Samsung Native Text Input

Version 0.9, Draft

INFORMATION GUIDE

COPYRIGHT

Trademarks and Service Marks

About This Document

This document contains the overview of Native Text Input (NTI) and procedure for invoking the NTI keypad, followed by a sample code snippet to use the NTI feature.

Scope:

This document is intended for Java ME developers wishing to develop Java ME applications for Samsung touch based phones. It assumes good knowledge of java programming language and hands-on experience with MIDlet development environment.
This document focuses on Samsung Native Text Input and therefore explaining the Java ME technology is out of the scope of this document.

Document History:

Date	Version	Comment
04/02/09	0.9	Draft

Abbreviations:

Java ME	Java Micro Edition
MIDP	Mobile Information Device Profile
NTI	Native Text Input
GTB	Global Toolbar
JAD	Java Application Descriptor
UI	User Interface

Table of Contents

Table of Figures

Introduction

Samsung Native Text Input (NTI) is mostly available with touch-based Samsung devices. NTI provides larger virtual keypad for user inputs navigation. Non Touch based phones contain normal keypad, which allows user for navigation and text inputs (mostly hardware based). In order to facilitate similar user experience touch based phones in Samsung are loaded with NTI. This document explains the user inputs and navigations from Java ME prospective on Samsung handsets. This document does not explain the non touch based phones navigation and text inputs.

Overview

There are three modes of navigation and text inputs for Textbox.

1. Normal Textbox having Global Toolbar (GTB) navigation.

Figure 1: Textbox with GTB

2.After clicking the key shown as “123" of Global Toolbar (GTB) will lead to following screen known as Textbox with Java keypad. Figure 2 is referred as Java Key Pad.

Figure 2: Textbox with Java Key Pad

3. After clicking the textbox directly, virtual keypad will appear as shown in Figure 3. This virtual keypad is called as Samsung Native Text Input.

Figure 3: Text Box with Virtual Keypad (NTI)

Native Text Input using Textfield and Textbox

Textfield shown below in the figure 4 (Numeric GTB Textfield) is an item in the form. Textbox is a place where you enter text, digits etc. It is displayable object. *Textbox* can be invoked as:

```
{
  …
 Display display=Display.getDisplay(this);
 TextBox textBox = new TextBox(title, "", 50, TextField.ANY);
 Textfield can be invoked as:

 TextField textfield= new TextField(title, "", 50, TextField.ANY);
 Form form=new Form("TextField");
 display.setCurrent(textbox/form);
 …
}
```

In touch-based devices, there are two ways to enter the characters:

1. Focus on the textfield / textbox and press the GTB that is present in the lower portion of the screen. GTB is visible for all the MIDlets by default and the same can be removed by adding *MIDlet-TouchSupport* attribute in JAD with value *true*.

```
MIDlet-TouchSupport: true
```

2. Click the textfield / textbox directly in the form. Samsung Native Text Input is invoked. It has good UI and with bigger keys for better navigation and user input.

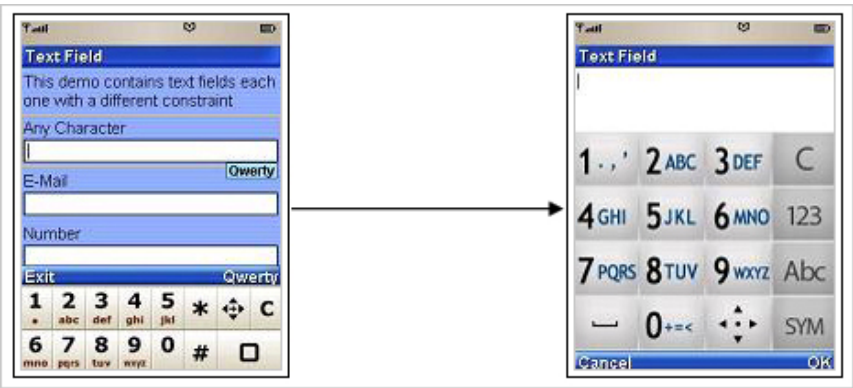

Figure 4: Numeric GTB Textfield to Virtual Keypad (NTI)

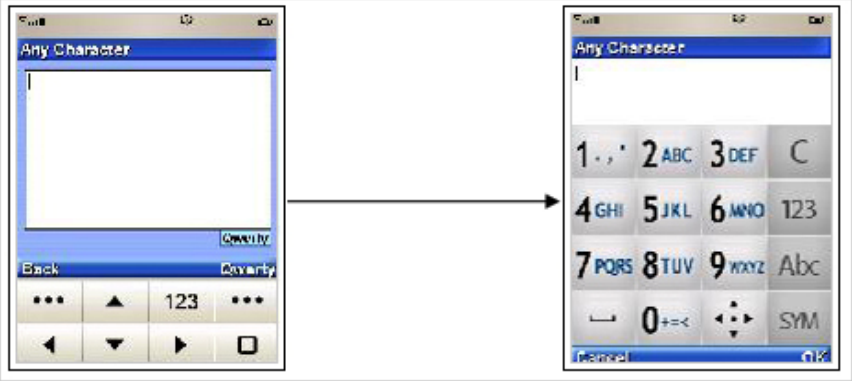

Figure 5: Normal GTB Textbox to Virtual Keypad (NTI)

For displaying Native Text Input

Optimization is applicable for textbox and not for textfield. When a MIDlet developer displays the textbox as shown below:

```
{
  …
  Display display=Display.getDisplay(this);
  TextBox textBox = new TextBox(title, "", 50, TextField.ANY);
  display.setCurrent(textbox);
  …
}
```

It shall display Textbox with blue screen i.e. Normal Textbox with GTB. If user clicks on this textbox, Samsung Native Text Input (NTI) appears. NTI has rich UI and bigger keys for navigation and user inputs.

Native Text Input before optimization is as follows:

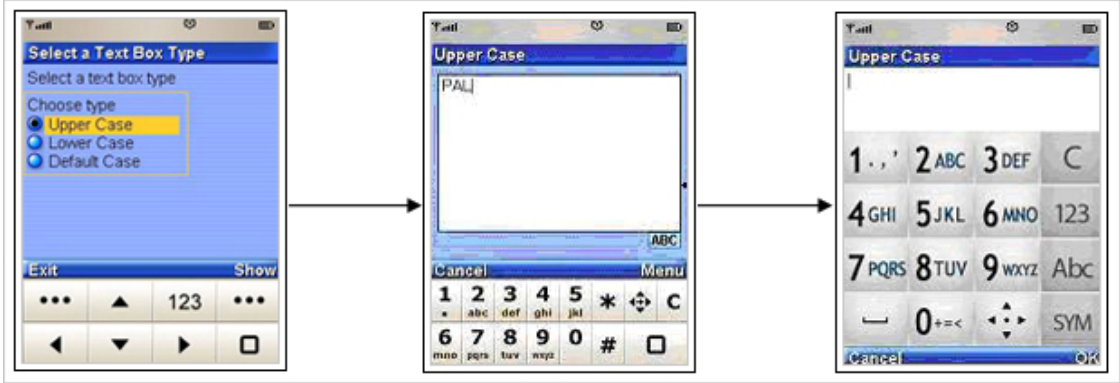

Figure 6: Invoking NTI without optimization

By optimizing this process, Samsung Native Text Input can be displayed after invoking textbox. It leaves the Normal GTB Textbox (which is middle in the picture. See Figure 6) so that users directly jump to Samsung Native Text Input mode. See Figure 7 that shows the NTI after optimization.

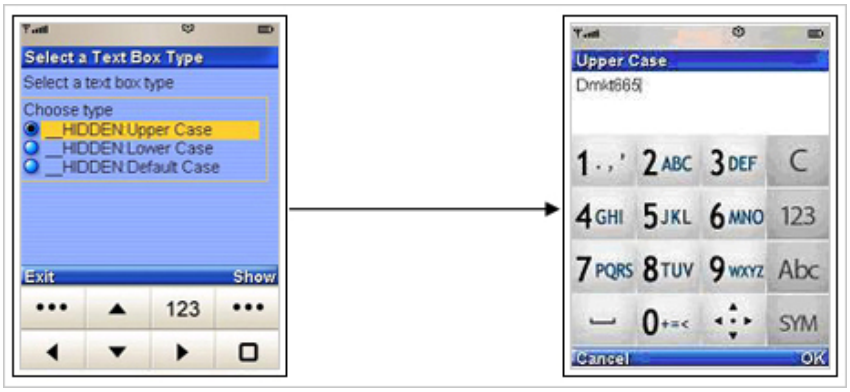

Figure 7: Invoking NTI with Optimization

Steps for Optimization:

Title of the textbox should start with *"__HIDDEN"*.

```
{
    . . . .  TextBox textBox = new TextBox("__HIDDEN:Title", "", 50, TextField.ANY); . . . .
}
```

When MIDlet developer invokes the *Textbox* object the Samsung implementation will ensure of removing the *"__HIDDEN"* string from the title and will leave the Normal GTB textbox to display directly Samsung Native Text Input.

```
{
  …
```

```
    display.setCurrent(textbox);
    …
}
```

There should be only two commands in the textbox. One should be *Command.OK* and another *Command.CANCEL*

```
{
    …
    textBox.addCommand(CMD_OK);
    textBox.addCommand(CMD_CANCEL);
    …
}
```

Procedure for displaying Samsung Native Text Input

There are two ways in which NTI can be displayed:

- NTI with optimization
- NTI without optimization

Native Text Input with optimization

Create *Form* class object.

```
Form mainForm = new Form ("Select a Text Box Type");
```

Then create a Choicegroup object and append it in the Form class.

```
static final string [] textBoxLabels = { "__HIDDEN:Upper Case", "__HIDDEN:Lower
    Case","__HIDDEN:Default Case" };
Image[] imageArray = null;
ChoiceGroup types = new ChoiceGroup("Choose type", Choice.EXCLUSIVE,
    textBoxLabels, imageArray);
mainForm.append(types);
```

Add commands to *Form* class as follows:

```
mainForm.addCommand(CMD_SHOW);
mainForm.addCommand(CMD_EXIT);
```

Now invoke the *Form*

```
mainForm.setCurrent(mainForm);
```

See Figure 8. The form is displayed on the screen.

Figure 8: Initial Form Screen

Now select the Show [Command]. Create a textbox object in the *commandAction* method when Show [Command] is called to display the textbox.

```
TextBox textBox = new TextBox(title, "", 50, TextField.ANY);
```

Where title String should start with *__Hidden* parameter that helps Samsung implementation to detect and display Samsung Native keypad thereby leaving numeric GTB textbox.

In the textbox, add two commands named as OK [Command] and CANCEL [Command] to display Samsung native keypad. Samsung Implementation will check whether there are two commands added to the textbox and that are OK [Command] and CANCEL [Command] and displays Samsung Native Keypad.

```
textBox.addCommand(CMD_OK);
textBox.addCommand(CMD_CANCEL);
```

Now when you call this method,

```
display.setCurrent(textBox);
```

Samsung Native Keypad screen appears. See Figure 9.

Figure 9: Virtual Keypad (NTI)

Click OK soft key to return to main form class as shown below:

```
display.setCurrent(mainForm);
```

Native Text Input without optimization

Create *Form* class object.

```
Form mainForm = new Form ("Select a Text Box Type");
```

Then create a ChoiceGroup object and append it in the Form class.

```
static final string [] textBoxLabels = { "Upper Case", " Lower Case",
  " Default Case" };
Image[] imageArray = null;
ChoiceGroup types = new ChoiceGroup("Choose type", Choice.EXCLUSIVE,
   textBoxLabels, imageArray);
mainForm.append(types);
```

Ad commands to *Form* class as follows:

```
mainForm.addCommand(CMD_SHOW);
mainForm.addCommand(CMD_EXIT);
```

Then invoke Form

```
mainForm.setCurrent(mainForm);
```

Figure 10 shows the form displayed on the screen.

Figure 10: Initial Form Screen

Now select the Show [Command].

Show [Command] is used for displaying the textbox.

```
TextBox textBox = new TextBox(title, "", 50, TextField.ANY);
```

The title of the string is not started with *__Hidden*.

Add commands to the Textbox.

```
textBox.addCommand(CMD_OK);
textBox.addCommand(CMD_CANCEL);
```

Now *Textbox* is invoked,

```
display.setCurrent(textBox);
```

A numeric GTB textbox will be displayed. See Figure 11.

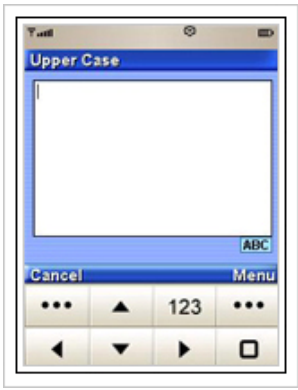

Figure 11: Normal GTB TextBox

On touching the textbox directly, Samsung Native Keypad screen appears. See Figure 12. There are two commands OK [Command] and CANCEL [Command] at the bottom of the screen.

Figure 12: Virtual Keypad (NTI)

Clicking on OK [Command] or Cancel [Command] soft key shall leads to textbox with GTB.

Sample Code Example

The Sample example given below shows how Samsung Native Text Input is displayed by avoiding the Numeric GTB textbox.

Class: TextBoxDemo.java

```
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Displayable;
import javax.microedition.lcdui.Form;
import javax.microedition.lcdui.Image;
import javax.microedition.lcdui.ChoiceGroup;
import javax.microedition.lcdui.Choice;
import javax.microedition.lcdui.TextBox;
import javax.microedition.lcdui.TextField;
import javax.microedition.midlet.MIDlet;

/* Midlet class which shows native text input*/
public class TextBoxDemo extends MIDlet implements CommandListener {

   /*this command is used to exit from the current application*/
   private static final Command CMD_EXIT = new Command("Exit", Command.EXIT,
    1);
   //*this command is used to move to previous screen*/
```

```
private static final Command CMD_BACK = new Command("Back",
  Command.BACK, 1);
/*this command is used to go to next screen*/
private static final Command CMD_SHOW = new Command("Show",
  Command.SCREEN, 1);
/*this command is used for getting acceptence*/
private static final Command CMD_OK = new Command("OK", Command.OK, 1);
/*this command is used to reject the current screen action*/
private static final Command CMD_CANCEL = new Command("Cancel",
  Command.CANCEL, 1);
/*title of textbox stored in an String Array which starts with "__HIDDEN" String*/
private static final String[] textBoxLabels
   = {"__HIDDEN:Upper Case", "__HIDDEN:Lower Case", "__HIDDEN:Default Case"};
private Display display;
private ChoiceGroup types;
private Form mainForm;
/**
 * Constructor
 */
public TextBoxDemo() {
    display = Display.getDisplay(this);
    mainForm = new Form("Select a Text Box Type");
    mainForm.append("Select a text box type");
    Image[] imageArray = null;
    types = new ChoiceGroup("Choose type", Choice.EXCLUSIVE, textBoxLabels,
     imageArray);
    mainForm.append(types);
    mainForm.addCommand(CMD_SHOW);
    mainForm.addCommand(CMD_EXIT);
    mainForm.setCommandListener(this);
}

public void startApp() {
    display.setCurrent(mainForm);
}

public void pauseApp() {
    /*all interupts can be handled here*/
}

public void destroyApp(boolean unconditional) {
}

public void commandAction(Command c, Displayable d) {
    if (c == CMD_EXIT) {
        destroyApp(false);
        notifyDestroyed();
```

```
    } else if (c == CMD_SHOW) {
      int index = types.getSelectedIndex();
      /*The selected case type will be taken as title for textbox*/
      String title = textBoxLabels[index];
      /*for diplaying textbox UI whose title starts with __HIDDEN*/
      TextBox textBox = new TextBox(title, "", 50, TextField.ANY);
      if (index == 0) {
        textBox.setInitialInputMode("MIDP_UPPERCASE_LATIN");
      } else if (index == 1) {
        textBox.setInitialInputMode("MIDP_LOWERCASE_LATIN");
      }
      /*For bypassing the Java TextBox 2 commands are added CMD_OK and
        CMD_CANCEL*/
      textBox.addCommand(CMD_OK);
      textBox.addCommand(CMD_CANCEL);
      textBox.setCommandListener(this);
      display.setCurrent(textBox);
    } else if (c == CMD_BACK || c == CMD_OK || c == CMD_CANCEL) {
      display.setCurrent(mainForm);
    }
  }
}
```