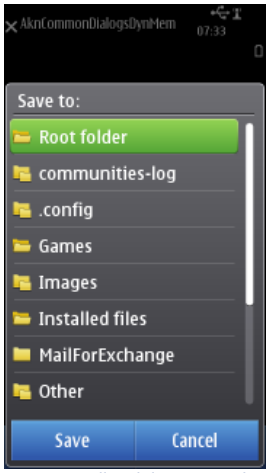
Qt: QFileDialog

S60: AknCommonDialogsDynMem

(C) Copyright Nokia 2013. All rights reserved.



Application Windows and Dialogs, QFileDialog
`QFileDialog::getSaveFileName()`



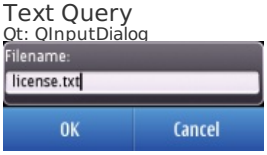
Common File Dialogs API, AknCommonDialogsDynMem

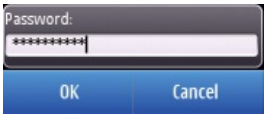
Define the destination path when saving a file. For example, the user's home directory is `QDir::home().path()`.

Input Dialog

Text Query

Qt: QInputDialog





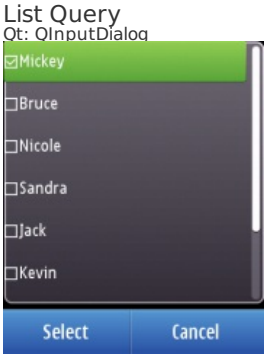
Application Windows and Dialogs, QInputDialog
`QInputDialog::getText()`

S60: CAknTextQueryDialog

Queries API

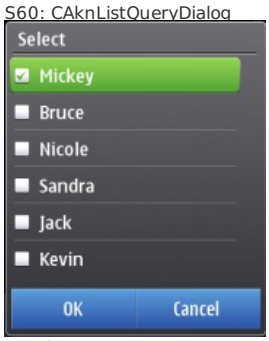
List Query

Qt: QInputDialog



Application Windows and Dialogs, QInputDialog
`QInputDialog::getItem()`

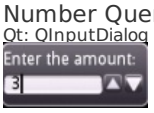
S60: CAknListQueryDialog



Queries API

Number Query

Qt: QInputDialog



Application Windows and Dialogs, QInputDialog
`QInputDialog::getInt()`

S60: CAknNumberQueryDialog

Queries API

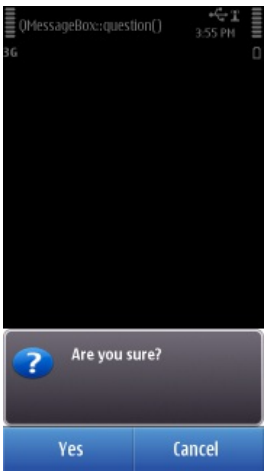
Message Dialog

Question Message

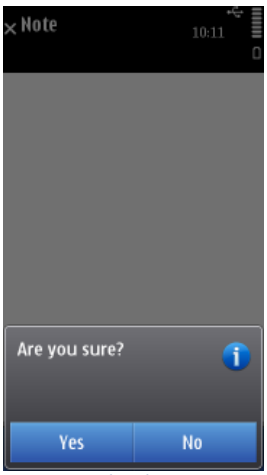
Qt: QMessageBox (QMessageBox::question())

S60: CAknQueryDialog (Confirmation Query)

(C) Copyright Nokia 2013. All rights reserved.



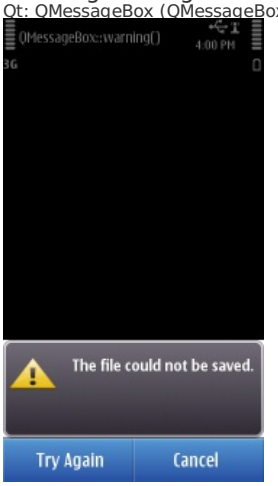
QMessageBox



Notes API, Showing Notes

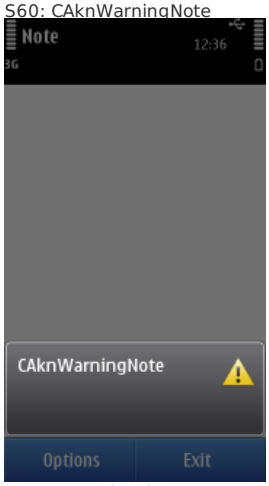
Warning Message

Qt: QMessageBox (QMessageBox::warning())



QMessageBox

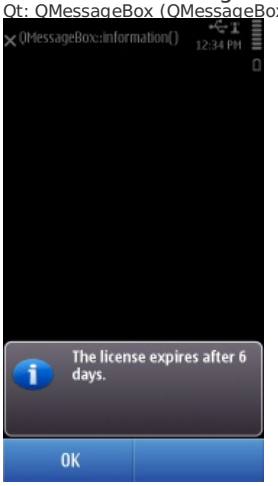
S60: CAknWarningNote



Notes API, Showing Notes

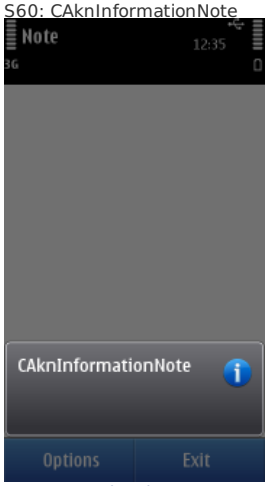
Information Message

Qt: QMessageBox (QMessageBox::information())



QMessageBox

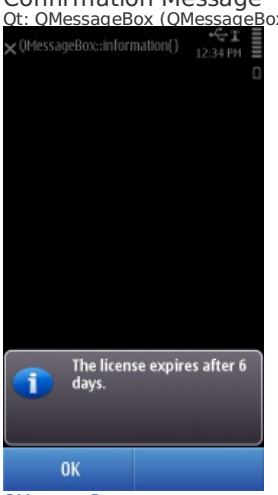
S60: CAknInformationNote



Notes API, Showing Notes

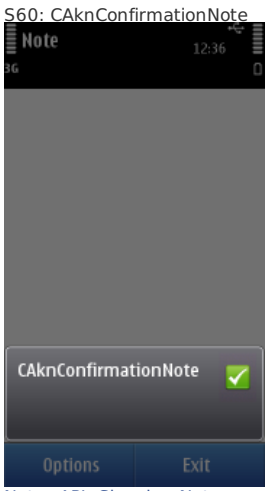
Confirmation Message

Qt: QMessageBox (QMessageBox::information())



QMessageBox

S60: CAknConfirmationNote



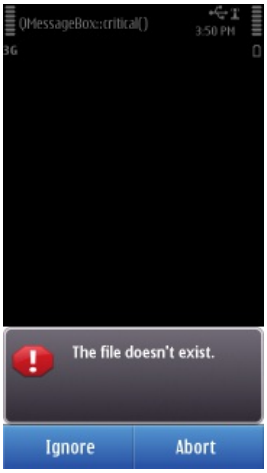
Notes API, Showing Notes

Error Message

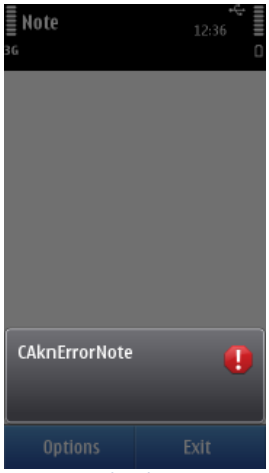
Qt: QMessageBox (QMessageBox::critical())

S60: CAknErrorNote

(C) Copyright Nokia 2013. All rights reserved.



QMessageBox



Notes API, Showing Notes

References

- Symbian^3 Classic UI Guide
- Classic UI Services for S60 5th Edition
- Qt widget gallery

Bold text

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.