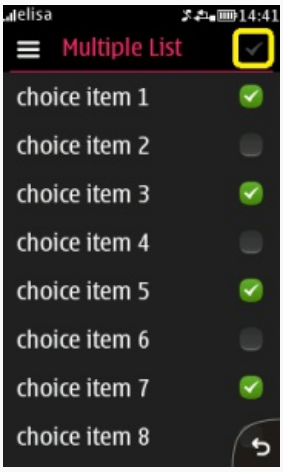
How to enable Action Button 1 when using Lists on Series 40 Full Touch

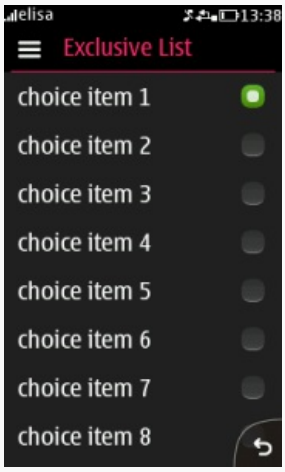
This code example shows how you can implement lists with a Command on Action Button 1. This is useful for creating lists which can be accepted and closed using a single click on Series 40 Full Touch devices.

Overview

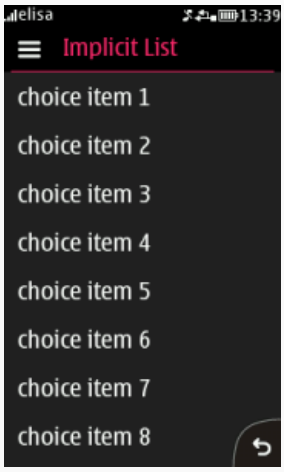
On Series 40 Full Touch devices (Java Runtime 2.0.0), LCDUI Lists do not support support Command Actions on Action Button 1. As shown in the screenshots below, lists of type *Multiple* are displayed with a dimmed Action Button 1 (shown circled in yellow) that cannot be removed or assigned to a `Command`, while lists of type *Exclusive* and *Implicit* are displayed with no Action Button 1 at all.



List instance of Multiple type: Action Button 1 is dimmed



List instance of Exclusive type: Action Button 1 is disabled



List instance of Implicit type: Action Button 1 is disabled

This means that it is not possible to implement an LCDUI list (other than type implicit) that is closed and acknowledged in a single click (actions assigned to the list *are* available under Action Button 2 (the menu), which means that they require two taps to invoke).

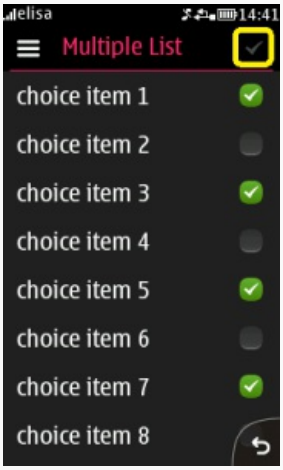
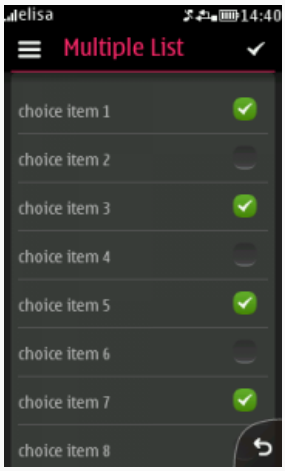
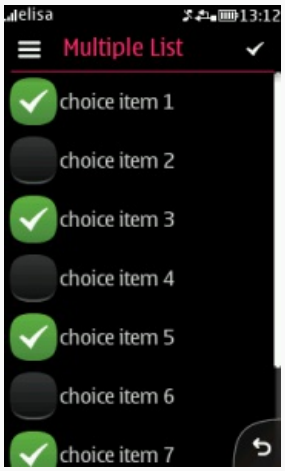
The following sections explain how you can implement the three list types with a command on Action Button 1, using either Form or LWUIT.

Multiple-type lists

The table below provides screenshots (and source code) for "Multiple" lists, implemented using LCDUI, a `Form` and a `ChoiceList`, and LWUIT. Note that the LCDUI and LWUIT implementations have a functional Command on Action Button 1.

	Original Code with a List	Solution Code with a Form	Solution with LWUIT
Source	Media:ListMultipleMIDletSource.zip	Media:ListMultipleSolutionSource.zip	Media:ListMultipleLWUITSource.zip
Binaries	Media:ListMultipleMIDletBinaries.zip	Media:ListMultipleSolutionBinaries.zip	Media:ListMultipleLWUITBinaries.zip

(C) Copyright Nokia 2014. All rights reserved.

Final Result	 LCDUI List: Dimmed Action Button 1	 LCDUI Form with ChoiceGroup: Action Button 1 is fully functional	 LWUIT Form for Series 40 with CheckBox: Action Button is enabled

The original code is a simple List instance of type `MULTIPLE` with several items appended and two actions, an **Exit** and **OK** Command, added:

```
Command exitCmd = new Command("Exit", Command.EXIT,0);
Command okCmd = new Command("OK",Command.OK,0);

protected void startApp() throws MIDletStateChangeException {
 list = new List("Multiple List", List.MULTIPLE);
 list.append("choice item 1", null);
 list.append("choice item 2", null);
 list.append("choice item 3", null);
 list.append("choice item 4", null);
 list.append("choice item 5", null);
 list.append("choice item 6", null);
 list.append("choice item 7", null);
 list.append("choice item 8", null);
 list.append("choice item 9", null);

 list.addCommand(exitCmd);
 list.addCommand(okCmd);
 list.setCommandListener(this);

 Display.getDisplay(this).setCurrent(list);
}

public void commandAction(Command c, Displayable d) {
 if(c == okCmd) {
  System.out.println("Some task for Action Button 1");
 }

 if(c == exitCmd) {
  notifyDestroyed();
 }
}
```

 (C) Copyright Nokia 2014. All rights reserved.

We can achieve a similar result by using a combination of a Form and a `ChoiceGroup.MULTIPLE` component as follows:

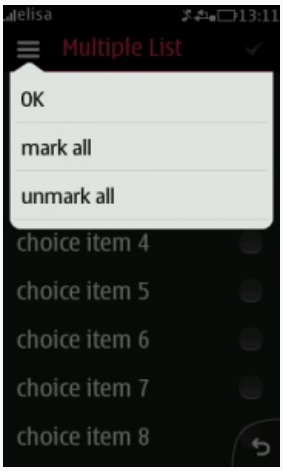
```
Command okCommand = new Command("OK", Command.OK, 0);
Command markAll = new Command("Mark All", Command.ITEM, 0);
Command unmarkAll = new Command("Unmark All", Command.ITEM, 1);
Command exitCommand = new Command("Exit", Command.EXIT, 0);

protected void startApp() throws MIDletStateChangeException {
 mainScreen = new Form("Multiple List");

 original = new ChoiceGroup(null, ChoiceGroup.MULTIPLE);
 original.append("choice item 1", null);
 original.append("choice item 2", null);
 original.append("choice item 3", null);
 original.append("choice item 4", null);
 original.append("choice item 5", null);
 original.append("choice item 6", null);
 original.append("choice item 7", null);
 original.append("choice item 8", null);
 original.append("choice item 9", null);

 mainScreen.append(original);
        mainScreen.addCommand(okCommand);
 mainScreen.addCommand(markAll);
 mainScreen.addCommand(unmarkAll);
 mainScreen.addCommand(exitCommand);
 mainScreen.setCommandListener(this);
 Display.getDisplay(this).setCurrent(mainScreen);
```

Similarly, we can use the `CheckBox` component in LWUIT for Series 40, with a `Command` of type `OK`:

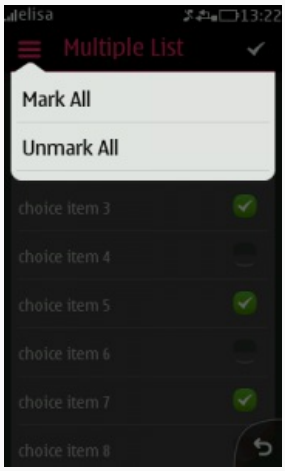
```
protected void startApp() throws MIDletStateChangeException {

        Display.init(this);

        form = new Form("Multiple List");
        form.addCommand(exitCommand);
        form.setBackCommand(exitCommand);

        form.addCommand(markAll);
        form.addCommand(unmarkAll);
        form.addCommand(okCommand);
        form.addCommandListener(this);

        form.setLayout(new BoxLayout(BoxLayout.Y_AXIS));

        checkBox = new CheckBox[9];
        for(int i = 0; i < checkBox.length; i++) {
         checkBox[i] = new CheckBox("choice item " + (i + 1));
         form.addComponent(checkBox[i]);
        }

        form.show();

}
```

 (C) Copyright Nokia 2014. All rights reserved.

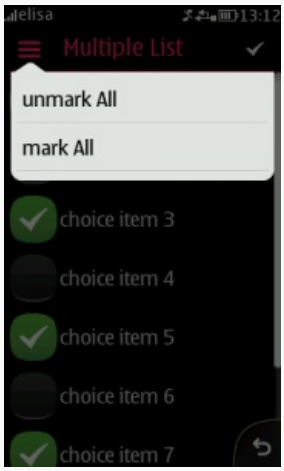
Manually adding the Mark All and Unmark All functionality



Mark and unmark options are added automatically in List instances but the OK Command is added under Action Button 2



Mark and unmark options are added manually in Form instances. The OK Command remains in Action Button 1



Mark and unmark options are added manually in LWUIT for Series 40. The OK Command remains in Action Button 1

Note: When porting from an LCDUI List to a LCDUI Form or LWUIT Form, we need to manually add the mark and unmark Commands, in order to maintain the same functionality, because `ChoiceGroup`s and `CheckBox`es do not provide these options by default.

`ChoiceGroup`s receive an array of boolean values as flags in order to set each choice item to the selected or unselected state. By passing a flag boolean array where all values are set to true or false to the `ChoiceGroup` we can achieve a similar mark all and unmark all functionality respectively:

```
public void commandAction(Command c, Displayable d) {
 if(c == okCommand) {
  System.out.println("Action Button 1 selected!");
 }

 if(c == markAll) {
  boolean[] flags = new boolean[original.size()];
  for (int i = 0; i < flags.length; i++) {
   flags[i] = true;
  }
  original.setSelectedFlags(flags);
 }

 if(c == unmarkAll) {
  boolean[] flags = new boolean[original.size()];
  for (int i = 0; i < flags.length; i++) {
   flags[i] = false;
  }
  original.setSelectedFlags(flags);
 }

 if(c == exitCommand) {
  notifyDestroyed();
 }
```

 (C) Copyright Nokia 2014. All rights reserved.

```
}
```

Similarly in LWUIT for Series 40, we need to iterate through the list of `CheckBoxes` and call the `setSelected()` method for each one of them:

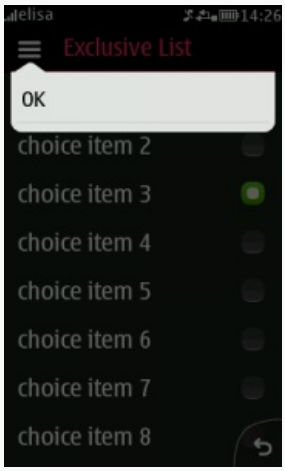
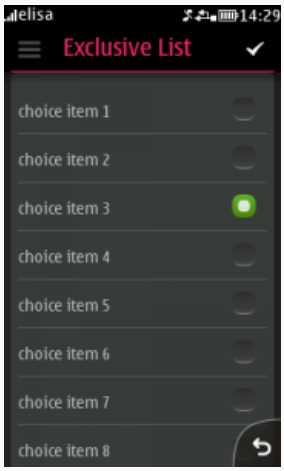
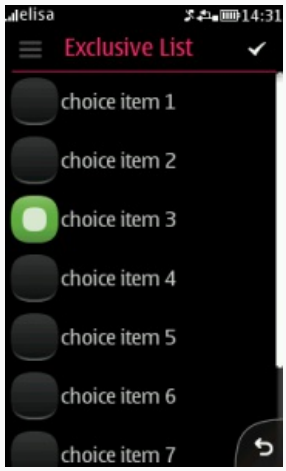
```
public void actionPerformed(ActionEvent event) {

       if(event.getCommand() == okCommand) {
        System.out.println("Action Button 1 selected!");
       }

       if(event.getCommand() == exitCommand) {
        notifyDestroyed();
       }

       if(event.getCommand() == markAll) {
        for(int i = 0; i < checkBox.length; i++){
         checkBox[i].setSelected(true);
        }
       }

       if(event.getCommand() == unmarkAll) {
        for(int i = 0; i < checkBox.length; i++){
         checkBox[i].setSelected(false);
        }
       }

}
```

Exclusive lists

The table below provides screenshots (and source code) for exclusive lists, implemented using LCDUI, a `Form` and a `ChoiceList`, and LWUIT. Note that the LCDUI and LWUIT implementations have a functional Command on Action Button 1.

	Original Code with a List	Solution Code with a Form	Solution with LWUIT
Source	Media:ListExclusiveMIDletSource.zip	Media:ListExclusiveSolutionSource.zip	Media:ListExclusiveLWUITSource.zip
Binaries	Media:ListExclusiveMIDletBinaries.zip	Media:ListExclusiveSolutionBinaries.zip	Media:ListExclusiveLWUITBinaries.zip

 (C) Copyright Nokia 2014. All rights reserved.

Final Result	 LCDUI List: OK Command under Action Button 2. Action Button 1 is disabled	 LCDUI Form with ChoiceGroup: Action Button 1 is fully functional	 LWUIT Form for Series 40 with RadioButtons: Action Button is enabled

The problem is that LCDUI exclusive lists force the Command of type OK under Action Button 2. Action Button 1 is entirely disabled in this case:

```
List list;
Command exitCmd = new Command("Exit", Command.EXIT,0);
Command okCmd = new Command("OK",Command.OK,0);

protected void startApp() throws MIDletStateChangeException {
 list = new List("Exclusive List", List.EXCLUSIVE);
 for(int i = 0; i < 9; i++){
  list.append("choice item " + (i + 1), null);
 }
 list.addCommand(exitCmd);
 list.addCommand(okCmd);
 list.setCommandListener(this);
 Display.getDisplay(this).setCurrent(list);
}

public void commandAction(Command c, Displayable d) {
 if(c == okCmd) {
  System.out.println("Some task for Action Button 1");
 }

 if(c == exitCmd) {
  notifyDestroyed();
 }
}
```

We can overcome this issue, if we replace the List instance with a `Form` instance and a `ChoiceGroup` of type `EXCLUSIVE`, similarly to what we did in the case of Multiple Lists:

```
Form mainScreen;
ChoiceGroup original;
Command okCommand = new Command("OK", Command.OK, 0);
Command exitCommand = new Command("Exit", Command.EXIT, 0);
```

 (C) Copyright Nokia 2014. All rights reserved.

```
protected void startApp() throws MIDletStateChangeException {
 mainScreen = new Form("Exclusive List");

 original = new ChoiceGroup(null, ChoiceGroup.EXCLUSIVE);
 for(int i = 0; i < 9; i++){
  original.append("choice item " + (i + 1), null);
 }
 mainScreen.append(original);
 mainScreen.addCommand(okCommand);
 mainScreen.addCommand(exitCommand);
 mainScreen.setCommandListener(this);
 Display.getDisplay(this).setCurrent(mainScreen);
}

public void commandAction(Command c, Displayable d) {
 if( c == okCommand) {
  System.out.println("Action Button 1 selected!");
 }

 if( c == exitCommand) {
  notifyDestroyed();
 }
}
```

What is more, LWUIT for Series 40, provides a `RadioButton` Component that we could use in combination with an LWUIT Form and a `ButtonGroup` as shown below:

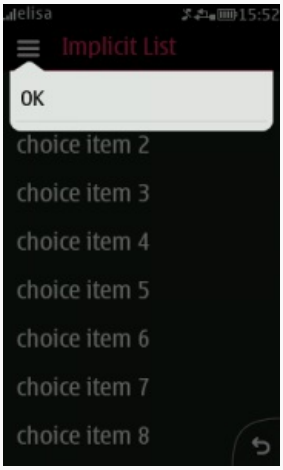
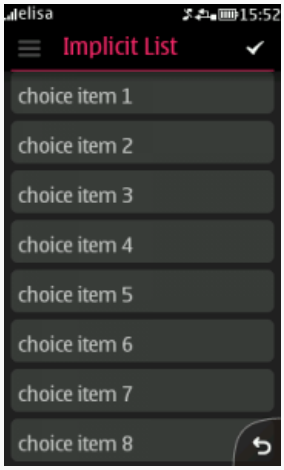
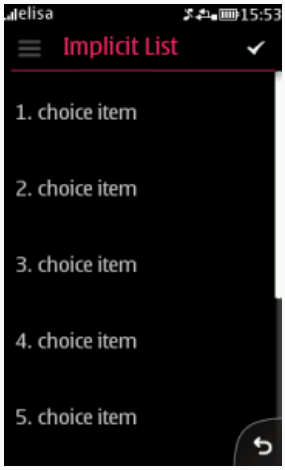
```
Form form;
Command exitCommand = new Command("Exit");
Command okCommand = new Command("OK");

protected void startApp() throws MIDletStateChangeException {

 Display.init(this);
        form = new Form("Exclusive List");
        form.addCommand(exitCommand);
        form.setBackCommand(exitCommand);
        form.addCommand(okCommand);
        form.addCommandListener(this);
        form.setLayout(new BoxLayout(BoxLayout.Y_AXIS));

        ButtonGroup group = new ButtonGroup();
        RadioButton[] radios = new RadioButton[9];

        for(int i = 0; i < 9; i++){
         radios[i] = new RadioButton("choice item " + (i + 1));
         group.add(radios[i]);
         form.addComponent(radios[i]);
        }

        form.show();
}

public void actionPerformed(ActionEvent event) {

       if(event.getCommand() == okCommand) {
```

 (C) Copyright Nokia 2014. All rights reserved.

```
        System.out.println("Action Button 1 selected!");
       }

       if(event.getCommand() == exitCommand) {
        notifyDestroyed();
       }
}
```

Implicit lists

The table below provides screenshots (and source code) for implicit lists, implemented using LCDUI, a `Form` and a `ChoiceList`, and LWUIT. Note that the LCDUI and LWUIT implementations have a functional Command on Action Button 1.

	Original Code with a List	Solution Code with a Form	Solution with LWUIT
Source	Media:ListImplicitMIDletSource.zip	Media:ListImplicitSolutionSource.zip	Media:ListImplicitLWUITSource.zip
Binaries	Media:ListImplicitMIDletBinaries.zip	Media:ListImplicitSolutionBinaries.zip	Media:ListImplicitLWUITBinaries.zip
Final Result	 LCDUI List: OK Command under Action Button 2. Action Button 1 is disabled	 LCDUI Form with CustomItems: Action Button 1 is fully functional	 LWUIT Form for Series 40 with List Component: Action Button is enabled

LCDUI Implicit Lists, suffer from the same problem where, all the high level Commands, with the exception of the EXIT Command, are added under the **Options** Menu.

Note: For implicit lists this isn't such a big problem, because choosing a list item is also implicit selection. However while an "OK" or "tick" is not required, other use cases may require a command in Action Button 1.

```
protected void startApp() throws MIDletStateChangeException {
 list = new List("Implicit List", List.IMPLICIT);
 for(int i = 0; i < 9; i++){
  list.append("choice item " + (i + 1), null);
 }
 list.addCommand(exitCmd);
 list.addCommand(okCmd);
 list.setCommandListener(this);
 Display.getDisplay(this).setCurrent(list);
}

public void commandAction(Command c, Displayable d) {
 if(c == okCmd) {
```

 (C) Copyright Nokia 2014. All rights reserved.

```
  System.out.println("Some task for Action Button 1");
 }
 else if(c == exitCmd) {
  notifyDestroyed();
 }
 else {
  System.out.println("choice item " + (list.getSelectedIndex() + 1) + " selected!");
 }
}
```

This can be fixed with LCDUI `CustomItem`s, but this solution is not as easy, because `CustomItem`s tend to have low level Canvas-like paint operations. On top of that we need to adjust the height of the item to match our needs. Each item in this case is represented as a separate class, that we call `TouchItem`. When we instantiate a `TouchItem`, we define its height. We also set the direct touch trait to `true`:

```
//Creates and appends new items
for(int i = 0; i < newItems.length; i++) {
        TouchItem item = new TouchItem(newItems[i], 20, this);

 //If the device is Series 40 Touch and Type
 if(supportsS40Touch()) {
  //enables the single touch trait
            LCDUIUtil.setObjectTrait(item, "nokia.ui.s40.item.direct_touch", new
Boolean(true));
  }
  //If the device is older Series 40 with keyboard
  else {
         //adds a Select Command as Middle Softkey
         item.addCommand(selectCmd);
  item.setItemCommandListener(this);
        }
        mainForm.append(item);
}
```

Then inside the `TouchItem` class, we paint the `CustomItem`'s Canvas starting from the topmost and leftmost pixel of the item.

Note: If the item contains text that exceeds the width of the `CustomItem`, we need to manually adjust the height of the `CustomItem` or simply cut the remaining characters.

```
    public TouchItem( String label, int itemsHeight, ListImplicitSolution midlet) {
        super( "" );
        this.label = label;
        this.itemsHeight = itemsHeight;
        this.midlet = midlet;
    }

    public void paint(Graphics g, int width, int height) {

     g.setColor(midlet.getForegroundColor());
     g.drawString(label,0, 0, Graphics.TOP|Graphics.LEFT);
    }
```

In order to return back to the MIDlet the item that has been selected we need to add some code within the `pointerPressed()` method:

 (C) Copyright Nokia 2014. All rights reserved.

```
protected void pointerPressed(int x, int y) {
 midlet.showSelected(label);
}
```

LWUIT for Series 40 has a simpler List Component that can be added to a LWUIT Form as follows:

```
protected void startApp() throws MIDletStateChangeException {

 Display.init(this);

        form = new Form("Implicit List");
        form.addCommand(exitCommand);
        form.addCommand(okCommand);
        form.setBackCommand(exitCommand);
        form.addCommandListener(this);

        form.setLayout(new BoxLayout(BoxLayout.Y_AXIS));

        String[] listItems = new String[9];

        for(int i = 0; i < 9; i++){
   listItems[i] = "choice item";
  }

  list = new List(listItems);
  list.addSelectionListener(this);
        list.addActionListener(this);
        form.addComponent(list);


        form.show();

}
```

As seen from the code above, we need to add a `SelectionListener` to the List Component in order to retrieve the selected index, each time an item is chosen:

```
public void selectionChanged(int oldIndex, int newIndex) {
 System.out.println( "choice item " + (newIndex + 1) + " selected");
}
```

See also

- LWUIT for Series 40
- CustomItem class
- ChoiceGroup class

 (C) Copyright Nokia 2014. All rights reserved.

 (C) Copyright Nokia 2014. All rights reserved.