Porting from Nokia Asha to Series 40

This article reflects on core considerations for porting MIDlets backwards from Nokia Asha software platform to the older Series 40 devices.

11 Aug 2013

Introduction

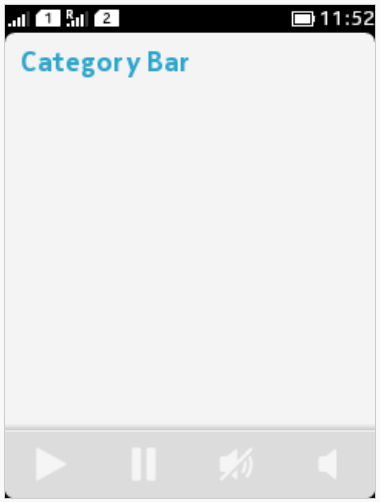
Nokia Asha software platform 1.0 and Series 40 full touch (Developer Platform 2.0) come both with full touch screen displays. Due to the similarities in the interaction mode (full touch only) and the supported APIs, porting from Nokia Asha to Series 40 full touch can be considered an easy task. However there are certain details that developers should be particularly aware of.

This article focuses on these details and further extends the porting scope to legacy Series 40 devices.

CategoryBar

The `CategoryBar` on Nokia Asha can be used for switching between tabbed views or launching specific actions within the same view. Series 40 full touch also support the `CategoryBar` component, but only within the context of switching between different tabbed views. MIDlets that use `CategoryBar` actions on Nokia Asha, i.e. the `ELEMENT_MODE_RELEASE_SELECTED` mode, should replace the `CategoryBar` with LCDUI Commands.

Note: There is no action to be taken for MIDlets that use `CategoryBar` tabs, i.e. the `ELEMENT_MODE_STAY_SELECTED` mode. This mode is implied by default, if not explicitly set. These apps can be directly ported to Series 40 full touch with no code modifications.



A four-element `CategoryBar` on Asha software platform 1.0 with actions

The four-element `CategoryBar` on Asha software platform 1.0 (with actions) shown above can be implemented with the following code:

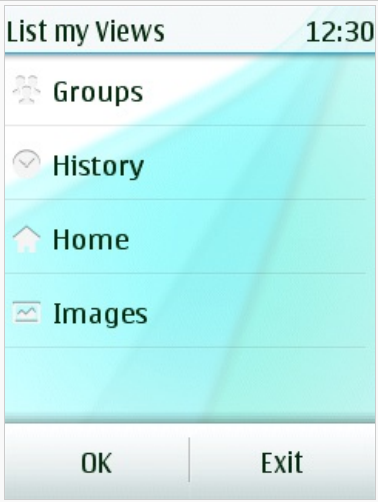
```
String[] labels = { "Play", "Pause", "Unmute", "Mute"};
bar = new CategoryBar(unselectedIcons, selectedIconsl, labels);
bar.setMode(CategoryBar.ELEMENT_MODE_RELEASE_SELECTED);

bar.setElementListener(this);
bar.setVisibility(true);
```

This would be ported to Series 40 full touch by removing the `CategoryBar` and by placing the actions on LCDUI Commands:

```
Command groupsCmd = new Command("Play", Command.OK, 1);
Command historyCmd = new Command("Pause", Command.OK, 2);
Command homeCmd = new Command("Unmute", Command.OK, 3);
Command imagesCmd = new Command("Mute", Command.OK, 4);
```


(C) Copyright Nokia 2014. All rights reserved.

On Series 40 Developer Platform 1.1 or earlier devices there is no support for the `CategoryBar`. Porting a `CategoryBar` MIDlet from Series 40 full touch to older Series 40 devices, however should not be done by utilizing Commands. Unlike Nokia Asha, the `CategoryBar` on Series 40 is used for switching between views, not for performing actions. We would therefore need to implement the different views as implicit list:



Implementing views as an implicit list

This is done using the following code:

```
list = new List("List my Views", List.IMPLICIT);
try {
 image = Image.createImage("/icons.png");
 groups = Image.createImage("/qgn_catbar_groups.png");
 history = Image.createImage("/qgn_catbar_history.png");
 home = Image.createImage("/qgn_catbar_home.png");
 images = Image.createImage("/qgn_catbar_images.png");
} catch (IOException e) {
 // TODO Auto-generated catch block
}

list.append("Groups", groups);
list.append("History", history);
list.append("Home", home);
list.append("Images", images);
```

Virtual Keyboard input

Another consideration is porting MIDlets that retrieve input from the `VirtualKeyboard`, provided that the application listens for key presses and interprets these to custom actions. This technique is useful for reading the keyboard typing and manually overriding its default behaviour. In a `CustomItem` or `Canvas` instance, the Virtual Keyboard can be launched with the following code:

```
CustomKeyboardControl vkbControl = VirtualKeyboard.getCustomKeyboardControl();
vkbControl.launch(
   VirtualKeyboard.VKB_TYPE_ITUT,
   VirtualKeyboard.VKB_MODE_ALPHA_LOWER_CASE);
```

and then the keyboard typing is caught within the `keyPressed()` method.

```
protected void keyPressed(int keyCode){
 // read manually the keyboard typing here
```


(C) Copyright Nokia 2014. All rights reserved.

```
}
```

However some of the returned key codes are different between Nokia Asha and Series 40. The table below shows the key codes returned for each virtual key from the Series 40 and Asha ITUT Virtual Keyboard:

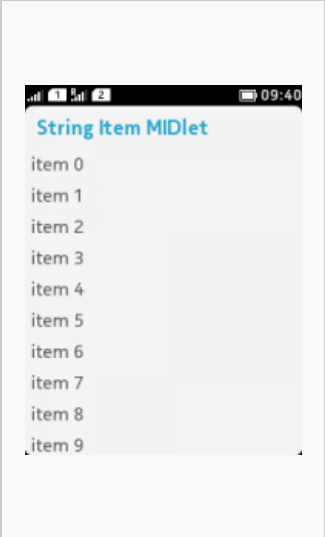
key	Series 40 full touch	Nokia Asha
.;	49	49
abc	50	50
def	51	51
ghi	52	52
jkl	53	53
mno	54	54
pqrs	55	55
tuv	56	56
wxyz	57	57
[space]	48	-10
[shift]	35	-15
[back space]	N/A	-8

StringItem rendering

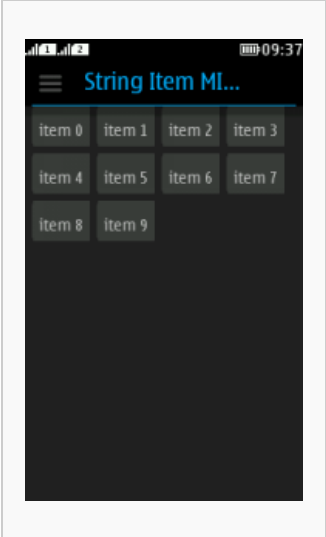
MIDlets that use lists of `StringItem` appended to a `Form` instance can generally be ported without any effort. `StringItem` with labels occupy the entire row in a Form. However the exception is when the label argument of the `StringItem` is the empty `String`. The rendering of the `StringItem` is then different on the screen of a Nokia Asha device compared to the rendering result on a Series 40 full touch device.

The following code places the `StringItem` on a new line on Nokia Asha, but appends the item on the same line on Series 40 full touch devices:

```
for(int i = 0; i < 10; i++) {
    StringItem item = new StringItem("", "item " + i);
    form.append(item);
}
```



String Items with empty label on Asha Software Platform



String Items with empty label on Series 40 full touch

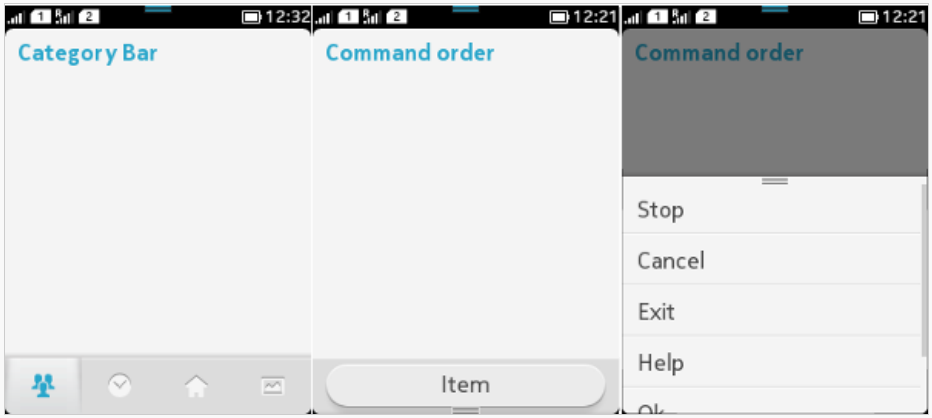
(C) Copyright Nokia 2014. All rights reserved.

For this reason, if we also want the `StringItem`s to be appended to separate lines on Series 40 full touch, we need to get the device type in runtime by using the `com.nokia.keyboard.type` system property and append a new line at the end of the string in the case of Series 40 full touch as shown below:

```
String keyboardType = System.getProperty("com.nokia.keyboard.type");
if (keyboardType.equals("OnekeyBack")){  // Nokia Asha
            for(int i = 0; i < 10; i++) {
                StringItem item = new StringItem("", "item " + i);
                form.append(item);
            }
}
else if (keyboardType.equals("None")) {  // Series 40 full touch
            for(int i = 0; i < 10; i++) {
                StringItem item = new StringItem("", "item " + i + "\n");
                form.append(item);
            }
}
```

Command mapping and IconCommands

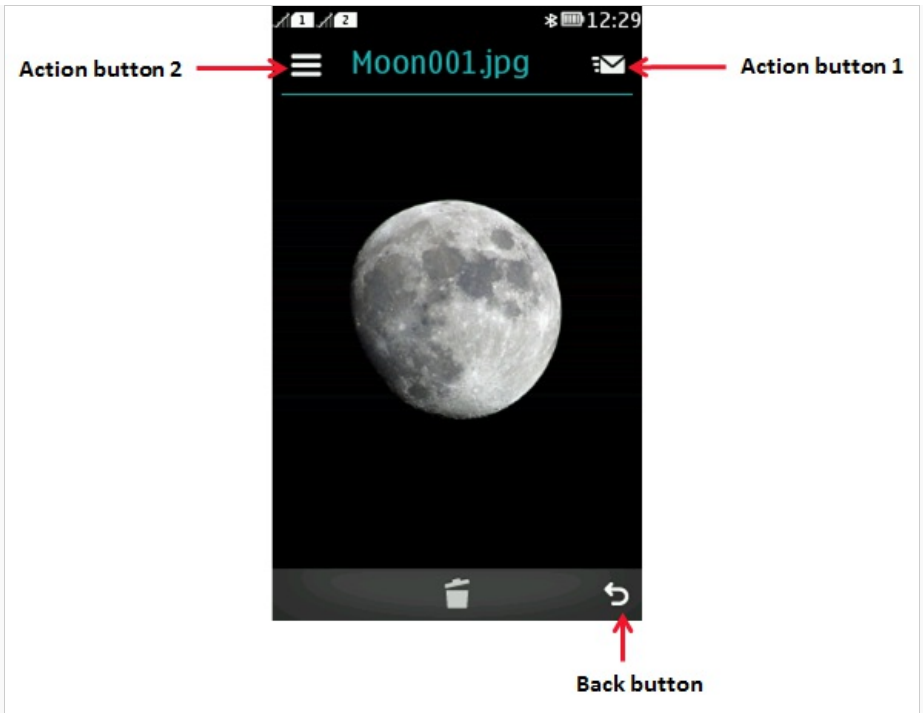
Please note that the Command mapping is different between Nokia Asha and Series 40 full touch. On Asha software platform 1.0, it's the priority of the Commands that decides the placement order. The highest priority command is placed in the *Toolbar* and the remaining Commands are placed in the *Options* menu:



The priority of the Commands defines the placement in Toolbar and Options Menu on Nokia Asha

On Series 40 full touch, it is the type of the Command that decides the placement of the Command in either Action Button 1, Action Button 2 (Options menu) or the Back button. The highest priority positive Command of type ITEM, HELP, OK or SCREEN is placed in Action Button 1, while the highest priority negative Command of type BACK or EXIT is placed in the Back button. Any additional commands, either positive or negative are placed in Action Button 2:

(C) Copyright Nokia 2014. All rights reserved.



The type of the Commands defines the placement on Series 40 full touch. For Commands of the same type, the priority is taken into account

Note: On Series 40 full touch, the Command's priority number is only used to distinguish between Commands of the same type

If we need to display the Toolbar Command on Action button 1, we need to ensure that

- it is a positive Command and
- it has the highest priority among the screen's positive Commands.

For displaying the Commands found in the Options menu on Action button 2, with the same order, we need to ensure that the Commands are of the same type. If the Command type is not the same, the Commands will be displayed on Action button 2 with STOP Commands placed first, then OK Commands, followed by CANCEL, ITEM, SCREEN, HELP and EXIT Commands in that order.

When porting from Nokia Asha to Series 40 full touch, the Toolbar Command can be replaced by an IconCommand for Action Button 1 in order to optimize the user experience. However, `IconCommand`s are not supported on Series 40 Developer Platform 1.1 or earlier devices. All `IconCommand`s should be replaced by standard LCDUI Commands when porting from Series 40 full touch to older platforms. The following code demonstrates how to switch from a Command to an `IconCommand` depending on the platform the application is running on:

```
if (keyboardType.equals("OnekeyBack")){  // Nokia Asha
 Command realIconOK = (Command) new Command("OK", Command.OK, 1);
 mainForm.addCommand(realIconOK);
}
else if (keyboardType.equals("None")) {  // Series 40 full touch
 try {
  // Image provided to represent selected icon
  img = Image.createImage(this.getClass().getResourceAsStream("icon_selected.png"));
  // Image provided to represent unselected icon
  unselected_img =
Image.createImage(this.getClass().getResourceAsStream("icon_unselected.png"));
 }
 catch(Exception e) {
  // Something went wrong
 }
 IconCommand realIconOK = (IconCommand) new IconCommand("img OK", unselected_img, img,
Command.OK, 1);
```


(C) Copyright Nokia 2014. All rights reserved.

```
  mainForm.addCommand(realIconOK);
 }
 else {  // touch and type or non touch
  Command realIconOK = (Command) new Command("OK", Command.OK, 1);
  mainForm.addCommand(realIconOK);
 }
```

Note: This code will only compile with Nokia SDK 2.0 for Java or later, however the binary package will run on older platforms

Capitalization In labels

Please note that different capitalization rules are recommended for list item labels or Commands depending on the target platform. On Nokia Asha, the global rule is to use a capital first letter followed by lower case letters. This rule applies to button labels, the Options menu, list items and Header bar. However on Series 40 full touch, the recommendation is to use ALL CAPS for your button labels while Options menu items, list items, and header labels should be all lower-case.

Launcher icon

The size of the launcher icon typically depends on the resolution of the device. The following table shows the size of the icon for Nokia Asha as well as the older Series 40 platforms:

Platform	icon size
Nokia Asha	50 x 50
Series 40 full touch	56 x 56
Series 40 240 x 320 displays	46 x 48
Series 40 208 x 208 displays	46 x 46
Series 40 128 x 160 displays	24 x 24
Series 40 128 x 128 displays	16 x 16

Besides the size, the icon needs to comply with certain design guidelines. For more information see

- the Iconography guidelines for Series 40 non touch
- the UI Style Guide for Series 40 touch and type
- the Launcher icon basics in the Series 40 Full Touch Design Guidelines
- the Iconography in Nokia Asha Design Guidelines

Note: Given that the launcher icon is specified in the application descriptor which is part of the installation package, there can be only one icon associated with each installation package. This means that it is not possible to choose the proper icon in runtime and that a separate build should be generated for each target platform.

Positioning graphics in Canvas applications

The screen size of Series 40 touch and type and Nokia Asha devices is 240 x 320 pixels and the aspect ratio is 3:4. The screen size of a Series 40 full touch device is 240 x 400 pixels and the aspect ratio is 3:5. If absolute values have been used for defining the dimensions of the UI components, they may not scale properly.

Even though the width remains the same, the height in canvas full screen mode changes from 320 pixels in Series 40 touch and type and Nokia Asha to 400 pixels in Series 40 full touch. When not in full screen mode, the height changes from 240 pixels in Series 40 touch and type and 258 pixels in Nokia Asha to 342 pixels in Series 40 full touch

As a result, if absolute values are used for drawing an object in the middle of the screen, for canvas applications, the object will not be properly centred. Also, some components might be hidden because the Y coordinate of their anchor point might fall outside the visible canvas due to the additional reduction on the height of the canvas due to the appearance of the virtual keyboard.

To avoid this, developers should place the UI components relative to the width and height of the screen and scale the UI accordingly instead of using the absolute values.

The following code draws two strings in the middle of the screen (in terms of X and Y axis) when not in full screen mode on Series

(C) Copyright Nokia 2014. All rights reserved.

40 full touch devices. One string uses absolute values for its coordinates, while the other uses relative values. The strings are on top of each other, when the Canvas is not in full screen mode. In full screen mode, the string with absolute coordinates is not vertically centred any more. The relative string is properly centred backwards on Series 40 touch and type and Series 40 non touch devices.

```
public void paint(Graphics g) {
        int width = getWidth();
        int height = getHeight();
        g.setColor(0xFFFFFF);
        g.fillRect(0, 0, width, height);
        g.setColor(0x000000);
        g.drawLine(0, height/2, width, height/2);
        g.drawLine(width/2, 0, width/2, height);
        g.setColor(0xFF0000);
        g.drawString("Absolute String", 120, 161, Graphics.TOP | Graphics.HCENTER); //
NOT LIKE THIS!
        g.setColor(0x00FF00);
        g.drawString("Relative String", getWidth() / 2, (getHeight() / 2) -
(font.getHeight() / 2), Graphics.TOP | Graphics.HCENTER);
}
```

Retrieving Location

While Nokia Asha and Series 40 full touch devices support Cell ID location retrieval, Series 40 non touch and a few Series 40 touch and type devices, do not support this type of location retrieval. Even though Cell ID location retrieval is not supported on earlier Series 40 devices, some of these devices come with integrated GPS module that can be used either as standalone or in the assisted mode (A-GPS). The table below shows the location retrieval methods supported by Nokia Asha and Series 40:

Platform	Screen type	Location retrieval methods
Nokia Asha software platform 1.0	full touch	Cell ID
Developer Platform 2.0	full touch	Cell ID
Developer Platform 1.1	touch and type	Cell ID
Developer Platform 1.0	touch and type	Cell ID
Series 40 6th Edition Feature Pack 1	touch and type	N/A
Series 40 6th Edition	non touch	GPS
Series 40 5th Edition Feature Pack 1 or earlier	non touch	N/A

The `LocationUtil` class that enables Cell ID location retrieval is not supported on Series 40 6th Edition Feature Pack 1 or Series 40 6th Edition devices. In order to retrieve the `LocationProvider` instance on these platforms we would need to replace this code

```
//Specify the retrieval method to Online/Cell-ID
int[] methods = {(Location.MTA_ASSISTED | Location.MTE_CELLID | Location.MTE_SHORTRANGE
| Location.MTY_NETWORKBASED)};
// Retrieve the location provider
provider = LocationUtil.getLocationProvider(methods, null);
location = provider.getLocation(50000);
coordinates = location.getQualifiedCoordinates();
form.append("Latitude:" + coordinates.getLatitude()+ "\n");
form.append("Longitude:" + coordinates.getLongitude());
```

with code that utilizes the GPS module. The closest equivalent to the Cell ID method, is an Assistant GPS (A-GPS) location retrieval, where the device will use the data connection (Internet) to contact an assistance server, by allowing costs to occur, i.e. by calling the method `setCostAllowed(true)`. This is a sample snippet on how the `QualifiedCoordinates` instance can be

(C) Copyright Nokia 2014. All rights reserved.

retrieved on Series 40 6th Edition:

```
public class LocationMIDlet
  extends MIDlet
  implements LocationListener {
  …
  Criteria criteria = new Criteria();
  criteria.setSpeedAndCourseRequired(false)
  criteria.setCostAllowed(true)
  criteria.setPreferredPowerConsumption(Criteria.POWER_USAGE_LOW)
  LocationProvider lp = LocationProvider.getInstance(criteria);
  lp.setLocationListener(this, -1, -1, -1);
  …
  public void locationUpdated(LocationProvider provider, Location location) {
    if(location.isValid()) {
      QualifiedCoordinates c = location.getQualifiedCoordinates();
    }
  }
}
```


(C) Copyright Nokia 2014. All rights reserved.