Research In Motion

C40 Using MIDLets on BlackBerry

For BlackBerry SmartPhones

Andre Fabris

09

Contents

C40 Using MIDLets on BlackBerry

In this tutorial we will discuss how to use MIDLet applications on the BlackBerry Platform.

We will define what a MIDLet is, what are the advantages and disadvantages of using it over so called RIMLets.

We will touch topics of security, signing applications, distribution and so on.

RIMLet & MIDLet

There is no such thing as a RIMLet or MIDLet.

Let me explain. As Java Micro Edition (Java ME), formerly known as J2ME, develops, more and more features (APIs) are added. They are usually categorized as Java Specification Requests (JSR).

Each device manufacturer chooses which ones to implement on each of their development platforms, and results in greater fragmentation of the market and it makes the life of developers more difficult.

As the standard today, Mobile Information Device Profile (MIDP) 2.0 and Connected Limited Device Configuration (CLDC) 1.1 are on most common Java based devices.

All current BlackBerry models implement MIDP 2.0 and CLDC 1.1. On top of APIs used in these two specifications, the BlackBerry platform has additional APIs to cover extra features on our devices, as well as to improve the capabilities of existing Java ME APIs.

These additional BlackBerry APIs are not available on other platforms, and therefore your application will only work on BlackBerry devices, if you decide to use them.

When developing applications, your Java code is compiled using the IDE tool and then usually packed with resource files into a JAR file to be distributed.

A JAR file is just like a ZIP file and any desktop ZIP software can open it. It is possible the classes themselves can be decompiled using various decompiling tools. If you do not obfuscate your application, everyone will be able to see your code. Obfuscating does not prevent others from looking at your code, it just makes it more difficult as all classes and variables are renamed.

A JAR file as a method of distribution is not secure at all; we at RIM developed a more robust method and file format - COD. COD file is optimized for BlackBerry devices and cannot be unzipped and classes cannot be decompiled.

As an additional security feature RIM implemented Code Signing Keys. Developers can register and obtain their unique keys from RIM. Code Signing Keys also allow access to sensitive APIs, i.e. contacts, internet, phone etc. Note that only COD files can be signed. Some applications accessing secure APIs need to be signed but others do not. However as a best practice I recommend signing all your applications.

BlackBerry devices will also run JAR files. The JAR files cannot be signed, and they are actually converted into COD files on the device at the time of installation.

A JAR file can consist of Java ME classes only, RIM classes only, or a combination of both of these classes. Any such type of JAR file should work on the BlackBerry device,

The RIMLet and MIDLet do not exist as a concept. We can choose to call an application using COD files as a distribution method, BlackBerry APIs, and its main class that extends UiApplication, a

RIMLet. We can also choose to call an application that is distributed as a JAR file, which does not use BlackBerry APIs, extending the MIDLet class as a MIDLet application.

RIMLet & MIDLet Decision time

Which one should you choose when writing your application? It is a commercial decision.

If you want your application to run on a majority of devices on the market you will choose to write a MIDlet. The application will run, but it will likely not stand out. This is especially noticeable among other BlackBerry applications which use a wide range of additional features that take advantage of unique features of the BlackBerry platform.

Your application will be much more appealing and more successful on BlackBerry if you create a RIMLet. However, it will not work on other devices as they have not implemented BlackBerry specific APIs.

Or you can choose to write a hybrid application. You can deploy your application as a COD file, using BlackBerry APIs, and sign your application but extend MIDLet.

It is quite easy to add some of those specific BlackBerry features to your application. This tutorial will try to cover some of those features.

At the end of the day it depends what your application is doing, how customized you want it to be for your user, while taking into account development time and economics. Basic applications i.e. a currency converter may work fine as a MIDLet, but something more advanced which requires a richer UI experience, integration with PIM features on the device, i.e. instant messenger, social networking or CRM would probably benefit as RIMLets.

If you already have an existing MIDLet application, it may not be economically viable to rewrite it completely using BlackBerry APIs, however, in this article you will find some quick fixes, which can significantly improve user experience.

RIMLet & MIDLet Detailed

Converting JAR into COD

As mentioned before the device itself can convert a JAR file into a COD file.

However, you will want to do the conversion yourself before you distribute your application so you can sign it.

To find out more about code signing, please have a look at our ‘How and When to Sign’ tutorial. The links are provided at the end of this article.

1. Create a directory on your hard disk drive and copy the MIDlet JAD and JAR files to this location.
2. In the BlackBerry JDE, select **File** > **New Workspace**.
3. Change the location to the folder created in step 1, type a name for the new workspace, and click **OK**.
4. Once the workspace is loaded, right-click **Create new project in [*NameofWorkspace*].jdw**. Place the project that will be created in the same location as the folder in step 1. Type a name for the project and click **OK**.
5. Once the project is displayed, right-click **Add File to Project**.
6. From the **Add Files to [*Name of Project*]** screen, in the Files of Type drop-down list, select **All Files**.
7. Change to the folder created in step 1, select both the JAD and JAR files from the MIDlet, and click **Open**.

Both the JAD and JAR files appear under the project name within the tree.

8. Right-click the project name and select **Properties**.
9. On the Application tab, from the Project Type drop-down menu, select **MIDlet**.
10. Under the **Name of main MIDlet class** title, type the main MIDlet class and click **OK**.

The fully qualified package and class name are required here.

11. From the main menu, select **File** > **Build** (or press **F7**) to build the MIDlet.

The compiled COD file will now reside in the directory created in step 1.

If your application requires signing you can use JDE to sign your COD file once it is created.

You can also use the command line command:

```
rapc import="c:\Program Files\Research in Motion\Blackberry JDE
4.2\lib\net_rim_api.jar" codename=virca -midlet jad=Virca.jad Virca.jar
```

RAPC is bundled with any BlackBerry JDE, in the above sample, it is version 4.2 but you can and should replace it with targeted OS version.

Virca, Virca.jad and Virca.jar should be replaced with the name of your application.

For more information about the RAPC tool please follow the link to our KB article in the Links section.

Signing MIDLets

MIDLets do not require signing. The BlackBerry device will run any MIDlet application which uses standard APIs. Jar files cannot be signed using BlackBerry Signing keys. It means that the application will be regarded as untrusted, and will notify the user about that upon installing the application.

Some of the users might not be comfortable running such an application. On top of that, without signing, the application will not have access to a large number of “sensitive” APIs. Applications which use these “sensitive” APIs and are not signed will not run on the device. Some of the “sensitive” APIs will just prompt the user to allow or deny access to the certain features of the device. Signing will reduce the number of these prompts.

After you convert the JAR file into a COD file, it can be signed using the BlackBerry signing keys. Once signed, the application will be able to run all the “sensitive” APIs and will not notify the user that the application is not signed.

For more information on signing, please have a look at the tutorial ‘A60 – How And When To Sign’.

The Better Way

The existing MIDlet application, converted to a COD file and signed, will work on a BlackBerry device, but it needs to be optimized.

Many devices have soft keys these days; however BlackBerry SmartPhones have Menu and Escape keys instead.

There are also Volume Control keys, Clickable touch screens, Track pads, Trackballs, and Trackwheels, which might not always work with your application if you simply just convert an existing MIDlet to a COD file and sign it.

Also the BlackBerry screens are usually bigger and the screen resolutions higher than an average device on the market, and so the application might not render properly.

BlackBerry applications might also be deployed on BlackBerry Enterprise Server (BES) devices – Companies usually purchase BES and have their devices “activated” on this server. This allows IT departments to manage their deployment of devices and enforce permissions, such as which resources applications and/or users can and cannot access.

This might prevent your application from functioning the way you expect or can cause strange behaviour.

The “Better Way” is to actually add these and similar functionalities to your application so that users will have a much better experience.

So you as a developer can choose to:

1. Run an existing JAR application on BlackBerry
    a. No cost
    b. Problems with permissions and not optimized user experience
2. Convert a JAR file into COD and Sign the application
    a. Virtually no cost – Code Signing Keys cost about 20USD
    b. No problems with the permissions, but still problems with not optimized user experience
3. Convert and Sign an OPTIMIZED application.
    a. Very short development time – Usually couple of days
    b. Much better overall user experience.

This tutorial will show you how to optimize MIDLets.

Input Paradigm

Most BlackBerry devices use the full keyboard. Some of them have a SureType keyboard or use a touch screen. All BlackBerry models have the Menu and Escape (Back or Cancel) keys and do not have soft keys. Soft keys are widely used by other manufacturers, and if porting an application to the BlackBerry platform, the functions mapped to those keys will not work. One of the approaches taken by developers is to map these keys to “q” and “p” key, which is not recommended as is not intuitive for the user.

BlackBerry users are very used to actions performed by the Menu and Escape keys, so it is highly recommended that you modify your application to follow this standard.

If your application uses Command Actions they will be added to the Menu Key; they will not be assigned to the left, middle and right action key since these keys do not exist on the BlackBerry.

If you want to override the behaviour of the Menu Escape Keys (Figure 1), you will need to detect when the user clicks on them.

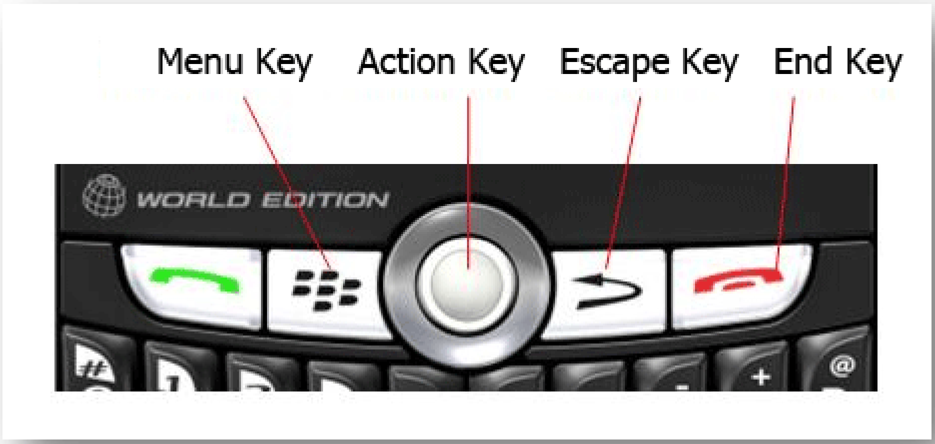


Figure 1

To be able to detect those keys, your application will need to implement a BlackBerry specific API - KeyListener interface. KeyListener has around 5 methods that you need to implement, but the one of interest is the keyDown method.

As you can see from the code below there is a specific number (integer) returned when user presses these and other keys in the application.

There is a class `Keypad` from `net.rim.device.api.ui` package which is used to map the keys and it can be used to check out the physical layout of the keyboard. For example, you can check whether the keyboard is QWERTY or QWERTZ using the `getHardwareLayout()` method.

```
import net.rim.device.api.system.Application;
import net.rim.device.api.system.KeyListener;

Public class myCanvas extends BlackBerryCanvas
 implements KeyListener
// you can also extend Canvas, GameCanvas or BlackBerryGameCanvas
…..

public myCanvas()
     {
         Application.getApplication().addKeyListener(this);
     }
……..
public boolean keyDown(int arg0, int arg1) {
        if (arg0 == Keypad.KEY_MENU) {
            //this is the menu key - code 1769472
            return true;
        } else if (arg0 == Keypad.KEY_ESCAPE) {
            // this is the escape key - code 268566528
            return true;
        } else {
            return false;
            // returning false here will not consume the event (key
press) and it can be processes by other key handling methods.
        }
    }
……..
```

Returning true will consume the event, and the system will not launch the standard BlackBerry menu, or perform the standard escape action. All the other keys are not consumed and are passed to other methods such as `keyPressed()`. keyPressed() is the standard Canvas method that handles user input.

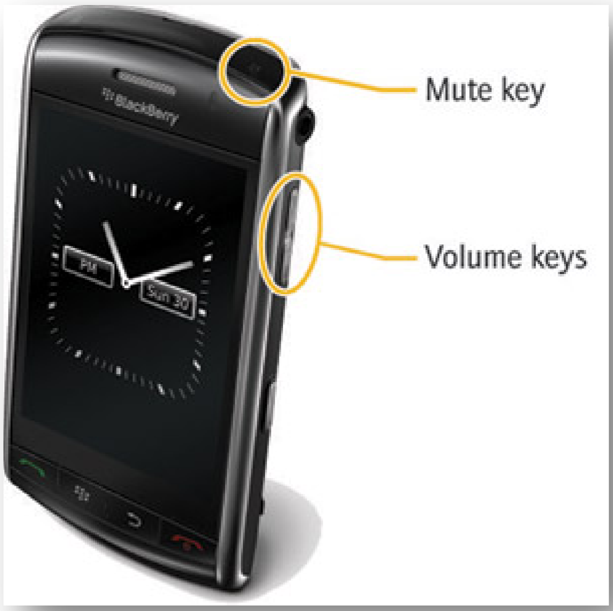


Figure 2

To detect the rest of the keys you can use the keyPressed method in any of the canvas classes.

Key codes -150 and -151 are for volume keys, and -8 are for the action key.

You can also use constants from the Characters interface such as `Characters.CONTROL_VOLUME_DOWN` or `Characters.CONTROL_VOLUME_UP` for instance. The Characters interface belongs to the RIM API set but if you do not want to use RIM APIs you can use literals. If you want to detect the Left and Right convenience keys, look for key codes -21 and -19 respectively.

In the sample I added a System.out call which will print all the codes for the keys when the user presses them so you can find out all the available codes.

```
protected void keyPressed(int keyCode) {
        try {
            if (keyCode == Characters.CONTROL_VOLUME_UP) {
                //volume up -150
                }
            } else if (keyCode == Characters.CONTROL_VOLUME_DOWN) {
                //volume down -151
            }
            if (keyCode == -8 | keyCode == 10) {
                //action key and enter keys respectively
            }
            System.out.println(keyCode);
        }
```

User Interface

Any successful application has to be visually appealing, easy to use and intuitive – user friendly.

RIM provides a huge range of RIM classes to control the screen and user interaction in the `net.rim.device.api.ui` package. These classes will allow you to create an application which looks and feels like a RIM application, and every BlackBerry user will find your application easy to use.

On top of these classes, developers create custom Fields, Managers and Screens, to provide a richer UI experience and more functional applications.

Standard Java ME provides a range of classes as well which are in the `javax.microedition.lcdui` package. These classes have far less capabilities then RIM ones, and since they have to work on all devices implementing this package they are very general. It means that every device will try to interpret these classes in its own way, and the same application will look and feel completely different on different devices.

That is the reason most developers using standard Java ME choose to use Canvas and GameCanvas classes in order to fully customize the way their application looks and works.

IMPORTANT: the only two packages you cannot use at the same time in your application are `net.rim.device.api.ui` and `javax.microedition.lcdui`.

What does this mean? Well it means if your application is using the Canvas class, you cannot add any nice BlackBerry UI components such as BrowserField or MapField for instance.

It also means if you use BlackBerry UI classes, they will not work on non BlackBerry devices.

Icon and Screen Sizes

Different BlackBerry devices have different screen sizes, even different layouts (landscape and portrait on the BlackBerry Storm device), as well as different icon sizes. To make your application look perfect please keep these sizes in mind.

You can use one size i.e. 80x80 pixels for all icons, and BlackBerry device will rescale them, however not all the icons have the same proportions and sometimes those icons might not look as good.

Your application can have a rollover icon, which changes as the user moves focus on and off your application's icon. You can assign two icons using the BlackBerry JDE or Eclipse plug-in tools.

BlackBerry device model	Display screen size	Default theme	Application icon size for the default theme
7100 Series	240 x 260 pixels	Dimension	Icon layout: 60 x 55 pixels Zen layout: 48 x 36 pixels
8700 Series	320 x 240 pixels	Dimension	Icon layout: 53 x 48 pixels Zen layout: 48 x 36 pixels
8800 Series	320 x 240 pixels	Dimension	Icon layout: 53 x 48 pixels Zen layout: 48 x 36 pixels
Bold Series	480 x 320 pixels	Precision	80 x 80 pixels
Curve 8300 Series	320 x 240 pixels	Dimension	Icon layout: 53 x 48 pixels Zen layout: 48 x 36 pixels
Curve 8350i	320 x 240 pixels	Precision	52 x 52 pixels
Curve 8520	320 x 240 pixels	Precision	52 x 52 pixels
Curve 8900	480 x 360 pixels	Precision	80 x 80 pixels
Pearl 8100 Series	240 x 260 pixels	Dimension	Icon layout: 60 x 55 pixels Zen layout: 48 x 36 pixels
Pearl Flip Series	240 x 320 pixels	Precision	46 x 46 pixels
Storm Series	portrait view: 360 x 480 pixels portrait view with keyboard: 360 x 247 pixels landscape view: 480 x 360 pixels landscape view with keyboard: 480 x 156 pixels	Precision	73 x 73 pixels
Tour 9630	480 x 360 pixels	Precision	80 x 80 pixels

To learn more about differences between MIDlet and RIMlet UI, I highly recommend watching the video ‘Contrasting MIDlets and BlackBerry Java Applications’. The link is given at the end of this tutorial.

Another interesting read is the BlackBerry Java Application MIDlet – Development Guide.

Canvas

If your application is intended for BlackBerry touch screen devices such as Storm, and uses the Canvas or GameCanvas class, you should replace these two with BlackBerryCanvas and BlackBerryGameCanvas classes. These two new classes were introduced in OS 4.7.

Standard Canvas and GameCanvas classes do support touch screens, but not in the way BlackBerry SureClick technology works. The main problem is that they cannot distinguish between a tap and click on the screen.

BlackBerryCanvas and BlackBerryGameCanvas have a method called touchEvent which will notify your application of any kind of user input on the touch screen, including clicking and gestures.

touchEvent

This method will be called every time the user interacts with the touch screen. So any tapping, clicking, moving, swiping and so on, will generate an event, which we can then process as we want.

If you use BlackBerry UI APIs (`net.rim.device.api.ui`), the system will automatically process most of these events depending on which component the user used – i.e. Buttons will click, password fields will bring the keyboard up, and * will appear as the user types on the keyboard and so on.

On the other hand, you will have to process all this information manually if you use standard Java ME UI APIs (`javax.microedition.lcdui`).

Let’s say for example you want to create a button, which will highlight when a user touches it, and perform an action when the user clicks on it.

The only way to do this if you use any canvas class is when touch screen interaction is detected, read the x and y coordinates of the area that was touched, and If they fall within the boundaries of the button (which you have to specify in pixels), you can perform one of the actions, highlight if tapped, or button action if clicked.

As I said before, canvases are low level tools, and it is not easy to program and create a beautiful and easy to use user interfaces. However, experts can create amazing results.

Storing Data

Standard MIDLet applications have to use RMS RecordStore APIs to store the data in the device memory. Anyone who uses this API is aware of how limited and relatively hard to use it is compared to the Serialization and De-serialization of objects in J2SE.

RIM provides Persistent Store APIs, which allow you to store entire objects in the device's flash memory.

Decision time again – should you use Persistent Store or Record Store? Once again the Persistent Store is a much more powerful API and will work only on BlackBerry devices. Record Store will work on both BlackBerry and other Java based devices.

For more information on the Persistent Store please have a look at the 'A13 Storing Persistent Data' tutorial.

On the BlackBerry platform you can also store data in the File System. You can create files in the device's memory or memory card (if present) and as of 5.0 you can even use SQLite to store data in a database on the device itself.

Beyond MIDLets

If you decide not to support BlackBerry APIs in your application to allow for portability of your application, you should know what you are missing:

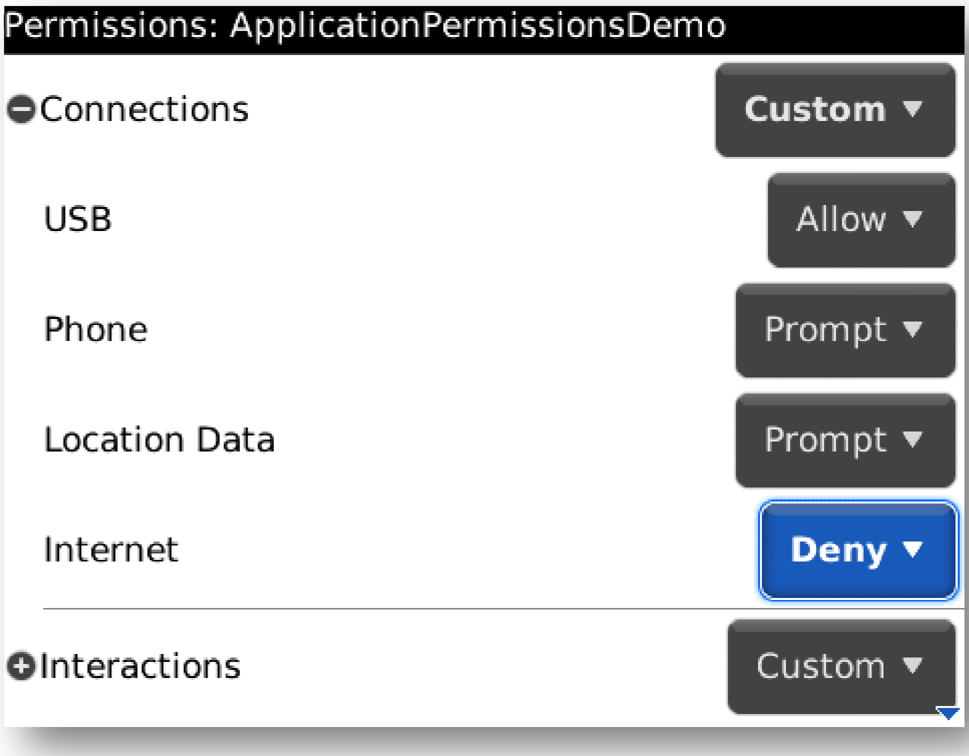
- BlackBerry UI
  - BitMapField
  - ButtonField
  - CheckBoxField
  - ChoiceField
  - NumericChoiceField
  - ObjectChoiceField
  - DateField
  - GaugeField
  - LabelField
  - ListField
  - KeywordFilterField
  - ObjectListField
  - MapField
  - NullField
  - RadioButtonField
  - SeparatorField
  - TextField
  - EditField
  - PasswordEditField
  - RichTextField
  - ActiveRichTextField
  - TreeField
- Integration with BlackBerry native applications & APIs:
  - Browser
  - E-Mail
  - Maps
  - Menu
  - PIM
  - Phone
  - Crypto
- PhoneListener
- GPRSInfo / CDMAInfo/ Radio Info/ Device Info / Coverage Info
- GlobalEventListener
- Persistent Store
- Application Permissions Manager
- Invoking applications
- Service Books

The BlackBerry specific features which you are losing if you don't use the BlackBerry APIs are very extensive. To find out more about those APIs please visit our BlackBerry developer web site and check available documentation.

I will mention some of them here:

Managing Application Permissions

The `ApplicationPermissionManager` class is very important when you use "sensitive" APIs. Each application on the device has a list of permissions which can prevent your application from running at all or display annoying prompts. Figure 3 shows the permission for Internet access for the application is denied. The application will throw an exception when trying to access the Internet.



Permissions: ApplicationPermissionsDemo

Connections	Custom
USB	Allow
Phone	Prompt
Location Data	Prompt
Internet	Deny
Interactions	Custom

Figure 3

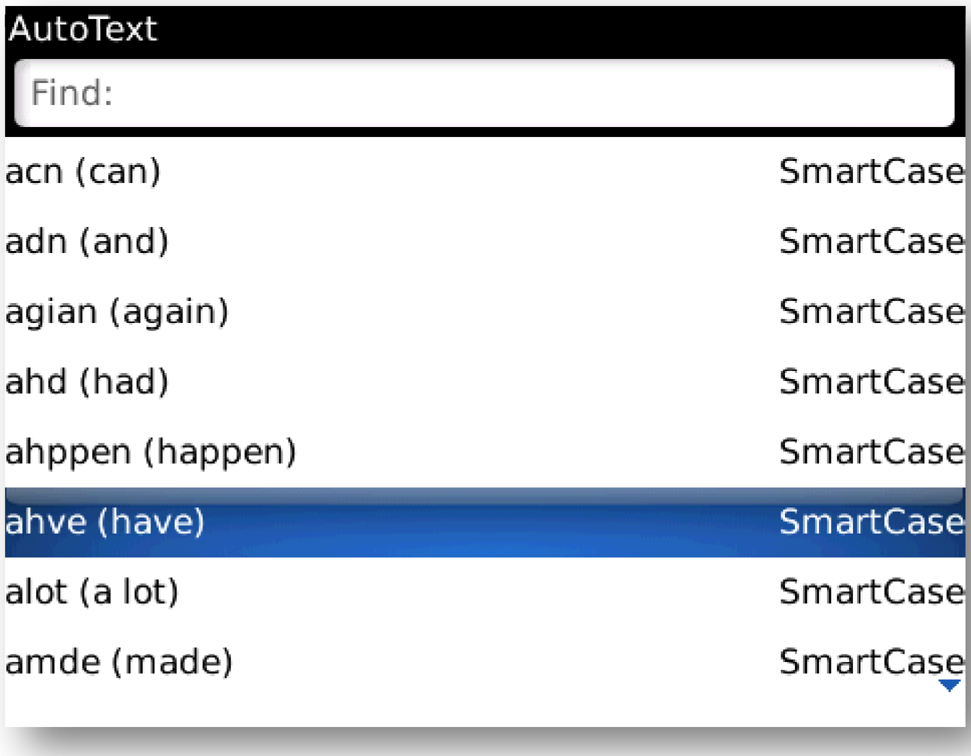
This class can be used to detect the correct permissions needed before the application starts, and request the user to change them if they are not already set.

Checking Network Coverage

Another interesting API is the `CoverageInfo` API. With this API, your application can obtain information on what type of coverage is currently available to the BlackBerry device. This API does not just check the coverage but whether the connection is actually routable.

Auto text correction

`ActiveAutoTextEditFields` is just one of a number of useful classes which extends the Field Class. Auto Text helps users while typing on the device, i.e. word “ahve” is automatically replaced with “have” (Figure 4).



AutoText

Find:

acn (can)	SmartCase
adn (and)	SmartCase
agian (again)	SmartCase
ahd (had)	SmartCase
ahppen (happen)	SmartCase
ahve (have)	SmartCase
alot (a lot)	SmartCase
amde (made)	SmartCase

Figure 4

Accelerometer Listener

The Accelerometer listener is used to detect changes in the actual device physical orientation. It was introduced in OS version 4.7. It is used for detecting whether the device is in portrait or landscape mode, and is very popular as an additional control in games.

If needed, all Canvas classes have the `sizeChanged(int w, int h)` method which is called when the screen size changes. It can be used to detect screen rotation changes.

BlackBerry OS 5.0

The new 5.0 APIs are targeted to improve the development platform within the following categories:

- **Increase developer productivity**
  JDE 5.0 will provide high-level APIs and hide complexity from developers, such as the Network API and Hotspot API.

- **Enable compelling Java user interfaces**
  This has been a long term ask from our developer community. In the JDE 5.0 release, we will publish 20 new UI APIs, and exposing RIM internal UI controls as well as high level UI APIs such as contact picker, data/time picker, spinner, location picker, file picker etc.

  On a high level, the following UI aspects will be addressed in JDE 5.0:

  - **UI Pickers**
    Spinner, date and time picker, file picker, location picker

  - **Customization**
    Menu customization (font, background, icon), sub menu, custom font support, and application management (IDE feature, not API)

  - **Transition and Effects**
    Screen transition, OpenGL (JSR 239)

  - **User Interaction**
    Picture scroll API (sliding images side to side), eyelid field API (header/foot display areas with animation effects), image zoom in/out.

  - **Screen Layout**
    Grid layout manager

- **Enhanced application and data integration**
  A significant amount of development work focuses on this category. The JDE 5.0 will include features such as SQLite, phone screen integration, embedded video capturing and streaming, link-to-Blackberry contacts, reverse geo-coding and LBS extensions, email attachment downloads etc.

OS 5.0 UI Transitions

Let me show you some of the new UI features in OS 5.0

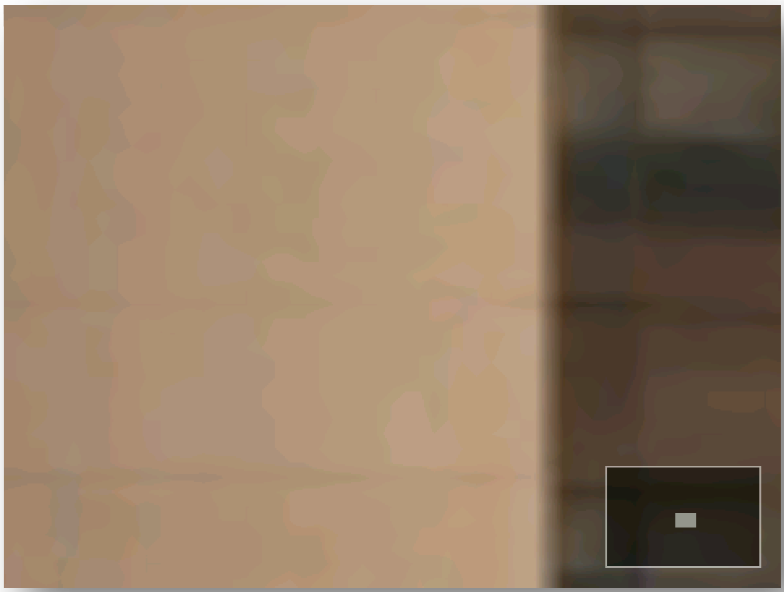
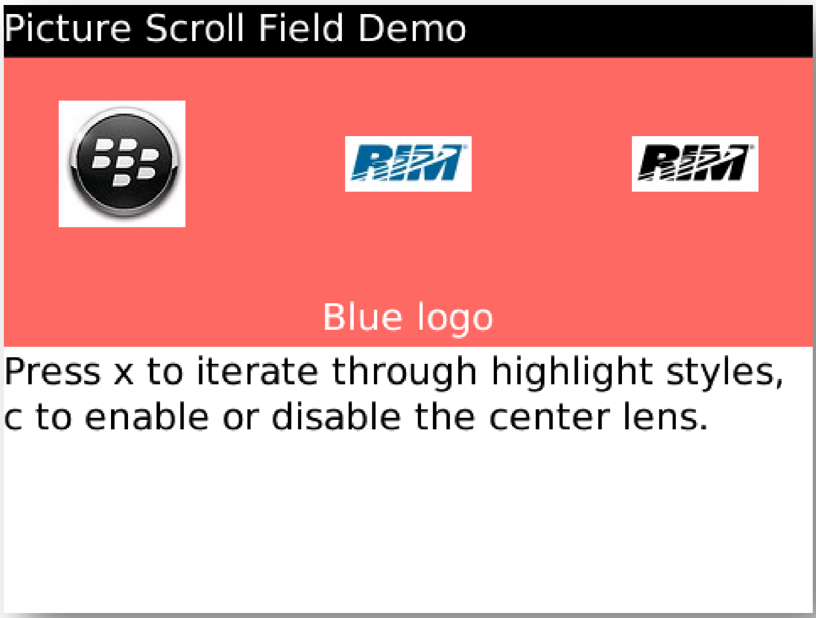


Figure 5

Figure 5 shows zooming in an out – very simple to zoom in and out on image as well as panning. There is even a mini map.



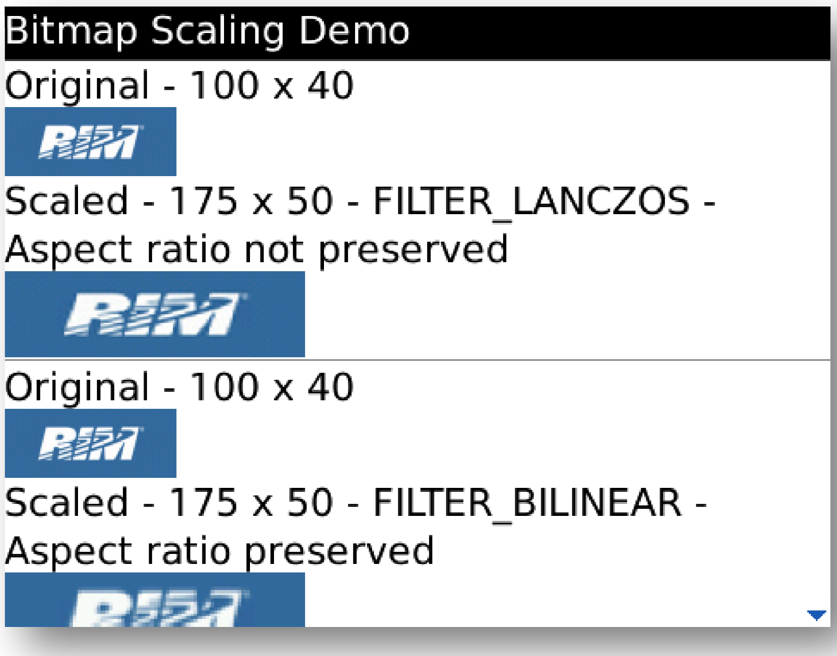
Picture Scroll Field Demo

Blue logo

Press x to iterate through highlight styles, c to enable or disable the center lens.

Figure 6

Figure 6 shows new Picture Scroll Field, which can be used to display carousel of images, use tool tips, select images etc.


Bitmap Scaling Demo

Original - 100 x 40

Scaled - 175 x 50 - FILTER_LANCZOS - Aspect ratio not preserved

Original - 100 x 40

Scaled - 175 x 50 - FILTER_BILINEAR - Aspect ratio preserved

Figure 7

Figure 7 shows the new API for handling Bitmap Images – different ways to scale bitmaps.

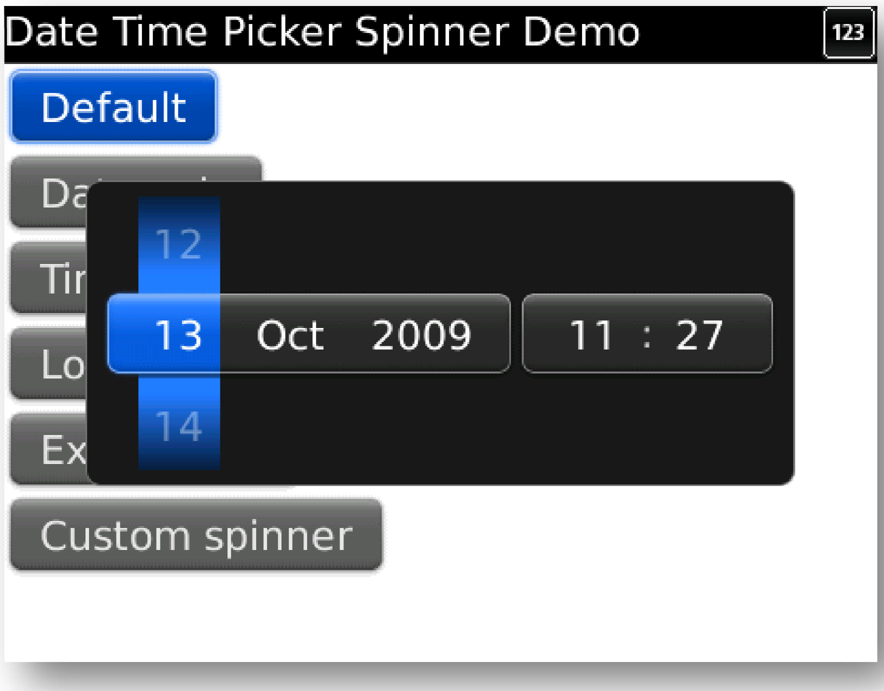

Date Time Picker Spinner Demo

Figure 8

Picker Spinner is shown on Figure 8. Apart from selecting dates and time, it can be used to select custom items.

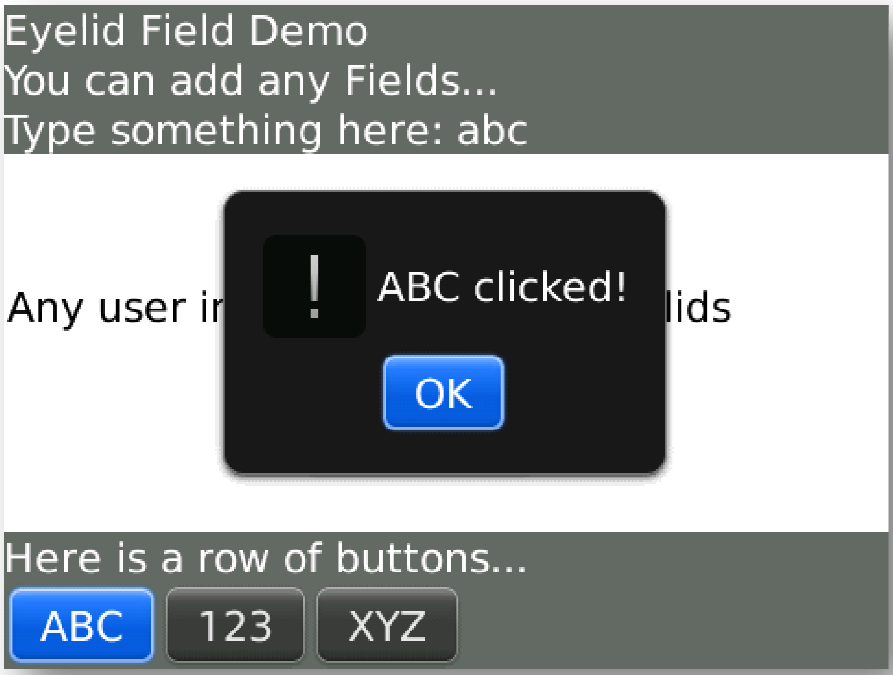


Figure 9

Figure 9 shows Eyelid Field. The top and bottom banner disappear after few seconds of inactivity, and come back as soon as user makes any input. This is very useful on small screens.

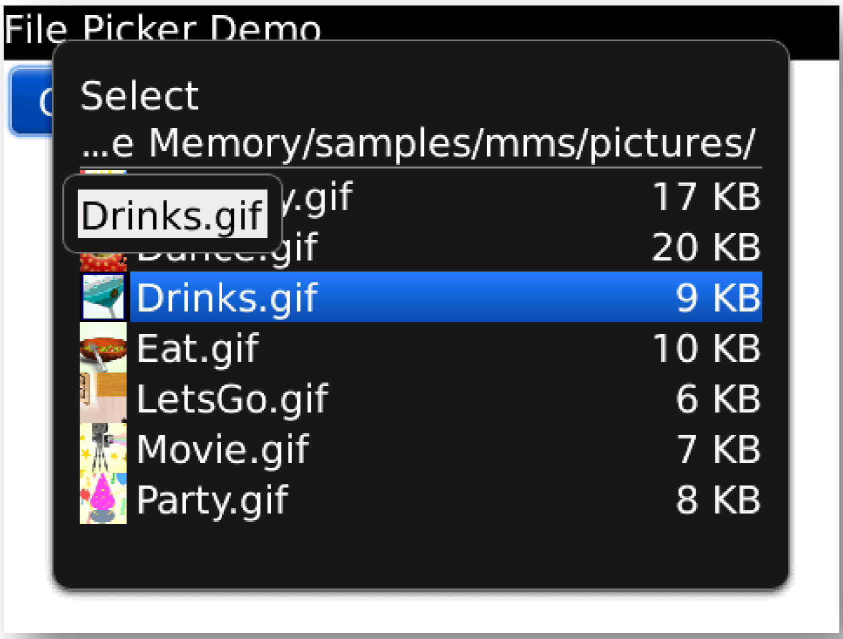
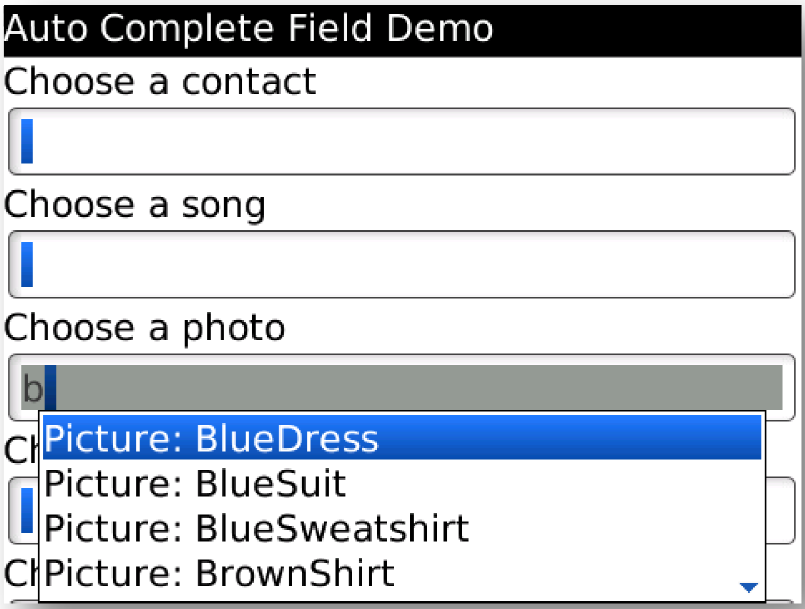


Figure 10

On Figure 10 we can see File Picker Demo. No need to write your own file pickers anymore. Even thumbnails are included.



Auto Complete Field Demo

Choose a contact

Choose a song

Choose a photo

b

Picture: BlueDress

Picture: BlueSuit

Picture: BlueSweatshirt

Picture: BrownShirt

Figure 11

As soon as the user starts typing in the new Auto Complete Field, the matching results are displayed as shown on Figure 11. This is a very user friendly way to handle user input.

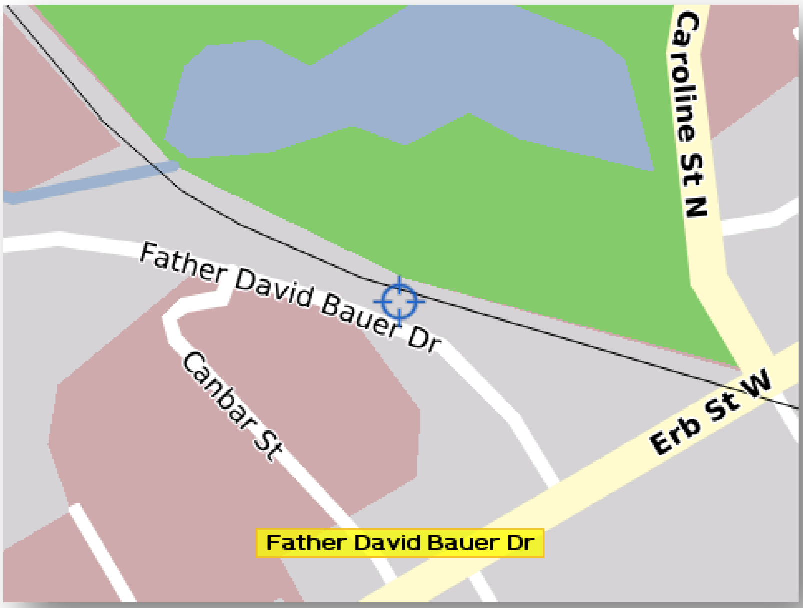
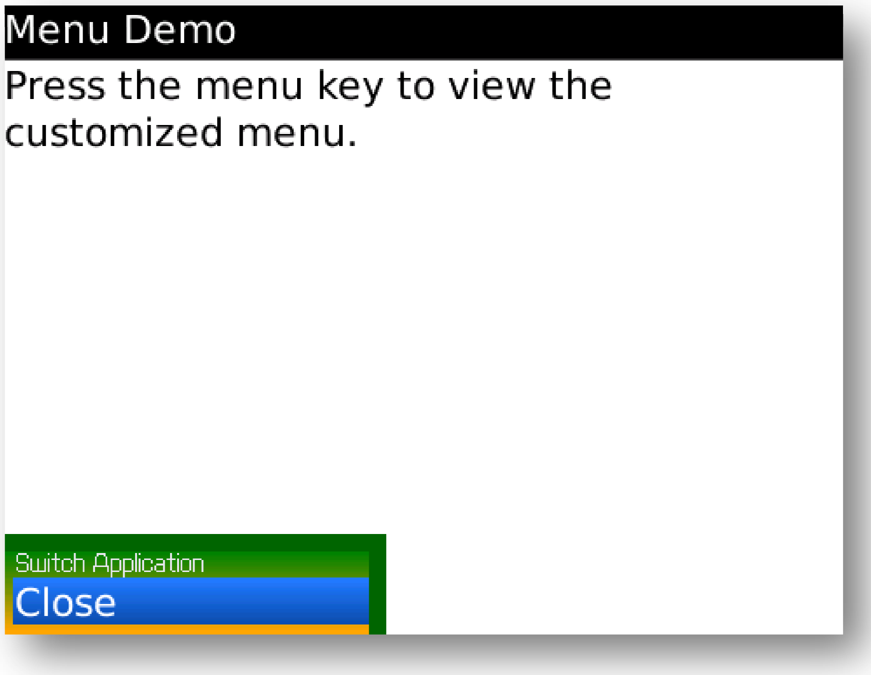


Figure 12

If the user needs to select a location they can use the new Location Picker. In Figure 12 the user selects the location by clicking on the map. Other methods of selection are supported as well.

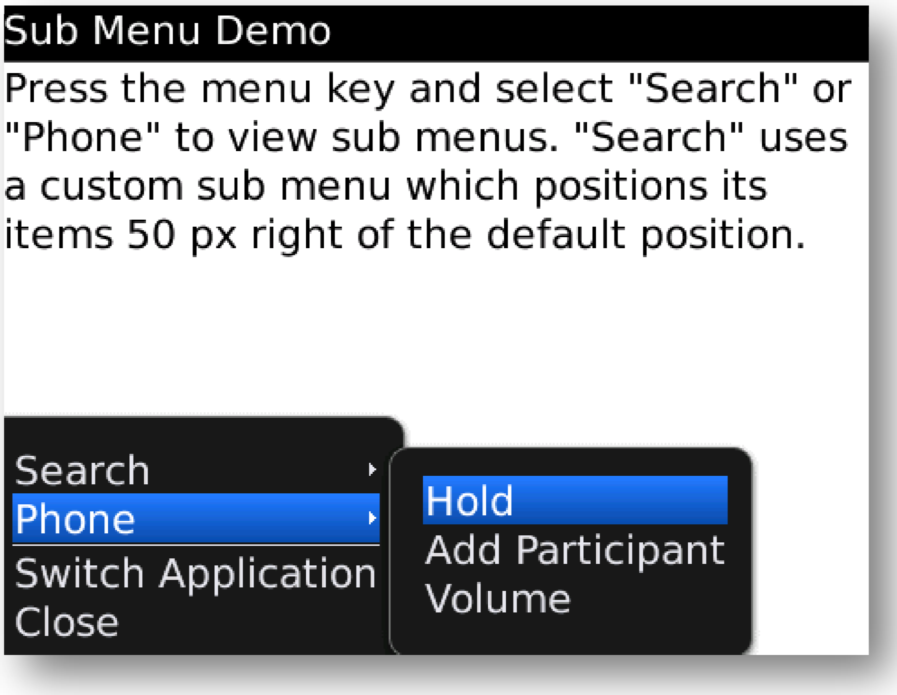


Menu Demo

Press the menu key to view the customized menu.

Figure 13

If you do not like standard BlackBerry menus, customized menus are for you. Figure 13 shows different colours, frames, and magnification of the selected menu item.

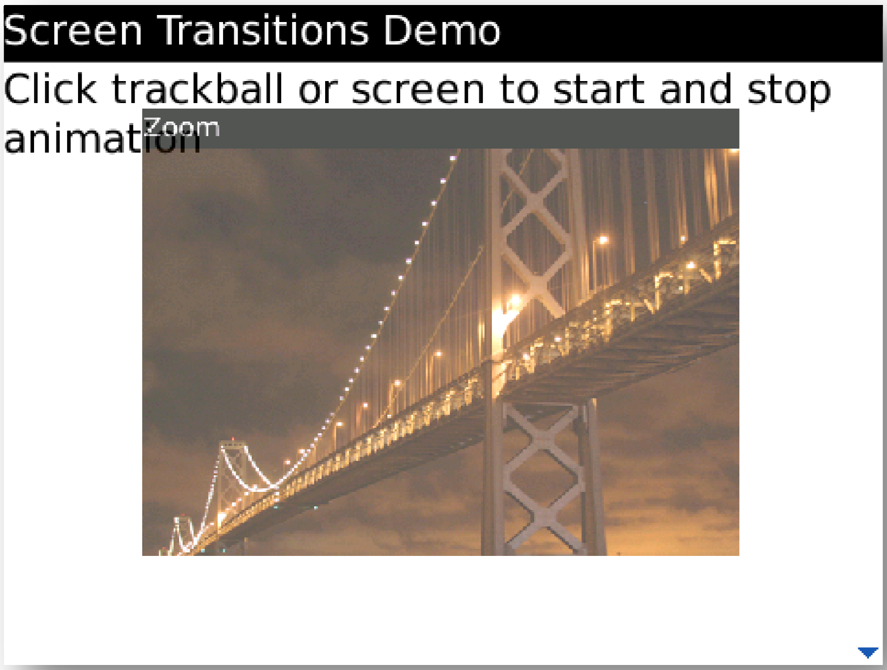


Sub Menu Demo

Press the menu key and select "Search" or "Phone" to view sub menus. "Search" uses a custom sub menu which positions its items 50 px right of the default position.

Figure 14

In Figure 14 you can see the sub menus presented to the user.



Screen Transitions Demo

Click trackball or screen to start and stop animation

Figure 15

Figure 15 shows one of many new screen transitions. Now with virtually no effort, your applications can look stunning.

Conclusion

Without a doubt, additional RIM APIs give developers much more flexibility and the ability to provide customized functionality for their BlackBerry applications. Existing MIDLets can be used as is on BlackBerry Smartphones or they can be enriched by adding BlackBerry specific features.

On the other hand applications customized for BlackBerry will not work on other platforms, and if portability is of major concern, RIM APIs should not be used, or you can opt to have two builds: one for BlackBerry and one for all other J2ME devices. Even if portability is a concern, you can see how easy it is to add some BlackBerry specific features, and I highly recommend doing so.

At the end of the day there are three options you can choose:

1. Run an existing J2ME (JAR file) application on BlackBerry
    a. No cost
    b. Problems with permissions and not optimized user experience
2. Convert the JAR file into COD and sign the application
    a. Virtually no cost – Code Signing Keys cost about 20USD
    b. No issues with application permissions, but still problems with non-optimized user experience
3. Converted and Signed OPTIMIZED application
    a. Very short development time – Usually couple of days
    b. Much better overall user experience.

Links

BlackBerry Developers Web Site:

http://na.blackberry.com/eng/developers/

Tutorials

- Storing Persistent Data

  http://na.blackberry.com/developers/resources/A13_Storing_Persistent_Data_V2.pdf

- How And When To Sign

  http://na.blackberry.com/developers/resources/A60_How_And_When_To_Sign_V2.pdf

Developer Video Library:

- Contrasting MIDlets and BlackBerry Java Applications

  http://www.blackberry.com/DevMediaLibrary/view.do?name=ContrastingMIDletsandBlackBerryJavaApplications

- How do I deploy my MIDlet (JAR file)

  http://www.blackberry.com/DevMediaLibrary/view.do?name=howfinal

Documentation:

- Documentation for developers can be found here:

  http://na.blackberry.com/eng/support/docs/developers/?userType=21

- BlackBerry Java Application MIDlet
  http://docs.blackberry.com/en/developers/deliverables/9127/JDE5.0_MIDletGuide_Beta.pdf

Knowledge Base Articles:

- How To – Compile a MIDlet into a COD file

  http://www.blackberry.com/knowledgecenterpublic/livelink.exe/fetch/2000/348583/800901/How_To_-_Compile_a_MIDlet_into_a_COD_file.html?nodeid=800921&vernum=0

- How To - Use the RAPC compiler

  http://www.blackberry.com/knowledgecenterpublic/livelink.exe/fetch/2000/348583/800901/How_To_-_Use_the_RAPC_compiler.html?nodeid=800818&vernum=0

- How to – Create an icon for an application

  http://www.blackberry.com/knowledgecenterpublic/livelink.exe/fetch/2000/348583/800332/800505/800608/How_To_-_Create_an_icon_for_an_application.html?nodeid=800515&vernum=0

- How to – Define a rollover icon for an application

  http://www.blackberry.com/knowledgecenterpublic/livelink.exe/fetch/2000/348583/800332/800505/800608/How_To_-_Define_a_rollover_icon_for_an_application.html?nodeid=1162799&vernum=0

Forums:

- The link to BlackBerry Development Forums:

  http://supportforums.blackberry.com/rim/?category.id=BlackBerryDevelopment

Blogs:

- BlackBerry Developer's Blog

  http://supportforums.blackberry.com/t5/BlackBerry-Developer-s-Blog/bg-p/dev_blog

Developer Issue Tracker:

- To submit issues and feature requests you can use this web application:

  http://na.blackberry.com/eng/developers/resources/issuetracker/