

# BLACKBERRY DEVELOPER JOURNAL

**main()**

**2**

**Java ME Gaming Under MIDP 2.0**

**3**

- A complete summary of the MIDP 2.0 Gaming API plus full source code for the Bones game!

**Mobilize the BlackBerry Resource Kit**

**32**

- How to take an existing business operation and enable it for use in a Windows® executable program, a web user interface, and as a BlackBerry MDS Studio Application on a BlackBerry device

**The BlackBerry Browser does JavaScript**

**48**

- Most everything you've ever wanted to know about JavaScript™ support on the BlackBerry Browser

**Profiling BlackBerry Devices for Microsoft  
.NET Mobile Controls**

**56**

- Creating a device profile for BlackBerry devices to work with the Microsoft .NET Mobile Controls framework

**Using String Pattern Matching to Add  
Dynamic Menu Items**

**68**

- Adding dynamic menu items based on the matching of text in active text fields with string patterns.

# main()

Noun: *inertia*

1. A disposition to remain inactive or inert as in “he had to overcome his inertia and get back to work.”
2. (physics) the tendency of a body to maintain a state of rest or uniform motion unless acted upon by an external force

Many people live in a state of perpetual inertia due to choices they make and labels assigned to them throughout their lives. Adults, friends and fools have incredible power to empower or hurt others by passing judgment on intelligence, strengths and weaknesses. In school, teachers often pass early judgment on kids who are good at achieving high grades, completing assignments and taking tests. These children are usually labeled as gifted, smart, special, and sure to succeed. The rest are labeled in ways that can often do more harm than good.

In grade school, my teachers would return tests and assignments in sequence of mark. The “smart” kids always got their work back first while the “dumb” kids were last. I was a “smart” kid. This led to problems where some of the kids resented the “smart” kids and retaliated during recess and lunch. I quickly learned to lower my grades to avoid being singled out. By late grade school the teachers had changed their ways, but the kids knew where everyone fit in the scale of life. The “smart” kids were still given special treatment where the “dumb” kids were ignored.

By high school, “smart” kids moved into advanced studies, “dumb” kids moved towards basic studies or into vocational schools, while everyone else stayed in general studies. At each level teachers would further refine the caste system in the classroom by focusing on the more promising kids while occasionally ignoring the others. Over time, personal attributes formed that help to set individual characteristics. For those who accepted their labels as true, positively-labeled people work towards expected goals while others try to get by and, hopefully, do the best they can. Some give up entirely, never expecting to achieve much in life.

The ones who do not accept negative labels tend to do far better than those who accept them. Occasionally they even do better than people who were blessed with positive labels throughout their lives. These people tend to work harder than most to compensate for their purported lack of intelligence, ability and talent. They believe that honesty, love of vocation, getting along with others, being an effective leader by example, family, fidelity, loyalty, and living beneath their means, are all key to success. As it turns out, these traits are far more important to success than a high IQ, great university credentials and a good job.

Even people labeled as “smart” in school and who now have high-paying jobs where they excel can suffer from inertia. In many situations, a high-paying job that no longer provides challenges can lead to inertia. The job becomes too easy, the money and benefits too good, and there appears to be no logical reason to take a risk to break the cycle. Most people dislike challenging the unknown and prefer living in a virtual state of comfortable inertia. Unfortunately, inertia is the anti-thesis of creativity and growth. People who break free of inertia make the largest difference in life. They are willing to do things that make others uncomfortable, take risks, challenge the unknown, fail from time to time, and occasionally win.

The BlackBerry® solution and all related technology has been designed and developed by people who detest inertia and enjoy tackling the unknown. If you trace the history of most successful companies, you’ll find that the founders challenged the unknown and won over time. Most didn’t win every time, and some took some major hits before they finally gained strength. But all kept working towards achieving their dream no matter the cost. Perhaps it’s time for you to do the same by taking a chance to realize your dreams.

As always we are interested in your comments, suggestions, letters, and anything else that you feel would be of benefit to our developer community.

Please feel free to email me at [Editor@BlackBerryDeveloperJournal.com](mailto:Editor@BlackBerryDeveloperJournal.com).

Richard Evers

# Java ME Gaming Under MIDP 2.0

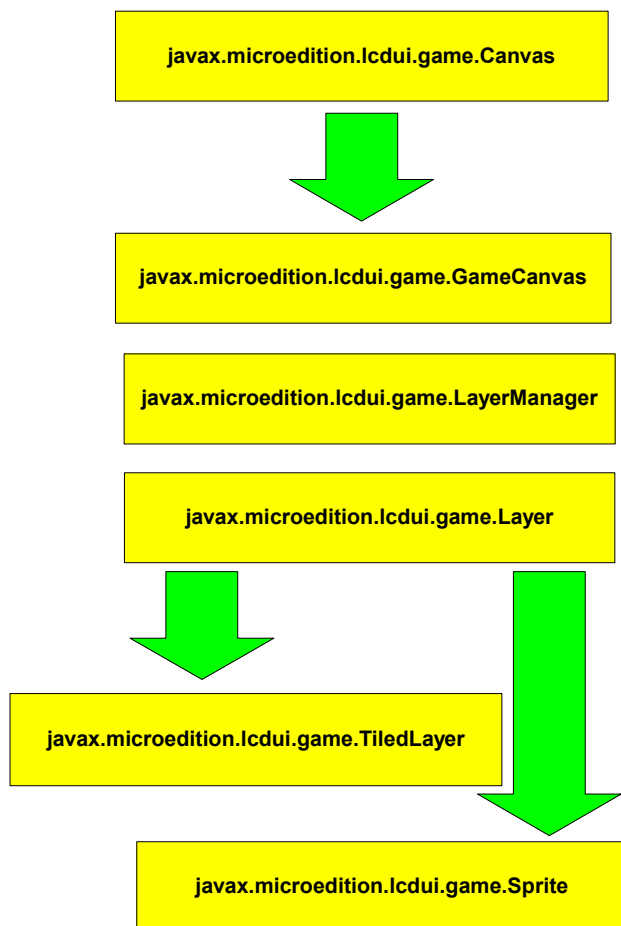
Richard Evers, Editor

In this article details will be provided on how to use the MIDP 2.0 gaming package, `javax.microedition.lcdui.game`. Source code will also be provided for a basic Java™ ME game that's a decent example of what's possible with very little time and coding effort. It was constructed to be easy to alter without focusing on optimization.

There are six classes in the MIDP 2.0 gaming package:

- Canvas
- GameCanvas
- LayerManager
- Layer
- TiledLayer
- Sprite

You will also need to use the `java.lang.Object.Thread` class to make any game you develop work.



## The Canvas and GameCanvas Classes

The `GameCanvas` class inherits from `Canvas` and then adds functionality to make gaming as easy as possible. The most useful addition is a dedicated offscreen buffer that is used to pre-create screens before display. I'll review the `Canvas` class first, then delve into `GameCanvas`.

### `javax.microedition.lcdui.Canvas`

`Canvas` is a base class that is used to write software that handles low-level events and makes graphic calls. It is interchangeable with standard `Screen` classes which makes it possible to use `Canvas` instead of higher-level screens.

Common game action fields are provided:

UP	DOWN	LEFT	RIGHT	FIRE
----	------	------	-------	------

BlackBerry® devices traditionally use '2' for UP, '8' for DOWN, '4' for LEFT, '6' for RIGHT and SPACE for FIRE.

General purpose key fields are also provided:

GAME_A	GAME_B	GAME_C	GAME_D
--------	--------	--------	--------

Call `getKeyCode()` to determine the physical key that corresponds to each game action field and `GAME_?` field.

Known key fields are also provided:

KEY_NUM0 - set to '0'	KEY_NUM6 - set to '6'
KEY_NUM1 - set to '1'	KEY_NUM7 - set to '7'
KEY_NUM2 - set to '2'	KEY_NUM8 - set to '8'
KEY_NUM3 - set to '3'	KEY_NUM9 - set to '9'
KEY_NUM4 - set to '4'	KEY_STAR - set to '*'
KEY_NUM5 - set to '5'	KEY_POUND - set to '#'

### Methods

`Canvas` has a single constructor that does not take an argument. The following is a detailed discussion of each method.

```
int getGameAction  
(int keyCode)
```

Returns the game action associated with the passed device `keyCode`, or returns zero if no action is associated with the `keyCode`. Returned game actions can be UP, DOWN, LEFT, RIGHT, FIRE, GAME\_A, GAME\_B, GAME\_C or GAME\_D. This method throws `IllegalArgumentException` if the passed `keyCode` is invalid.

```
int getKeyCode  
(int gameAction)
```

Returns the device-specific key value for the passed game action key code. Valid game actions can be UP, DOWN, LEFT, RIGHT, FIRE, GAME\_A, GAME\_B, GAME\_C or GAME\_D. This method throws `IllegalArgumentException` if the passed `gameAction` is invalid.

```
String getKeyName  
(int keyCode)
```

Returns a text string that describes a key. This usually matches the text printed on the key. The returned `String` can be displayed to aid users. In general, you'll get back text identifiers for UP, DOWN, LEFT, RIGHT and FIRE, plus keys such as ENTER, ESCAPE, SPACE, BACKSPACE and DELETE depending on mapping. If no text identifier exists, you'll get back an ASCII representation, such as '9'.

To get the key name of game action codes such as GAME\_A, GAME\_B, etc., try this:

```
String gameA = getKeyName(getKeyCode(GAME_A));
```

This method throws `IllegalArgumentException` if the passed `keyCode` is invalid.

```
boolean hasPointerEvents()
```

Returns true if the device supports pointer press and release events. Currently returns false on all BlackBerry devices.

```
boolean hasPointerMotionEvents()
```

Returns true if the platform supports pointer motion events (pointer dragged). Currently returns false on all BlackBerry® devices.

```
boolean hasRepeatEvents()
```

Returns true if the platform can generate repeat events when a key is held down. Currently returns false on all BlackBerry devices since trackwheel roll events don't repeat.

```
protected void hideNotify()
```

This method is always called after the Canvas has been removed from the display. Note that the BlackBerry JavaDocs states that calling `hideNotify()` will stop all calls to `key`, `pointer`, `paint`, and `commandAction()` methods, and that calls can be restarted by calling `showNotify()`. In reality, the default implementations of this method and `showNotify()` are empty and are also not overridden in the `GameCanvas` class.

```
boolean isDoubleBuffered()
```

Returns true if graphics is double buffered. Returns true on BlackBerry devices.

```
protected void keyPressed(int keyCode)
```

Automatically called when a key is pressed. Canvas has an empty implementation of this method and `GameCanvas` does not redefine it.

```
protected void keyReleased(int keyCode)
```

Automatically called when a key is released. Canvas has an empty implementation of this method and `GameCanvas` does not redefine it.

```
protected void keyRepeated(int keyCode)
```

Called when a key is repeated (held down). Canvas has an empty implementation of this method and `GameCanvas` does not redefine it.

```
protected abstract void paint  
(Graphics g)
```

Renders the Canvas where the `Graphic` object's clip region defines the area of the screen that is considered to be invalid. Your application must implement this method to paint any graphics.

The `Graphics` object that is passed to the `paint()` method has the following properties:

- the destination is the actual display, or if double buffering is in effect, a buffer for the display
- the clip region that includes at least one pixel within this Canvas
- the current color, which is set to black
- the font is the same as the font that is returned by `Font.getDefaultFont()`
- the stroke style, which is set to SOLID
- the origin of the coordinate system is located at the upper-left corner of the Canvas
- the Canvas is visible -- `isShown()` will return true

```
protected void pointerDragged  
(int x, int y)
```

Called when the pointer has been dragged. Make sure to call `hasPointerMotionEvents()` to determine if pointer events are supported. Canvas has an empty implementation of this method because pointer events are not currently supported on BlackBerry devices.

```
protected void pointerPressed  
(int x, int y)
```

Called when the pointer is pressed. Canvas has an empty implementation of this method because pointer events are not currently supported on BlackBerry devices.

```
protected void pointerReleased  
(int x, int y)
```

Called when the pointer is released. Canvas has an empty implementation of this method because pointer events are not currently supported on BlackBerry devices.

```
void repaint()
```

Requests a repaint for the entire Canvas. Calling `repaint(0, 0, getWidth(), getHeight());` will achieve the same result.

```
void repaint
(int x, int y, int width, int height)
```

Requests a repaint for the specified region of the Screen where 'x' and 'y' are the top-left coordinates to start repainting. Note that `repaint()` will not block while `paint()` is occurring, and calling `repaint()` can trigger a later call to `paint()`.

```
void serviceRepaints()
```

Forces any pending `repaint()` requests to be serviced immediately. This method blocks until the pending requests have been serviced. If there are no pending repaints, or if this canvas is not visible, it will do nothing and return immediately.

---

*Note: This method blocks until the call to `paint()` returns. Your application has no control over which thread calls `paint()`. If the caller of `serviceRepaints()` holds a lock that the `paint()` method acquires, this may result in deadlock. Callers of `serviceRepaints()` must not hold any locks that might be acquired within the `paint()` method.*

*The `Display.callSerially()` method provides a way to call back an application after painting has been completed, avoiding the danger of deadlock.*

---

```
void setFullScreenMode
(boolean mode)
```

Controls whether the Canvas is in full-screen mode (true) or in normal screen mode (false). The title and separator fields (if they exist) are removed from the vertical field manager in full screen mode, and are restored in normal screen mode.

```
protected void showNotify()
```

This method is always called before the Canvas is made visible. The default implementations of this method and the `hideNotify()` method are empty and are not overridden in the `GameCanvas` class.

```
protected void sizeChanged
(int w, int h)
```

Called when the drawable area of the Canvas has been changed. The default implementation of this method is empty and is not overridden in the `GameCanvas` class.

---

## javax.microedition.lcdui.GameCanvas

---

`GameCanvas` extends `Canvas`, and provides game-specific functionality such as an automatic off-screen graphics buffer and the ability to poll key status. Use of an off-screen graphics buffer (also known as double buffering) stops display flickering by writing to the offscreen screen buffer before making it visible.

Public constants defined in this class are bit representations of Canvas game actions.

- `UP_PRESSED = 1 << Canvas.UP;`  
Result = 00000000 00000010 (0x0002)
- `LEFT_PRESSED = 1 << Canvas.LEFT;`  
Result = 00000000 00000100 (0x0004)

- `RIGHT_PRESSED = 1 << Canvas.RIGHT;`  
Result = 00000000 00100000 (0x0020)
- `DOWN_PRESSED = 1 << Canvas.DOWN;`  
Result = 00000000 01000000 (0x0040)
- `FIRE_PRESSED = 1 << Canvas.FIRE;`  
Result = 00000001 00000000 (0x0100)
- `GAME_A_PRESSED = 1 << Canvas.GAME_A;`  
Result = 00000010 00000000 (0x0200)
- `GAME_B_PRESSED = 1 << Canvas.GAME_B;`  
Result = 00000100 00000000 (0x0400)
- `GAME_C_PRESSED = 1 << Canvas.GAME_C;`  
Result = 00001000 00000000 (0x0800)
- `GAME_D_PRESSED = 1 << Canvas.GAME_D;`  
Result = 00010000 00000000 (0x1000)

---

## Methods

---

`GameCanvas` has a single constructor:

```
GameCanvas(boolean suppressKeyEvents)
```

Pass true if you only need to query the status of keys using `getKeyStates()`. This will improve performance slightly during execution. Pass false if you also need to call the `keyPressed()`, `keyRepeated()` and `keyReleased()` methods.

---

*Note: `showNotify()` and `hideNotify()` are not implemented in `Canvas` or `GameCanvas` therefore you cannot start and stop key suppression after object creation.*

---

The following is a detailed discussion of each method.

```
void flushGraphics()
```

Writes the off-screen buffer to the display. The contents of the off-screen buffer are not changed after write. This method returns after the flush completes, so you can render the next frame to the off-screen buffer when the method returns. This method does nothing and returns immediately if the `GameCanvas` is not on display or if the flush request cannot occur because the system is busy.

```
void flushGraphics
(int x, int y, int width, int height)
```

Writes the passed region of the off-screen buffer to the display. Only the intersecting region is written if the passed region extends beyond the bounds of the `GameCanvas`. Nothing is written if the passed width or height is less than 1.

This method returns after the flush completes, so you can render the next frame to the off-screen buffer when the method returns. This method does nothing and returns immediately if the `GameCanvas` is not on display or if the flush request cannot occur because the system is busy.

The parameters are x (left), y (top), width and height.

```
protected Graphics getGraphics()
```

Gets the off-screen buffer used to render a `GameCanvas`. Once rendered, call `flushGraphics()` to display the buffer. Call this method after your game starts to get an off-screen

buffer then reuse the buffer throughout your game. A newly created Graphics object has the following properties:

- destination is set to the off-screen buffer
- clip region is the entire buffer
- current color is set to black
- font is identical to Font.getDefaultFont();
- stroke style is SOLID
- origin of the coordinate system is the upper-left corner of the buffer

#### **int getKeyStates()**

Gets the state of the game keys. Each bit in the returned integer represents a specific key on the device. Each bit will be set if the corresponding key is currently down or has been pressed at least once since the last time this method was called. The bit will be 0 if the key is currently up and has not been pressed at all since the last time this method was called. This latching behavior ensures that a rapid key press and release will always be caught by the game loop, regardless of how slowly the loop runs.

For example:

```
// Get the key state and store it
int keyState = getKeyStates();
```

---

```
package com.blackberrydeveloperjournal.bone;

import javax.microedition.lcdui.*;
import javax.microedition.lcdui.game.*;
import java.util.Timer;

public class BoneCanvas extends javax.microedition.lcdui.game.GameCanvas {
    // adjust font, background and grass colors
    private static final int FONT_COLOR = 0x000000; // black
    private static final int BACK_COLOR = 0xCCFFFF; // light cyan
    private static final int TUFT_COLOR = 0x33FF33; // light green - same as Tuft sprite color

    // adjust to arrange where the points, countdown and game over messages will display
    private static final int LINE_POINTS = 1;
    private static final int LINE_COUNTDOWN = 2;
    private static final int LINE_OVER = 3;
    private static final int LINE_MAX = LINE_OVER;

    // adjust grass depth
    private static final int DEPTH = 25;

    // adjust game duration
    private static final int GAME_MINUTES = 4;

    // work constants
    public static int WIDTH;
    private static int HEIGHT;
    private static int HEIGHT_FONT;
    private static int WIDTH_POINTS, WIDTH_COUNTDOWN, WIDTH_EOG;
    private static int X_TOP, Y_TOP;
    private static Font CURRENT_FONT;
    private static int RIGHT = 1;
    private static int LEFT = 0;
    private int GAME_DURATION = GAME_MINUTES * 60;
```

```
if ((keyState & LEFT_KEY) != 0) {
    positionX--;
} else if ((keyState & RIGHT_KEY) != 0) {
    positionX++;
}
```

Calling this method clears any latched state. Another call to getKeyStates() immediately after a prior call will report the system's best idea of the current state of the keys, the latched bits having been cleared by the first call.

This method will return 0 if the GameCanvas is not visible and will return 0 when the screen first becomes visible. If any key is held down while the screen state goes from invisible to visible, the user must release the key then try again before it can be reported.

#### **void paint(Graphics g)**

Overrides Canvas.paint() and renders the off-screen buffer at top left (0,0). Rendering of the screen buffer is subject to the clip region and the origin translation of the Graphics object. This method throws NullPointerException if the passed Graphics object is null

The following are two classes from the Bone game that show how to use Canvas.

```

private int gameCountdown = GAME_DURATION;
private int playerScore = 0;
private boolean gameIsOver;
private String countdownBuffer, countdownLabel, pointsLabel, eogLabel;

private BoneManager hManager;
private Bone        hMIDlet;
private BoneTime    hBoneTime;

public BoneCanvas( Bone obj ) throws Exception {
    super( false );
    hMIDlet = obj;
    X_TOP = Y_TOP = 0;
    WIDTH = getWidth();
    HEIGHT = getHeight();
    CURRENT_FONT = getGraphics().getFont();
    HEIGHT_FONT = CURRENT_FONT.getHeight();

    countdownLabel = "Countdown: ";
    WIDTH_COUNTDOWN = CURRENT_FONT.stringWidth( countdownLabel + "0:00" );

    pointsLabel = "Points: ";
    WIDTH_POINTS = CURRENT_FONT.stringWidth( pointsLabel + "0000" );

    eogLabel = "End of Game";
    WIDTH_EOG = CURRENT_FONT.stringWidth( eogLabel );

    if ( hManager == null ) {
        int yStart = Y_TOP + (LINE_MAX*HEIGHT_FONT);
        int yEnd = HEIGHT - (LINE_MAX*HEIGHT_FONT) - DEPTH;

        gameIsOver = false;

        // set a timer to go off every second for accurate time tracking
        hBoneTime = new BoneTime( GAME_DURATION );
        new Timer().schedule( hBoneTime, 0, 1000 );

        hManager = new BoneManager( X_TOP, yStart, WIDTH, yEnd );
    }
}

// overrides Canvas.paint()
public void paint( Graphics g ) {
    // paint the background
    g.setColor( BACK_COLOR );
    g.fillRect( X_TOP, Y_TOP, WIDTH, HEIGHT );

    // paint the grass
    g.setColor( TUFT_COLOR );
    g.fillRect( X_TOP, Y_TOP + HEIGHT - DEPTH, WIDTH, HEIGHT );

    // paint the layer manager
    try {
        hManager.paint( g );
    } catch( Exception err ) {
        hMIDlet.showException( err );
    }

    // determine current time
    if ( getTimeLeft() != gameCountdown ) {
        gameCountdown = getTimeLeft();
    }
}

```

```

int gameNow = getTimeLeft();
int seconds = gameCountdown % 60;

countdownBuffer = (gameCountdown / 60) + ":";

if ( seconds < 10 )
    countdownBuffer += "0" + seconds;
else
    countdownBuffer += seconds;
}

// draw the time remaining in the game and points scored
g.setFont( CURRENT_FONT );
g.setColor( FONT_COLOR );
g.drawString( countdownLabel + countdownBuffer, (WIDTH - WIDTH_COUNTDOWN)/2,
    Y_TOP + (LINE_COUNTDOWN*HEIGHT_FONT), g.LEFT | g.TOP );
g.drawString( pointsLabel + playerScore, (WIDTH - WIDTH_POINTS)/2,
    Y_TOP + (LINE_POINTS*HEIGHT_FONT), g.LEFT | g.TOP );

if ( gameIsOver ) {
    hMIDlet.setMenuStart();

    // clear the time region
    g.setColor( BACK_COLOR );
    g.fillRect( X_TOP, Y_TOP + (LINE_COUNTDOWN*HEIGHT_FONT), WIDTH, HEIGHT_FONT + 1 );

    // draw end of game message
    g.setColor( FONT_COLOR );
    g.setFont( CURRENT_FONT );
    g.drawString( eogLabel, (WIDTH - WIDTH_EOG)/2, Y_TOP + (LINE_OVER*HEIGHT_FONT),
        g.LEFT | g.TOP );
}
}

// returns time remaining until game ends
// also sets gameIsOver flag when time runs out
private int getTimeLeft() {
    int timeLeft = hBoneTime.getRemainingTime();

    if ( timeLeft == 0 )
        gameIsOver = true;

    return timeLeft;
}

private void gameStop() { gameIsOver = true; }
private void gameRun() { gameIsOver = false; }

public void canvasBegin() {
    gameRun();
    Display hDisplay = Display.getDisplay( hMIDlet );
    // make current, visible and in focus
    hDisplay.setCurrent( this );
    repaint();
    hDisplay = null;
}

public void canvasInitialize() {
    hManager.restoreToDefault();
    gameRun();
}

```

```

playerScore = 0;
gameCountdown = GAME_DURATION;
repaint();
}

public void canvasRefresh() {
    if ( !gameIsOver ) {
        int keyState = getKeyStates();

        // 6 on 87xx, 72xx and 71xx BlackBerry devices
        if ( ( keyState & LEFT_PRESSED ) > 0 )
            hManager.setDirection( LEFT );

        // 4 on 87xx, 72xx and 71xx BlackBerry devices
        if ( ( keyState & RIGHT_PRESSED ) > 0 )
            hManager.setDirection( RIGHT );

        // 2 on 87xx, 72xx and 71xx BlackBerry devices
        if ( ( keyState & UP_PRESSED ) > 0 )
            hManager.walkerHop();

        playerScore += hManager.movement();

        try {
            paint( getGraphics() );
            // display offscreen buffer
            flushGraphics();
        } catch(Exception err) {
            hMIDlet.showException(err);
        }

        // end the game, allow user to restart via menu or quit entirely
        if ( gameCountdown == 0 ) {
            gameStop();
            hMIDlet.pauseApp();
        }
    }
}
}

```

---

```

package com.blackberrydeveloperjournal.bone;

```

```

import java.util.TimerTask;

```

```

public class BoneTime extends TimerTask {
    private int remainingTime;

    public BoneTime( int timeLimit ) {
        remainingTime = timeLimit;
    }

    // called each time the timer is triggered
    public void run() {
        --remainingTime;
    }

    public int getRemainingTime() {
        return remainingTime;
    }
}

```

---

The BoneTime class is used by the BoneCanvas class to keep accurate track of elapsed game time. It's initialized by passing the time limit in seconds, and is called every second thereafter to decrement the time limit.

The BoneCanvas class draws and updates the screen, keeps track of the game duration, and handles keys. It was designed so you can easily change the color of the text, background and grass, the regions where game points, remaining time and the game over message are displayed, the depth of the grass, and duration of the game.

The constructor determines the width and height of the display, the height of the current font, sets up the game points, countdown and end of game strings, starts the game timer and then calls BoneManager to create and handle Sprites.

---

### **javax.microedition.lcdui.LayerManager**

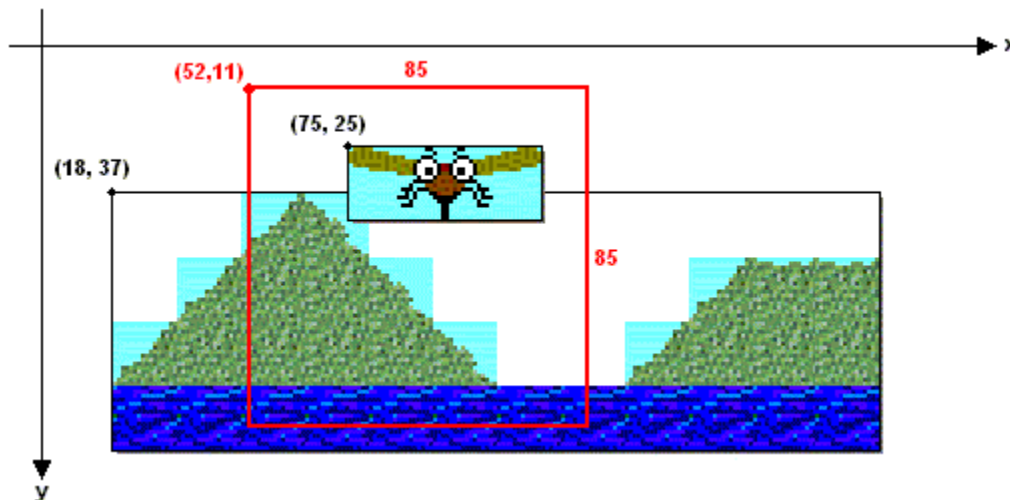
---

The LayerManager simplifies rendering of Layers that have been added to it by automatically rendering the correct regions of each Layer in the appropriate order.

The LayerManager maintains an ordered list to which Layers can be added, inserted and removed. A Layer at index 0 is closest to the user while a Layer with the highest index is furthest away from the user. If a Layer is removed, the indices of subsequent Layers will be adjusted to maintain continuity.

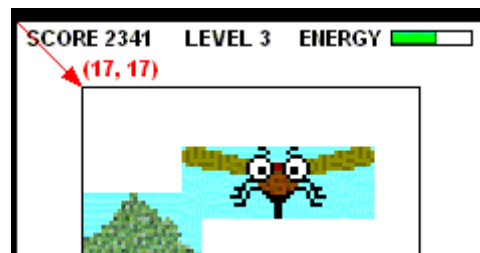
The view window controls the size of the visible region and its position relative to the LayerManager's coordinate system. Changing the position of the view window enables effects such as scrolling or panning of the user's view. For example, to scroll to the right, simply move the view window's location to the right. The size of the view window controls how large the user's view will be, and is usually fixed at a size that is appropriate for the device's screen.

In the following example, the view window is set to 85 x 85 pixels and is located at (52, 11) in the LayerManager's coordinate system. The Layers appear at their respective positions relative to the LayerManager's origin.



The paint() method includes an (x,y) location that controls where the view window is rendered relative to the screen. Changing these parameters does not change the contents of the view window - it simply changes the location where the view window is drawn. Note that this location is relative to the origin of the Graphics object, and is subject to the translation attributes of the Graphics object.

For example, if a game uses the top of the screen to display the current score, the view window may be rendered at (17, 17) to provide enough space for the score.



Note that the LayerManager() constructor does not take arguments.

---

## Methods

---

### **void append (Layer l)**

Appends a Layer to the list of existing Layers such that it has the highest index that is furthest from the user. The Layer is first removed from this LayerManager if it has already been added. Pass the Layer to be added when calling this function. This method will throw NullPointerException if the passed Layer is null.

### **Layer getLayerAt (int index)**

Returns the Layer at the passed index location. This method will throw IndexOutOfBoundsException if the passed index is less than zero, or if it is equal to or greater than the number of Layers added to this LayerManager.

### **int getSize()**

Returns the number of Layers in this LayerManager.

### **void insert (Layer l, int index)**

Inserts the passed Layer in this LayerManager at the passed index position.

This method throws:

- NullPointerException  
The passed Layer is null
- IndexOutOfBoundsException  
The passed index is less than 0 or greater than the number of Layers already added to this LayerManager

### **void paint (Graphics g, int x, int y)**

Renders the LayerManager's current view window at the passed location using the passed Graphics object.

LayerManager renders each layer in order of descending index. Layers that are completely outside of the view window are not rendered. The passed coordinates are used to render the LayerManager's view window relative to the origin of the passed Graphics object. For example, a game may use the top of the screen to display the current score. The view window could be rendered at (0, 20) to render the game's layers below that area. This location is relative to the Graphics object's origin, so translating the Graphics object will change where the view window is rendered on the screen.

The Graphics object clip region is intersected with a region having the same dimensions as the view window and located at (x,y). The LayerManager translates the graphics object such that the point (x,y) corresponds to the location of the

viewWindow in the coordinate system of the LayerManager. The Layers are then rendered in the appropriate order. The translation and clip region of the Graphics object are restored to their prior values before this method returns.

Rendering is subject to the clip region and translation of the Graphics object. Thus, only part of the passed view window may be rendered if the clip region is not large enough.

This method may ignore Layers that are invisible or that would be rendered entirely outside of the Graphics object's clip region to improve performance. The attributes of the Graphics object are not restored to a known state between calls to the Layers' paint methods. The clip region may extend beyond the bounds of a Layer.

To call this method, pass the graphics instance with which to draw the LayerManager and the horizontal and vertical locations to render the view window relative to the Graphics' translated origin.

This method will throw NullPointerException if the passed Graphics object is null.

### **void remove (Layer l)**

Removes the passed Layer from this LayerManager. This method does nothing if the passed Layer has not been previously added to this LayerManager.

This method will throw NullPointerException if the passed Layer is null.

### **void setViewWindow (int x, int y, int width, int height)**

Sets the view window on the LayerManager. The view window is the region that the LayerManager draws when its paint() method is called. This allows the developer to control the size of the visible region, as well as the location of the view window relative to the LayerManager's coordinate system.

The view window stays constant until it is modified by another call to this method. By default, the view window is located at (0,0) in the LayerManager's coordinate system and its width and height are both set to Integer.MAX\_VALUE.

To call this method, pass the horizontal and vertical locations of the view window relative to the LayerManager's origin, and the width and height of the view window.

This method throws IllegalArgumentException if the width or height is less than 0.

The following is the BoneManager class of the Bone game. It is derived from LayerManager.

---

```
package com.blackberrydeveloperjournal.bone;

import javax.microedition.lcdui.*;
import javax.microedition.lcdui.game.*;

public class BoneManager extends javax.microedition.lcdui.game.LayerManager {
```

```

public static int HEIGHT;
private static int WIDTH;
private static int X_TOP, Y_TOP;
private static int RIGHT = 1;
private static int LEFT = 0;
private int travelDirection = RIGHT;

// adjust total number of hurtling Sprites (left & right)
private static int MAX_MISSILES = 8;

private Walk walker;
private Missile[] missileRight;
private Missile[] missileLeft;
private Tuft tufts;

private int offsetTL, defTLOffset;

public BoneManager( int xPos, int yPos, int width, int height ) throws Exception {
    X_TOP = xPos;
    Y_TOP = yPos;
    WIDTH = width;
    HEIGHT = height;

    // set view window for entire screen
    setViewWindow( 0, 0, WIDTH, HEIGHT );

    offsetTL = defTLOffset = tufts.WIDTH;

    // create a walking Sprite
    if ( walker == null ) {
        walker = new Walk( offsetTL + WIDTH/2, HEIGHT - Walk.HEIGHT - 1 );
        append( walker );
    }

    // create RIGHT sprites that will hurtle towards the stick man!
    if ( missileRight == null ) {
        missileRight = new Missile[ MAX_MISSILES/2 ];

        for ( int element=missileRight.length-1; element >= 0; --element ) {
            missileRight[ element ] = new Missile( RIGHT );
            append( missileRight[ element ] );
        }
    }

    // create LEFT sprites that will hurtle towards the stick man!
    if ( missileLeft == null ) {
        missileLeft = new Missile[ MAX_MISSILES/2 ];

        for ( int element=missileLeft.length-1; element >= 0; --element ) {
            missileLeft[ element ] = new Missile( LEFT );
            append( missileLeft[ element ] );
        }
    }

    // create animated background
    if ( tufts == null ) {
        tufts = new Tuft();
        append( tufts );
    }
}

```

```

// set walker's direction: RIGHT or LEFT
public void setDirection( int direction ) {
    travelDirection = direction;
}

// make walker hop
public void walkerHop() {
    walker.doHop();
}

// reset game to default settings
public void restoreToDefault() {
    offsetTL = defTLOffset;
    travelDirection = RIGHT;

    if ( walker != null )
        walker.defaultState();

    if ( tufts != null )
        tufts.defaultState();

    if ( missileRight != null )
        for ( int element=missileRight.length-1; element >= 0; --element )
            missileRight[ element ].defaultState();

    if ( missileLeft != null )
        for ( int element=missileLeft.length-1; element >=0; --element )
            missileLeft[ element ].defaultState();
}

public void paint( Graphics g ) {
    setViewWindow( offsetTL, 0, WIDTH, HEIGHT );
    paint( g, X_TOP, Y_TOP );
}

// advance all moving objects
public int movement() {
    tufts.movement();
    walker.movement( travelDirection );
    offsetTL = ( travelDirection == RIGHT ) ? offsetTL+1 : offsetTL-1;
    int score = getScore( missileRight ) + getScore( missileLeft );
    handleMove();
    return score;
}

// calculate score of missiles hitting walker versus walker jumping over missiles
private int getScore( Missile[] missile ) {
    int score = 0;
    for ( int element=missile.length-1; element >= 0; --element ) {
        score += missile[ element ].movement( walker, offsetTL, offsetTL + WIDTH );
        score -= walker.isCollision( missile[ element ] );
    }
    return score;
}

private void handleMove() {
    // update after passing a single animated Tuft object
    if ( offsetTL % defTLOffset == 0 ) {
        int width = defTLOffset;

        if ( travelDirection == RIGHT )

```

```

        width = -defTLOffset;

        // Sprite inherits from javax.microedition.lcdui.game.Layer.move()
        walker.move( width, 0 );
        offsetTL += width;

        moveMissile( missileLeft, width );
        moveMissile( missileRight, width );
    }
}

private void moveMissile( Missile[] missile, int width ) {
    for ( int element=missile.length-1; element >=0; --element )
        missile[ element ].move( width, 0 );
}
}

```

The BoneManager class creates and maintains a Walk Sprite in the form of a stick man that walks in a stationary position, hops in the “air” and changes direction. It also creates and maintains simplistic Missile Sprites that hurtle towards the stick man. Points are earned or lost when the stick man either misses a Missile, or is hit by a Missile. A user-defined constant, MAX\_MISSILES, has been provided for you to adjust the total number of Missile Sprites in the game. Note that this constant value will split evenly to create Missiles that hurtle from the left and right sides.

There should be no major surprises in the BoneManager constructor. When called, the x and y offsets, plus the width and height, are passed. The width and height are used to set the view window to the entire screen.

A single Walk Sprite is created in the mid-point of the screen, with a minuscule offset of the width of an animated tuft of grass from the Tuft class. The vertical positioning is set by the height of the screen minus the height of the Walk Sprite image.

The right and left Missile Sprites are created next, followed by an animated TiledLayer Tuft of grass.

There are a few public and private methods that are worth mentioning.

setDirection() is called to set the direction that the Walk Sprite is travelling. Pass RIGHT (1) or LEFT(0).

walkerHop() is called to force the Walk Sprite to hop. You can adjust the height of the hop in the Walk class.

restoreToDefault() is called to reset the Sprites and values used in the game to their default settings.

movement() is called to advance all “moving” objects. This includes the animated Turf TiledLayer, the Walk Sprite, and the Missile Sprites. Note that handleMove() takes care of Sprite movement.

handleMove() only acts when the current x offset can be evenly divided by the width of an animated Turf object. If so, then the Walk object is “moved” by the width of a Turf object, then the right and left Missile Sprites are moved by the same distance by calling moveMissile().

getScore() is called to calculate the total number of Missiles avoided and the total number of Walk collisions that were detected.

---

### **javax.microedition.lcdui.Layer**

---

Layer is an abstract class used to represent a visual element within a game. Each Layer has the top-left position, width, height and visibility status. The Layer’s top-left co-ordinates (x,y) are always set relative to the origin of the Graphics object passed to Layer’s abstract paint() method, which must be implemented within a subclass. Note that paint() is the only abstract method within this class. All of the other methods and the constructor have been implemented.

---

### **Methods**

---

Layer has a single constructor that takes two arguments to set the internal width and height variables.

```
Layer(int width, int height)
```

```
int getHeight()
```

Returns the height of this layer in pixels.

```
int getWidth()
```

Returns the width of this layer in pixels.

```
int getX()
```

Returns the left-most upper corner position of this Layer in the painter's coordinate system.

```
int getY()
```

Returns the top corner position of this in the painter's coordinate system.

```
boolean isVisible()
```

Returns true if the Layer is visible else returns false.

```
void move(int dx, int dy)
```

Updates the upper-left origin of the Layer by adding “dx” pixels to the left coordinate and “dy” pixels to the top coordinate where either parameter can be a positive or a negative

value. The Layer's coordinates will wrap if the passed values would lead to either the x coordinate or y coordinate to exceed Integer.MAX\_VALUE or drop below Integer.MIN\_VALUE.

**abstract void paint(Graphics g)**

Abstract method to paint this Layer if it is visible. Note that a subclass must override and implement this method.

**void setPosition(int x, int y)**

Changes the upper-left origin of the Layer to the passed coordinates in the painter's coordinate system.

**void setVisible(boolean visible)**

Pass true to make the Layer visible, and pass false to make the Layer invisible.

---

### javax.microedition.lcdui.TiledLayer

---

Layer is an abstract class used to represent a visual element within a game. Each Layer has the top-left position, width, height and visibility status. The Layer's top-left co-ordinates (x,y) are always set relative to the origin of the Graphics object passed to Layer's abstract paint() method, which must be implemented within a subclass. Note that paint() is the only abstract method within this class. All of the other methods and the constructor have been implemented.

---

### Methods

---

TiledLayer has a single constructor that takes five arguments.

```
TiledLayer(
    int columns,
    int rows,
    Image image,
    int tileWidth,
    int tileHeight
)
```

"columns" is the width of the TiledLayer expressed in the number of "tileWidth" cells.

"rows" is the height of the TiledLayer expressed in the number of "tileHeight" cells.

"image" is the image used to create the static tile set.

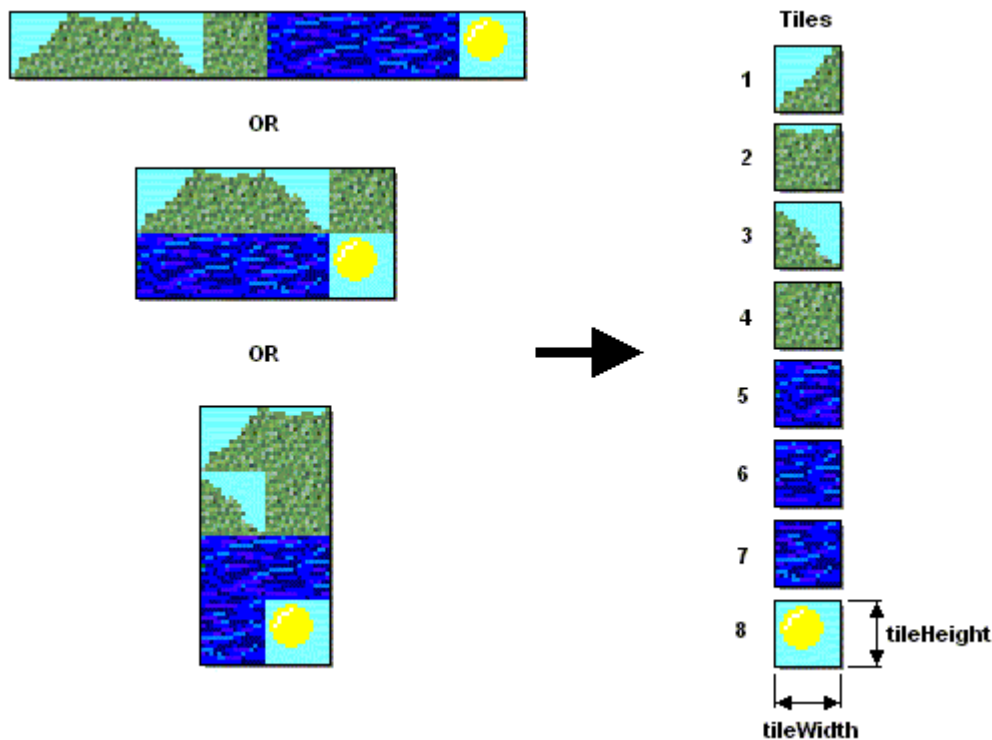
The final parameters, "tileWidth" and "tileHeight", are the width and height, in pixels, for one tile.

Here's the syntax in basic form:

```
Image myImage = Image.createImage(
    "MyImage.png");
TiledLayer myLayer = new TiledLayer(
    (20, 10, myImage, 200, 100);
```

The TiledLayer class is used to work with images that can be used to construct backgrounds and virtual gaming areas through use and reuse of static "tiles." For example, using ladders, platforms and more. The trick is to pass an image to the constructor that includes all background components that you'll need, organized into equal-sized cells that can be mathematically determined.

For example:



The graphic image used to create each individual tile can take several forms as shown on the left, where the static tiles will end up being referenced by cell identifier. as shown on the right. Note that index referencing of static tiles starts at 1.

Placing of static tiles on the display is handled by setting cells in the grid to use a tile at a specific starting offset on the screen. All cells in the screen grid are initially empty (set to 0), and can be modified by using `setCell()` and `fillCells()`.

Each static tile will have the dimensions of `tileWidth` and `tileHeight`. The source image width must be an integer multiple of `tileWidth`, and the source height must be an integer multiple of `tileHeight`. `IllegalArgumentException` will be thrown if either condition is not met.

The static tile set can be changed using `setStaticTileSet()`. Use of this method will come at a cost of time and increased memory use. You can avoid reliance on this method by using animated tiles instead of individual static tiles through the use of `createAnimatedTile()`. This method returns a negative index offset.

This method throws:

- `NullPointerException`
  - Image is null
- `IllegalArgumentException`
  - The number of rows or columns is less than 1
  - `tileHeight` or `tileWidth` is less than 1
  - Image width is not an integer multiple of the `tileWidth`
  - Image height is not an integer multiple of the `tileHeight`

```
int createAnimatedTile  
(int staticTileIndex)
```

Creates a new animated tile within a set and returns the index as a negative value (-1, then -2, -3, etc.). The animated tile can be associated with a blank area (index 0) or a static tile with a positive index value.

This method will throw `IndexOutOfBoundsException` if the `staticTileIndex` is invalid.

```
void fillCells  
(int col, int row, int numCols,  
int numRows, int tileIndex)
```

Sets a series of screen cells with the passed static tile index, animated tile index, or a blank tile using index 0. You also need to pass the start column and row of the top-left cell in the region, the number of columns and rows in the region along with the index of the tile to place in all cells in the region.

This method throws:

- `IndexOutOfBoundsException`
  - The passed rectangular region extends beyond the bounds of the `TiledLayer` grid
  - No tile is associated with index `tileIndex`

- `IllegalArgumentException`
  - `numCols` is less than zero
  - `numRows` is less than zero

```
int getAnimatedTile  
(int animatedTileIndex)
```

Returns the tile index referenced by the passed index of an animated tile. This method will throw `IndexOutOfBoundsException` if the passed animated tile index is invalid.

```
int getCell  
(int col, int row)
```

Pass the column and row of the cell to check to get the index value of the static or animated tile in a screen cell. An index value of 0 will be returned if the cell is empty. This method throws `IndexOutOfBoundsException` if the passed row or col is outside the bounds of the `TiledLayer` grid.

```
int getCellHeight()
```

Returns the pixel height of a cell in the `TiledLayer` grid.

```
int getCellWidth()
```

Returns the pixel width of a cell in the `TiledLayer` grid.

```
int getColumns()
```

Returns the number of columns in the `TiledLayer` grid. Call `Layer.getWidth()` to get the overall width of the `TiledLayer`, in pixels.

```
int getRows()
```

Returns the number of rows within the `TiledLayer` grid. Call `Layer.getHeight()` to get the overall height of the `TiledLayer`, in pixels.

```
void paint  
(Graphics g)
```

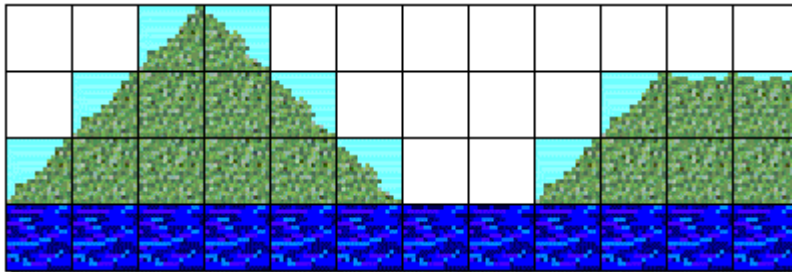
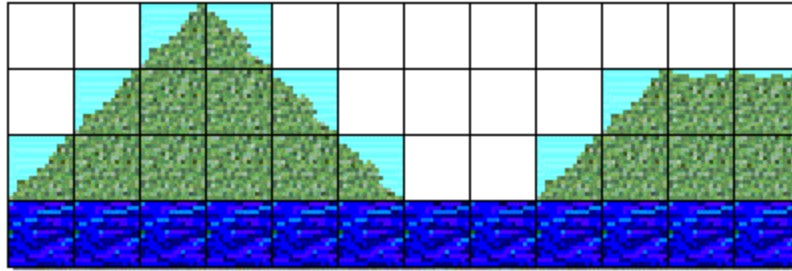
Draws the entire `TiledLayer` subject to the clip region of the `Graphics` object. The upper left corner is rendered at the `TiledLayer`'s current position relative to the origin of the `Graphics` object. The current position of the upper-left corner can be retrieved by calling `Layer.getX()` and `Layer.getY()`. If the `TiledLayer`'s `Image` is subject to change, then `TiledLayer` is rendered using the current contents of the `Image`. This method throws `NullPointerException` if the passed `Graphics` object is null.

Note that this method overrides `Layer.paint()`.

```
void setAnimatedTile  
(int animatedTileIndex,  
int staticTileIndex)
```

Sets a direct relationship between an animated tile and a static tile. This allows you to animate a section of the display grid through a simple change.

For example, compare the water in the following two images to see how animation can occur by calling `setAnimatedTile()` to change the static tile associated with a specific animated tile on display.



As shown, the water region is filled with an animated tile having an index of -1 which had been initially associated with static tile 5. The entire water area is animated by changing the associated static tile by using `setAnimatedTile(-1, 7)`.

In this example, the display grid could look like this:

0	0	1	3	0	0	0	0	0	0	0	0
0	1	4	4	3	0	0	0	0	1	2	2
1	4	4	4	4	3	0	0	1	4	4	4
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

Note that the parameter, `staticTileIndex`, must either be the index of a static tile, or 0 if a blank tile is to be used.

This method throws:

- `IndexOutOfBoundsException`
  - The `staticTileIndex` is invalid
- `IndexOutOfBoundsException`
  - The animated tile index is invalid

```
void setCell
(int col, int row, int tileIndex)
```

Sets the contents of a display cell to a static tile index or an animated tile index. It may also be left empty by passing an index of 0. Pass the column and row within the grid to set, and the index value.

This method throws:

- `IndexOutOfBoundsException`
  - There is no tile with index `tileIndex`
  - `row` or `col` is outside the bounds of the `TiledLayer` grid

```
void setStaticTileSet
(Image image, int tileWidth,
int tileHeight)
```

Replaces the current static tile set with a new static tile set. If the new static tile set has as many or more tiles than the previous static tile set, then the animated tiles and cell contents will be preserved. If not, then the contents of the grid will be cleared (all cells set to index 0) and all animated tiles will be deleted. Use this method sparingly since it can be memory and time consuming. Where possible, use animated tiles to animate tile appearance rather than change the tile set.

Pass the `Image` to use for creating the static tile set, and the width and height in pixels of a single tile.

This method throws:

- `NullPointerException`
  - Passed `Image` is null
- `IllegalArgumentException`
  - `tileHeight` or `tileWidth` is less than 1
  - `Image` width is not an integer multiple of the `tileWidth`
  - `Image` height is not an integer multiple of the `tileHeight`

```
package com.blackberrydeveloperjournal.bone;
```

```
import javax.microedition.lcdui.*;
import javax.microedition.lcdui.game.*;
```

```

// Draw background objects to give the appearance of movement
public class Tuft extends TiledLayer {
    // width & height of a single frame within the source image
    public static final int WIDTH = 16;
    private static final int HEIGHT = 16;
    // number of frames within the source image
    private static final int FRAMES = 4;
    // size of tile array where element 0 is a blank tile
    private static final int TILES = 2;
    // order to play the animated tile
    private static final int[] PLAY_ORDER = { 2, 3, 2, 4, 2, 5 };
    // number of elements in array above
    private static final int PLAY_SIZE = 5;
    // bitwise test to reduce polling period
    // set alternate patterns to increase delay
    private static final int POLLING_PERIOD = 5; // binary 0000101

    private static int COLUMNS;
    private static int X_UPPER = 0;
    private static int Y_UPPER = BoneManager.HEIGHT - HEIGHT;
    private int index = 0;
    private int thisTile = 0;
    private int delayCounter = 0;

    public Tuft() throws Exception {
        // Note: calcColumns() sets COLUMN
        super( calcColumns( BoneCanvas.WIDTH ), 1, Image.createImage( "img/tuft.png" ), WIDTH, HEIGHT );

        // set Layer's position
        setPosition( X_UPPER, Y_UPPER );

        // create a new animated tile
        // returns the index that refers to the new animated tile
        // note that referencing begins at 1 where 0 is a empty tile
        index = createAnimatedTile( TILES );

        // blanket the ground with grass
        for ( int element = COLUMNS-1; element >= 0; --element )
            setCell( element, 0, index );
    }

    // used in call to super() where nothing can be done before making the call
    private static int calcColumns( int screenWidth ) {
        COLUMNS = ((screenWidth / WIDTH) + 1)*(FRAMES-1);
        return COLUMNS;
    }

    // adjust to provide screen "movement"
    public void movement() {
        // decrease animation speed slightly
        if ((delayCounter++ & POLLING_PERIOD) == POLLING_PERIOD ) {
            // animate grass
            // remove animation to increase speed
            setAnimatedTile( index, PLAY_ORDER[thisTile++] );

            if ( thisTile > PLAY_SIZE )
                thisTile = 0;
        }
    }
}

```

```

// restore to default state
public void defaultState() {
    setPosition( X_UPPER, Y_UPPER );
    thisTile = delayCounter = 0;
    setAnimatedTile( index, PLAY_ORDER[thisTile] );
}
}

```

The Turf class handles an animated TiledLayer object that resembles blades of grass. This class was created to lay down a dense layer of animated grass, which is updated on a regular basis to switch image frames to generate an illusion of directional movement on the screen.

The class opens up with a series of constants that can be altered as required.

The WIDTH and HEIGHT are set to the x and y dimensions of each frame within the source image. In this case A 16 x 16 frame was used with 4 FRAMES in the image.

The TILES constant, set to 2, is a bit of a no-brainer in that referencing in TiledLayer objects starts at 1 where 0 is used to display a blank tile.

PLAY\_ORDER is an int array that lays down the sequence to play the animated tiles. PLAY\_SIZE indicates how large the PLAY\_ORDER array is.

POLLING\_PERIOD is a binary value used in a bitwise test to determine when the animated sequencing is refreshed. Set it to 0 or 1 for the fastest (aka most CPU wasting) refresh cycle. It has been set to 5 (binary 00000101) to reduce the refresh rate and speed up the game slightly.

The Tuft() constructor initially calls super(), and relies on calcColumns() to set and pass back the COLUMNS constant based on the canvas width. The result is used to load the tuft.png image.

The Layer's position is set via setPosition(), with a new animated tile created by calling createAnimatedTile()

Afterwards, a loop is entered to blanket the "ground" with animated "grass."

There are three other methods in this class.

calcColumns() to determine how many columns are available to display animated tufts of grass, movement() to animate the tufts of grass, and defaultState() to restore everything to the state in which it was first created.

---

### **javax.microedition.lcdui.Sprite**

---

This class is my favorite! While TiledLayer is really useful to create static and animated backgrounds and more, Sprites are cool in the extreme! I mean, what game could possibly exist without the use of tiny images blasting around, doing whatever needs to be done to make the game exciting? Sprites can be created to include a slew of static images that can displayed in whatever sequence you want. The Sprite can also be transformed through mirroring or changing the axis, always fixed on single point of reference. This has the potential to be really cool with the right approach.

---

### **Fields**

---

- TRANS\_MIRROR  
Reflect Sprite about its vertical center.
- TRANS\_MIRROR\_ROT90  
Reflect Sprite about its vertical center and then rotate clockwise by 90 degrees.
- TRANS\_MIRROR\_ROT180  
Reflect Sprite about its vertical center and then rotate clockwise by 180 degrees.
- TRANS\_MIRROR\_ROT270  
Reflect Sprite about its vertical center and then rotate clockwise by 270 degrees.
- TRANS\_NONE  
No transform applied to Sprite.
- TRANS\_ROT90  
Rotate Sprite clockwise by 90 degrees.
- TRANS\_ROT180  
Rotate Sprite clockwise by 180 degrees.
- TRANS\_ROT270  
Rotate Sprite clockwise by 270 degrees.

---

### **Constructors**

---

#### **Sprite (Image image)**

Creates a new non-animated Sprite from the passed Image. This does not allow splitting of the image into frames for animation. Calling this constructor is equivalent to calling new Sprite(image, image.getWidth(), image.getHeight()). The Sprite is visible by default and its upper-left corner is positioned at (0,0) in the painter's coordinate system. This method throws NullPointerException if the passed Image is null.

#### **Sprite (Image image, int frameWidth, int frameHeight)**

Creates a new animated Sprite using frames within the passed Image. All frames must be equally sized using the passed dimensions, and can be laid out horizontally, vertically, or as a grid within the passed Image. The image width must be an integer multiple of the frame width, and the height must be an integer multiple of the frame height. The values returned by Layer.getWidth() and Layer.getHeight() will reflect the frame width and frame height subject to the Sprite's current transform.

Sprites are organized in a default frame sequence that match the raw frame numbers, with the index starting at frame 0. Frame sequencing can be changed with setFrameSequence().

The Sprite is visible by default and its upper-left corner is positioned at (0,0) in the painter's coordinate system.

This method throws:

- `NullPointerException`
  - Image is null
- `IllegalArgumentException`
  - `frameHeight` or `frameWidth` is less than 1
  - Image width is not an integer multiple of the `frameWidth`
  - Image height is not an integer multiple of the `frameHeight`

### **Sprite (Sprite s)**

Creates a new Sprite from another Sprite. All attributes of the passed Sprite, such as the raw frames, position, frame sequence, current frame, reference point, collision rectangle, transform, and visibility are duplicated in the new Sprite.

This method throws `NullPointerException` if the passed Sprite is null.

---

### **Methods**

---

#### **boolean collidesWith (Image image, int x, int y, boolean pixelLevel)**

Checks for a collision between this Sprite and the passed Image that has its upper left corner in the passed location. A collision is only detected if opaque pixels collide when pixel-level detection is used. This means that an opaque pixel in the Sprite would have to collide with an opaque pixel in the Image for a collision to be detected. Only pixels within the Sprite's collision rectangle are checked.

If pixel-level detection is not used, then this method checks if the Sprite's collision rectangle intersects with the Image's bounds.

Any transform that has been applied to the Sprite will be accounted for. The Sprite must be visible to detect a collision.

To call this method, pass the image to test for collision, the horizontal and vertical locations of the Image's upper left corner, and true to test for collision on a pixel-by-pixel basis or false to test using simple bounds checking.

This method will only return true if this Sprite has collided with the Image.

This method throws `NullPointerException` if the Image is null.

#### **boolean collidesWith (Sprite s, boolean pixelLevel)**

Checks for a collision between this Sprite and the passed Sprite. A collision is only detected if opaque pixels collide when pixel-level detection is used. This means that an opaque pixel in the first Sprite would have to collide with an opaque pixel in the second Sprite for a collision to be detected.

Only those pixels within the Sprites' respective collision rectangles are checked.

This method checks if the Sprites' collision rectangles intersect when pixel-level detection is not used.

Any transform that has been applied to the Sprite will be accounted for. The Sprite must be visible to detect a collision.

When you call this method, pass the Sprite to test for collision with, and a boolean value of true to test for collision on a pixel-by-pixel basis, or false to test using simple bounds checking. This method only returns true if the two Sprites have collided.

This method throws `NullPointerException` if Sprite is null.

#### **boolean collidesWith (TiledLayer t, boolean pixelLevel)**

Checks for a collision between this Sprite and the passed Sprite. A collision is detected only if opaque pixels collide when pixel-level detection is used. This means that an opaque pixel in the first Sprite would have to collide with an opaque pixel in the second Sprite for a collision to be detected. Only those pixels within the Sprites' respective collision rectangles are checked.

This method checks if the Sprites' collision rectangles intersect when pixel-level detection is not used.

Any transform that has been applied to the Sprite will be accounted for. The Sprite must be visible to detect a collision.

This method returns true if the two Sprites have collided.

This method throws `NullPointerException` if Sprite is null.

#### **void defineCollisionRectangle (int x, int y, int width, int height)**

Defines the Sprite's bounding rectangle for collision detection purposes. This rectangle is relative to the un-transformed Sprite's upper-left corner and defines the area that is checked for collision detection. Only pixels within the collision rectangle are checked when pixel-level detection is used. By default, a Sprite's collision rectangle is located at 0,0 and has the Sprite's dimensions. The collision rectangle may be larger or smaller than the default rectangle. If it's larger, then the pixels outside the bounds of the Sprite are considered transparent for pixel-level collision detection.

To call this method, pass the horizontal and vertical location of the collision rectangle relative to the untransformed Sprite's left and top edges, and the width and height of the collision rectangle.

This method throws `IllegalArgumentException` if the passed width or height is less than 0.

#### **void defineReferencePixel (int x, int y)**

Defines the Sprite's reference pixel by its location relative to the upper-left corner of the Sprite's un-transformed frame, which may lay outside of the frame's bounds.

When a transformation is applied, the reference pixel is defined relative to the Sprite's initial upper-left corner before transformation. This corner may no longer appear as the upper-left corner in the painter's coordinate system under current transformation.

A Sprite's reference pixel is located at (0,0) by default. This means that the pixel is in the upper-left corner of the raw frame.

Changing the reference pixel does not change the Sprite's position in the painter's coordinate system. This means that the values returned by `getX()` and `getY()` will not change as a result of defining the reference pixel. However, subsequent calls to methods that involve the reference pixel will be impacted by its new definition.

Call this method by passing the horizontal location of the reference pixel, relative to the left edge of the un-transformed frame, and the vertical location of the reference pixel, relative to the top edge of the un-transformed frame.

```
int getFrame()
```

Returns the current index in the frame sequence, and not the index of the actual frame that is displayed.

```
int getFrameSequenceLength()
```

Returns the number of elements in the frame sequence. This does not reflect the number of raw frames, which will be the same if the default frame sequence is used.

```
int getRawFrameCount()
```

Returns the number of raw frames for this Sprite and not the length of the Sprite's frame sequence. These will be the same if the default frame sequence is used.

```
int getRefPixelX()
```

Returns the horizontal position of this Sprite's reference pixel in the painter's coordinate system.

```
int getRefPixelY()
```

Returns the vertical position of this Sprite's reference pixel in the painter's coordinate system.

```
void nextFrame()
```

Positions to the next frame in the frame sequence, which will be the first entry in the sequence after reaching the end of the sequence.

```
void paint  
(Graphics g)
```

Draws the current Sprite frame using the passed Graphics object. The Sprite's upper left corner is rendered at the Sprite's current position relative to the origin of the Graphics object. The current position of the Sprite's upper-left corner can be determined by calling `Layer.getX()` and `Layer.getY()`. Rendering is subject to the clip region of the Graphics object. The Sprite will be drawn only if it is visible.

This method overrides `Layer.paint()`

This method throws `NullPointerException` if the passed Graphics object is null.

```
void prevFrame()
```

Positions to the previous frame in the frame sequence. Note that this will be the last entry in the sequence after reaching the first entry of the sequence.

```
void setFrame  
(int sequenceIndex)
```

Selects the current frame within the frame sequence, which will be rendered after `paint(Graphics)` is called. The parameter, "sequenceIndex", is the desired entry within the frame sequence, and not the index of the actual frame.

This method throws:

- `IndexOutOfBoundsException`
  - sequenceIndex is less than 0
  - sequenceIndex is equal to or greater than the length of the current frame sequence, or the number of raw frames for the default sequence.

```
void setFrameSequence  
(int[] sequence)
```

Sprite frames are displayed in order by default. This method permits setting of an arbitrary sequence of available frames by passing an array of integers containing the frame sequence. The Sprite will revert to the default frame sequence if null is passed as a parameter. Note that the current index in the frame sequence is reset to zero after calling this method. Also note that the contents of the sequence array are copied when this method is called, therefore any changes made to the array after this method returns will have no effect on the Sprite's frame sequence.

This methods throws:

- `ArrayIndexOutOfBoundsException`
  - sequence is populated and any member of the array has a value less than 0 or greater than or equal to the number of frames as reported by `getRawFrameCount()`
- `IllegalArgumentException`
  - The array has less than 1 element

```
void setImage  
(Image img, int frameWidth,  
int frameHeight)
```

Replaces the current raw frames of the Sprite with a new set of raw frames. The values returned by `Layer.getWidth()` and `Layer.getHeight()` will reflect the new frame width and height subject to the Sprite's current transform.

Changing the image for the Sprite could change the number of raw frames. The current frame will remain unchanged if the new frame set has as many or more raw frames than the previous frame set.

If `setFrameSequence()` is used to define a custom frame sequence, the default sequence will remain unchanged. If a custom frame sequence is not defined, then the default frame sequence will be updated to be the default frame sequence for the new frame set. In other words, the new default frame sequence will include all of the frames from the new raw

frame set, as if this new image had been used in the constructor.

If the new frame set has fewer frames than the previous frame set, then the current frame will be reset to entry 0. Any custom frame sequence will be discarded and the frame sequence will revert to the default frame sequence for the new frame set.

The reference point location will be unchanged after calling this method in terms of its defined location within the Sprite and its position in the painter's coordinate system. However, if the frame size is changed and the Sprite has been transformed, the position of the Sprite's upper-left corner may change such that the reference point remains stationary.

If the Sprite's frame size is changed by this method, the collision rectangle is reset to its default value. This means that it is set to the new bounds of the untransformed Sprite.

To call this method, pass the Image to use for the Sprite, and the frame width and height in pixels of the individual raw frames.

This method throws:

- `NullPointerException`
  - The passed Image is null
- `IllegalArgumentException`
  - `frameHeight` or `frameWidth` is less than 1
  - The Image width is not an integer multiple of the `frameWidth`
  - The Image height is not an integer multiple of the `frameHeight`

```
void setRefPixelPosition  
(int x, int y)
```

Sets this Sprite's reference pixel to the passed horizontal and vertical location in the painter's coordinate system.

```
void setTransform  
(int transform)
```

Sets the Sprite's transform to alter its rendered appearance. Transforms are applied to the original Sprite image. They are not cumulative and can't be combined.

Since some transforms involve rotations of 90 or 270 degrees, their use may result in the overall width and height of the Sprite being swapped. As a result, the values returned by `Layer.getWidth()` and `Layer.getHeight()` may change.

The collision rectangle is also modified by the transform so that it remains static relative to the pixel data of the Sprite. Similarly, the defined reference pixel is unchanged by this method, but its visual location within the Sprite may change as a result.

This method repositions the Sprite so that the location of the reference pixel in the painter's coordinate system does not change as a result of changing the transform. Thus, the reference pixel effectively becomes the centerpoint for the transform. Consequently, the values returned by `getRefPixelX()` and `getRefPixelY()` will be the same before and after the transform is applied, but the values returned by `getX()` and `getY()` may change.

To call this method, simply pass the desired transform for this Sprite. This includes:

- `TRANS_MIRROR`  
Reflect Sprite about its vertical center.
- `TRANS_MIRROR_ROT90`  
Reflect Sprite about its vertical center and then rotate clockwise by 90 degrees.
- `TRANS_MIRROR_ROT180`  
Reflect Sprite about its vertical center and then rotate clockwise by 180 degrees.
- `TRANS_MIRROR_ROT270`  
Reflect Sprite about its vertical center and then rotate clockwise by 270 degrees.
- `TRANS_NONE`  
No transform applied to Sprite.
- `TRANS_ROT90`  
Rotate Sprite clockwise by 90 degrees.
- `TRANS_ROT180`  
Rotate Sprite clockwise by 180 degrees.
- `TRANS_ROT270`  
Rotate Sprite clockwise by 270 degrees.

This method throws `IllegalArgumentException` if the passed transform is invalid.

```
package com.blackberrydeveloperjournal.bone;  
  
import javax.microedition.lcdui.*;  
import javax.microedition.lcdui.game.*;  
  
public class Walk extends Sprite {  
    // sprite size in pixels  
    public static final int WIDTH = 23;  
    public static final int HEIGHT = 60;  
  
    // frame display sequence  
    private static final int[] PLAY_ORDER = { 0, 1, 2, 3, 4 };  
    // adjust hopping frame # of PLAY_ORDER as required  
    private static final int BONE_FRAME = 3;  
    // walking speed: 0 and 1 are the fastest, 2 and greater are slower  
    private static final int POLLING_PERIOD = 1;
```

```

// # of steps to reach top of hop
// adjust to alter hop height
private static final int GROUND_LEVEL = 7;
private int hopStage = GROUND_LEVEL;

private static int RIGHT = 1;
private static int LEFT = 0;
private int currentDirection = RIGHT;

private int xSprite, ySprite;

private int pointsScored = 0;
private int delayCounter = 0;

public Walk( int xOrigin, int yOrigin ) throws Exception {
    super( Image.createImage( "img/walk.png"), WIDTH, HEIGHT );

    xSprite = xOrigin;
    ySprite = yOrigin;

    // set reference pixel to the center of the sprite
    defineReferencePixel( WIDTH/2, 0 );
    setRefPixelPosition( xSprite, ySprite );
    setFrameSequence( PLAY_ORDER );
}

// make sprite hop if warranted
public void doHop() {
    if ( hopStage == GROUND_LEVEL ) {
        setFrameSequence( null );
        setFrame( BONE_FRAME );
        // trigger hop
        hopStage--;
    }
}

// handle sprite direction, animation and hopping
public void movement( int direction ) {
    // default to right
    int transform = TRANS_NONE;
    int toMove = 1;

    if ( direction == LEFT ) {
        transform = TRANS_MIRROR;
        toMove = -1;
    }

    // keep walker stationary as background moves
    move( toMove, 0 );

    // change direction is warranted
    if ( currentDirection != direction ) {
        currentDirection = direction;
        setTransform( transform );
    }

    // delay movement based on bit match to POLLING_PERIOD
    if ((delayCounter++ & POLLING_PERIOD) == POLLING_PERIOD ) {
        // not hopping
        if ( hopStage == GROUND_LEVEL ) {

```

```

        nextFrame();

    } else {
        int yPos = getRefPixelY();

        // climbing
        if ( --hopStage > 0 ) {
            yPos += (hopStage * -hopStage);

            // dropping
        } else if ( hopStage != -GROUND_LEVEL + 1 ) {
            yPos += (hopStage * hopStage);

            // hop completed
        } else {
            pointsScored = 0;
            hopStage = GROUND_LEVEL;
            yPos = ySprite;
            setFrameSequence( PLAY_ORDER );
        }

        setRefPixelPosition( getRefPixelX(), yPos );
    }
}

// increase score for successful hop
public int bumpPoints() {
    return ++pointsScored;
}

// decrease score on collision
public int isCollision( Missile missile ) {
    int collisions = 0;

    if ( collidesWith( missile, true ) ) {
        missile.defaultState();
        collisions++;
    }

    return collisions ;
}

// restore default state
public void defaultState() {
    pointsScored = delayCounter = 0;
    hopStage = GROUND_LEVEL;
    setFrameSequence( PLAY_ORDER );
    setRefPixelPosition( xSprite, ySprite );
    setTransform( TRANS_NONE );
    currentDirection = RIGHT;
}
}

```

The Walk class is responsible for the walking “stick man” Sprite. A few constants can be adjusted to reflect whatever you want to do with this code. WIDTH and HEIGHT reflect the dimensions of the Sprite image. In this case, each frame within the walk.png image is 23 pixels wide and 60 pixels high.

PLAY\_ORDER, which is the frame display sequence, is set to play the Sprite in frame sequence. Feel free to adjust the sequencing if you believe that it looks better.

BONE\_FRAME, which is set to array element 3 in the PLAY\_ORDER, is the image used when the Sprite “hops.”

POLLING\_PERIOD is set to quickly refresh the Sprite frame every second increment. Set more complex bit patterns if you want to slow it down even more.

currentDirection is set by default to make the Sprite walk towards the right.

GROUND\_LEVEL is set to 7, which means that when the Sprite “hops”, it will take 7 steps to reach the top, and another 7 steps to return to the original position.

---

```
package com.blackberrydeveloperjournal.bone;

import javax.microedition.lcdui.*;
import javax.microedition.lcdui.game.*;
import java.util.Random;

public class Missile extends Sprite {
    private static final int WIDTH = 20;
    private static final int HEIGHT = 20;

    // ints are faster than booleans
    private static final int YES = 1;
    private static final int NO = 0;

    private static final int RIGHT = 1;
    private static final int LEFT = 0;

    // speed of movement: 0 is fastest, 1 is slower
    // other bit patterns will slow it down even more
    private static final int POLLING_PERIOD = 1;

    // adjust: range 2 to any positive value within reason where 2 is slowest
    private static final int ATTACK_RATE = 5;

    // adjust: a lower value means Spites return to game faster
    private static final int RND_MAX = 25;
    private Random rndObj = new Random();
    private int rndCompare;

    private int travelDirection;
    private int hoppedOver;
    private int spriteY;
    private int delayCounter = 0;

    public Missile( int direction ) throws Exception {
        super( Image.createImage( "img/missile.png" ), WIDTH, HEIGHT );

        // create random value to test for Sprite's reentrance into the game
        rndCompare = rndObj.nextInt( RND_MAX );

        spriteY = BoneManager.HEIGHT - HEIGHT - 2;
        hoppedOver = NO;
        travelDirection = direction;

        int transform = TRANS_NONE;

        if ( travelDirection == RIGHT )
            transform = TRANS_MIRROR;

        setTransform( transform );
        setVisible( false );
    }

    // move and otherwise maintain Sprite
```

```

public int movement( Walk walker, int leftEdge, int rightEdge) {
    int xRef= getRefPixelX();
    int xRefPos = xRef + WIDTH;
    int xRefNeg = xRef - WIDTH;

    // hide Sprite when it moves beyond the edge of the display
    if ( xRefNeg >= rightEdge || xRefPos <= leftEdge )
        setVisible( false );

    int ret = 0;
    int toMove = 1;

    // Sprite is visible and in play
    if ( isVisible() ) {
        // show next animated frame
        if ( ( delayCounter++ & POLLING_PERIOD) == POLLING_PERIOD )
            nextFrame();

        // precalc for speed and reduced code size
        int walkX = walker.getRefPixelX();
        boolean isMore = xRef > (walkX + walker.WIDTH );
        boolean isLess = xRef < walkX;
        boolean bump = false;

        if ( travelDirection == RIGHT ) {
            toMove = ATTACK_RATE;

            if ( hoppedOver == NO && isMore ) {
                hoppedOver = YES;
                bump = true;
            }
            // leftward bound
        } else {
            toMove = -ATTACK_RATE;

            if ( hoppedOver == NO && isLess ) {
                hoppedOver = YES;
                bump = true;
            }
        }

        if ( bump )
            ret = walker.bumpPoints();

        move( toMove, 0 );

        // Sprite is not visible
        // test if it's time to let the Sprite back in the game
    } else if ( rndObj.nextInt( RND_MAX ) == rndCompare ) {
        int edge = leftEdge;
        toMove = 1;

        // place Sprite's outer edge on screen
        if ( travelDirection == LEFT ) {
            edge = rightEdge;
            toMove = -1;
        }

        hoppedOver = NO;
        setVisible( true );
        setRefPixelPosition( edge, spriteY );
    }
}

```

```

        move( toMove, 0 );
    }

    return ret;
}

// restore Sprite to default state
public void defaultState() {
    hoppedOver = NO;
    delayCounter = 0;
    setVisible( false );
}
}

```

The Missile class is responsible for the Sprite missiles that are hurtled towards the “stick man” Sprite from the right and left sides.

As before, a few constants have been provided to adjust as required. WIDTH and HEIGHT reflect the dimensions of the Sprite image. In this case, each frame within the missile.png image is 20 pixels square.

POLLING\_PERIOD is set to quickly refresh the Sprite frame every second increment.

ATTACK\_RATE is set to move the Missile Sprites 5 pixels each time. Adjust this to increase or decrease the speed of the missiles.

RND\_MAX, which is set to 25, is used to set the highest number generated by the random number generator. A lower value means that Missile Sprites will return to the game faster, where a higher value will keep them out longer.

The constructor takes a single parameter used to set the direction that the Missile moves. Pass 1 for RIGHT and 0 for LEFT.

The movement() method handles movement of a Missile Sprite and takes three parameters: The Walk Sprite, and the left and right edges of display area.

A test is first made to determine if the Missile has travelled beyond the display border. If so, then it is made invisible.

If the Missile is still visible, then a series of events may occur.

To start, the next Sprite frame is displayed if the delayCounter matches the bit pattern of the POLLING\_PERIOD.

After that, some calculations are made to increase speed, reduce code size and improve readability.

First, the x coordinate of the Walk Sprite’s reference pixel is used to set two boolean values to true if the Walk Sprite managed to hop over the Missile Sprite. These booleans are indirectly used later to increase the game points.

The Missile Sprite is also moved by ATTACK\_RATE pixels in the direction of movement.

If the Missile Sprite is not visible, then a test is made to determine if it’s okay to let the Missile back in the game. If so, then the Missile’s reference pixel is set, it is made visible, and is moved by one pixel in the direction it is travelling.

The final method in this class, defaultState(), resets the Missile Sprite to its default state.

---

## The Bone Class

---

The following is the main Bone class that takes care of all the pedestrian duties of the application such as creating menus, starting, pausing and stopping the application, and displaying exceptions. The code is fairly straightforward so I won’t go into detail about it afterwards.

```

package com.blackberrydeveloperjournal.bone;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class Bone extends MIDlet implements CommandListener {
    private boolean MENU_ADD      = true;
    private boolean MENU_REMOVE   = false;
    private boolean pauseMenuUsed = MENU_REMOVE;
    private boolean playMenuUsed  = MENU_REMOVE;
    private boolean replayMenuUsed = MENU_REMOVE;
    private Command gamePlay      = new Command( "Play", Command.SCREEN, 1 );
    private Command gamePause     = new Command( "Pause", Command.SCREEN, 1 );
    private Command gameReplay    = new Command( "Replay", Command.SCREEN, 1 );
    private Command gameExit      = new Command( "Exit", Command.EXIT, 10 );
}

```

```

private BoneThread gameThread;
private BoneCanvas gameCanvas;

public Bone() {
    try {
        gameCanvas = new BoneCanvas( this );
        gameCanvas.addCommand( gamePause );
        pauseMenuUsed = true;
        gameCanvas.addCommand( gameExit );
        gameCanvas.setCommandListener( this );
    } catch( Exception err ) {
        showException( err );
    }
}

// signals the MIDlet that it has entered the Active state
public void startApp() throws MIDletStateChangeException {
    if ( gameCanvas != null ) {
        if ( gameThread != null ) {
            handleMenuPlay( MENU_REMOVE );
            handleMenuPause( MENU_ADD );
            gameCanvas.getKeyStates();
            gameThread.restart();
        } else {
            gameThread = new BoneThread( gameCanvas );
            gameCanvas.canvasBegin();
            gameThread.start();
        }
    }
}

// signals the MIDlet to stop and enter the Paused state
public void pauseApp() {
    // pause game
    if ( gameThread != null )
        gameThread.pause();

    // adjust menu to allow restart of game
    if ( gameCanvas != null ) {
        handleMenuPause( MENU_REMOVE );
        handleMenuReplay( MENU_REMOVE );
        handleMenuPlay( MENU_ADD );
    }
}

// signals the MIDlet to terminate and enter the Destroyed state
public void destroyApp( boolean unconditional ) throws MIDletStateChangeException {
    // stop game
    if ( gameThread != null )
        gameThread.stop();

    // deactivate objects and then call garbage collector
    gameCanvas = null;
    gameThread = null;
    System.gc();
}

// handle play, pause, replay and exit commands
public void commandAction( Command c, Displayable d ) {
    // start the game
    if ( c == gamePlay ) {
        handleMenuPlay( MENU_REMOVE );
        handleMenuPause( MENU_ADD );
        gameCanvas.getKeyStates();
    }
}

```

```

    gameThread.restart();
}

// pause the game
if ( c == gamePause ) {
    handleMenuPause( MENU_REMOVE );
    handleMenuPlay( MENU_ADD );
    gameThread.pause();
}

// restart the game
if ( c == gameReplay ) {
    handleMenuReplay( MENU_REMOVE );
    handleMenuPause( MENU_ADD );
    gameCanvas.canvasInitialize();
    gameThread.restart();
}

// quit the game
if ( c == gameExit || c == Alert.DISMISS_COMMAND ) {
    try {
        destroyApp( false );
        notifyDestroyed();
    } catch ( MIDletStateChangeException err ) {
        showException( err );
    }
}

// display exception
public void showException( Exception err ) {
    String theError = (err.getMessage() == null) ? err.getClass().getName()
        : err.getClass().getName() + " - " + err.getMessage();
    Alert promptError = new Alert( "Exception", theError, null, AlertType.ERROR );
    promptError.setCommandListener( this );
    promptError.setTimeout( Alert.FOREVER );
    Display.getDisplay( this ).setCurrent( promptError );
}

// add or remove Play menu item
private void handleMenuPlay( boolean enable ) {
    if ( enable == MENU_ADD && playMenuUsed == MENU_REMOVE ) {
        gameCanvas.addCommand( gamePlay );
        playMenuUsed = MENU_ADD;
    } else if ( enable == MENU_REMOVE && playMenuUsed == MENU_ADD ) {
        gameCanvas.removeCommand( gamePlay );
        playMenuUsed = MENU_REMOVE;
    }
}

// add or remove Pause menu item
private void handleMenuPause( boolean enable ) {
    if ( enable == MENU_ADD && pauseMenuUsed == MENU_REMOVE ) {
        gameCanvas.addCommand( gamePause );
        pauseMenuUsed = MENU_ADD;
    } else if ( enable == MENU_REMOVE && pauseMenuUsed == MENU_ADD ) {
        gameCanvas.removeCommand( gamePause );
        pauseMenuUsed = MENU_REMOVE;
    }
}

// add or remove Replay menu item
private void handleMenuReplay( boolean enable ) {
    if ( enable == MENU_ADD && replayMenuUsed == MENU_REMOVE ) {
        gameCanvas.addCommand( gameReplay );
    }
}

```

```

    replayMenuUsed = MENU_ADD;
} else if ( enable == MENU_REMOVE && replayMenuUsed == MENU_ADD ) {
    gameCanvas.removeCommand( gameReplay );
    replayMenuUsed = MENU_REMOVE;
}
}
// called externally
public void setMenuStart() {
    handleMenuPause( MENU_REMOVE );
    handleMenuPlay( MENU_REMOVE );
    handleMenuReplay( MENU_ADD );
}
}

```

---

### The BoneThread Class

---

The final class handles the game thread that is responsible for all gaming activity. The code is even more straightforward than the Bone class and doesn't need further explanation.

```

package com.blackberrydeveloperjournal.bone;

public class BoneThread extends Thread {
    private BoneCanvas hCanvas;
    private boolean requestStop = false;
    private boolean requestPause = false;

    // constructor
    BoneThread( BoneCanvas target ) {
        hCanvas = target;
    }
    // begin the game
    public void run() {
        hCanvas.getKeyStates();
        requestStop = requestPause = false;

        do {
            while ( requestPause ) {
                synchronized(this) {
                    try {
                        // make thread wait for another thread to invoke the
                        // notify() method or the notifyAll() method for this object
                        wait();
                    } catch( Exception err ) {}
                }
            }

            // poll keys and move display layers
            hCanvas.canvasRefresh();

            // pause to allow other threads to operate
            synchronized( this ) {
                try {
                    // make thread wait for 1 millisecond or for another thread to invoke
                    // the notify() method or the notifyAll() method for this object
                    wait( 1 );
                } catch( Exception err ) {}
            }
        } while ( !requestStop );
    }
    // pause game
    public void pause() {

```

```

    requestPause = true;
}
// restart game
public void restart() {
    requestPause = false;
    doNotify();
}
// stop game
public void stop() {
    requestStop = true;
    doNotify();
}
// wakes up a single thread that is waiting on this object's monitor
private void doNotify() {
    synchronized( this ) {
        notify();
    }
}
}
}

```

---

### The Final Components

---



**The Walk Sprite image**



**The Missile Sprite image**



**The Tuft TiledLayer image**



**The Bone icon**

---

### Summary

---

This article was written to make it easier for you to develop games. In documenting the Java ME Gaming API and providing working source code for the Bones game, you now should have a reasonable grasp of game development on devices that support Java ME and MIDP 2.0.

# Mobilize the BlackBerry Resource Kit User Administration Function to BlackBerry Devices

Garry Seifried, Research In Motion

In this article I will explore how to take an existing business operation and enable it for use in a Windows® executable program, a web user interface, and as a BlackBerry® MDS Studio application.

---

*Note: The BESUserAdminClient.exe program from the BlackBerry Resource Kit is discussed in this article.*

---

We will create a reusable Dynamic Link Library (DLL) to invoke the BESUserAdminClient.exe command line process. This will include necessary interface operations for the commands that we wish to support. After completion, this library could be reused in a variety of presentation mechanisms such as a web page. A web service can also be created to expose the library interface as web service operations. This could be used by BlackBerry MDS Studio to mobilize the user administration operations to BlackBerry devices.

A use case scenario would be to mobilize the Set Password operation to the BlackBerry solution using a DLL, Web Services and BlackBerry MDS Studio.

These concepts could then be applied to other applications and processes in a similar manner.

Microsoft .NET is the technology used for the programming components within this article.

Attachments for this article include:

- BlackBerry MDS Studio Application
- Web Service
- A Dynamic Link Library

These can be downloaded from the BlackBerry Developer Journal web site.

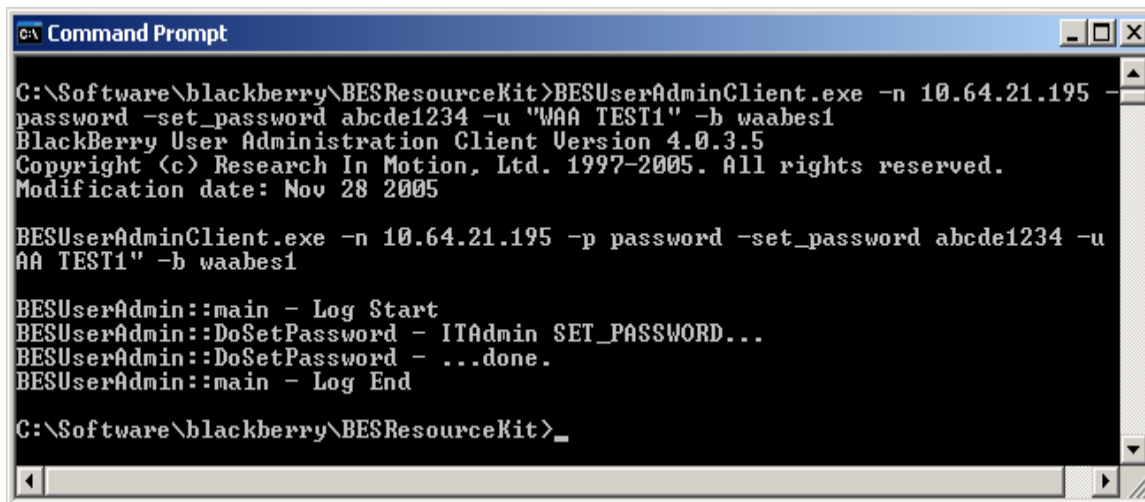
---

## Step 1: Identify the output from BESUserAdminClient commands

---

Command line operations return the command result and output in standard error and standard output respectively. The BESUserAdminClient follows this pattern. Only a few-commands actually return information via standard output. These are “-stats” and “-find”.

As an example, a “-stats -user” operation will return the success/failure of the command using stderr, and the output about the user from the command execution using stdout. The following shows “set\_password” in use:



```
C:\> Command Prompt
C:\Software\blackberry\BESResourceKit>BESUserAdminClient.exe -n 10.64.21.195 -p
password -set_password abcde1234 -u "WAA TEST1" -b waabes1
BlackBerry User Administration Client Version 4.0.3.5
Copyright (c) Research In Motion, Ltd. 1997-2005. All rights reserved.
Modification date: Nov 28 2005

BESUserAdminClient.exe -n 10.64.21.195 -p password -set_password abcde1234 -u
AA TEST1" -b waabes1

BESUserAdmin::main - Log Start
BESUserAdmin::DoSetPassword - ITAdmin SET_PASSWORD...
BESUserAdmin::DoSetPassword - ...done.
BESUserAdmin::main - Log End

C:\Software\blackberry\BESResourceKit>_
```

Use “BESUserAdminClient -?” to get a list of the commands and the command syntax:

```
C:\Software\blackberry\BESResourceKit>BESUserAdminClient.exe -?
BlackBerry User Administration Client Version 4.0.3.7
Copyright (c) Research In Motion, Ltd. 1997-2005. All rights reserved.
Modification date: Dec 21 2005

BESUserAdminClient.exe -?

Usage: BESUserAdminClient.exe [options]

    -?
    -h : help

    -p <password> : administration password
    -n <network address> : network address (or computer name) of where BESUserAdminService is running
    [-v] : verbose output

    -add : add user
    -u <user name> : unique user name
    -b <BES instance> : target BES
    [-pin <PIN>] : PIN for the user
    [-w <password>] : user's activation password
    [-wt <expiry time>] : activation password expiry time (number of hours)
    [-it_policy <ITPolicy>] : ITPolicy for the user
    [-i <input filename>] : over-rides '-u', '-b', '-pin', '-w' and 'it_poli
```

---

## Step 2: Create a mechanism to invoke the BESUserAdminClient executable

---

The Process command executes a properly formed command that is passed as a parameter. This would be provided as command line input if BESUserAdminClient.exe was directly used.

We will use the Process object to invoke the BESUserAdminClient.exe program, read the stdout first because it

returns a larger amount of data, and then read the stderr for the command execution. We will want to return that information to the caller of the command. Since some commands do not have information returned from stdout, we will create a complex structure that will allow us to work with the results from all commands more easily.

---

*Note:* The ProcessCommand method is in the BRKLibrary class.

---

```
namespace BBResourceKitLibrary
{
    /// <summary>
    /// BlackBerry Resource Kit Library is a DLL wrapper around the
    /// BESUserAdminClient.exe process.
    /// Methods call the BRKCommand.ProcessCommand() method to
    /// invoke the sub-process.
    /// BESUserAdminClient.exe returns output using standard output (stdOut)
    /// and standard error (stdErr).
    /// These are captured into an ArrayList of two items, as follows:
    /// item[0] is the standard error
    /// item[1] is an ArrayList of string for lines of standard output.
    /// </summary>
    public class BRKLibrary
    {
        /// <summary>
        /// ProcessCommand() uses a Process object as an execution shell to
        /// invoke BESUserAdminClient.exe
        /// A complex structure is returned which uses an ArrayList for the
        /// stdErr and stdOut.
        /// </summary>
        /// <param name="cmd"></param>
        /// <returns>ArrayList[0] string for stdErr</returns>
        /// <returns>ArrayList[1] ArrayList for stdOut defined as:</returns>
        /// <returns>ArrayList[0] string for stdOut header</returns>
    }
}
```

```

/// <returns>ArrayList[1] ArrayList for stdout content</returns>
public ArrayList ProcessCommand(String cmd)
{
    // A Process object is used for execution of the EXE
    ArrayList result = new ArrayList();
    Process myProcess = new Process();
    try
    {
        myProcess.StartInfo.UseShellExecute = false;
        myProcess.StartInfo.FileName = app; // BESUserAdminClient
        myProcess.StartInfo.Arguments = cmd; // the command string
        myProcess.StartInfo.RedirectStandardError = true;
        myProcess.StartInfo.RedirectStandardOutput = true;
        myProcess.StartInfo.CreateNoWindow = true;
        myProcess.StartInfo.WindowStyle = ProcessWindowStyle.Normal;
        myProcess.Start();
        // Read StdOut before StdErr so we don't block
        ArrayList readLine = new ArrayList();
        string stdout;
        while ((stdout = myProcess.StandardOutput.ReadLine()) != null)
        {
            readLine.Add(stdout);
        }
        myProcess.WaitForExit();
        stderr = myProcess.StandardError.ReadToEnd();
        // Determine Command Success - search for 'error'
        CheckCommandSuccess(stderr);
        // ArrayList[0] contains the stderr string
        result.Add(stderr);
        // ArrayList[1] contains the stdout ArrayList
        result.Add(readLine);
    }
    catch (Exception e)
    {
        Console.WriteLine(e.Message);
    }
    finally
    {
        myProcess.Close();
    }
    return result;
}
}

```

The execution of the BESUserAdminClient.exe program is specified by this statement:

```
myProcess.StartInfo.FileName = app;
```

This is externalized in a settings.config file as shown:

```
<add key="CommandName" value="C:\Software\blackberry\BESResourceKit\BESUserAdminClient.exe" />
```

This is accessed in the program using the ConfigurationSettings:

```
String app = ConfigurationSettings.AppSettings["CommandName"];
```

The CheckCommandSuccess(stderr) method searches the stderr for 'Error' using values defined in the settings.config:

```
<add key="LogStart" value="BESUserAdmin::main - Log Start" />
```

```
<add key="LogError" value="Error" />
```

```
<add key="LogEnd" value="BESUserAdmin::main - Log End" />
```

The CheckCommandSuccess(stderr) method searches the stderr for 'Error' using values defined in the settings.config and sets the values for commandResult and commandSuccess:

```

/// <summary>
/// Check command success - false if stderr contains "Error"
/// </summary>

```

```

/// <param name="stdErr"></param>
/// <returns>bool</returns>
private bool CheckCommandSuccess(String stdErr)
{
    string logStart = ConfigurationSettings.AppSettings["LogStart"];
    string logEnd = ConfigurationSettings.AppSettings["LogEnd"];
    string error = ConfigurationSettings.AppSettings["LogError"];
    // Start of the message section is prefixed by Log Start
    int ix = stdErr.IndexOf(logStart);
    stdErr = stdErr.Substring(ix + logStart.Length + 2);
    // End of the message section is suffixed by Log End
    ix = stdErr.IndexOf(logEnd);

    if (ix>-1)
    {
        stdErr = stdErr.Substring(0,ix);
    }
    // CommandResult
    commandResult = stdErr;
    // CommandSuccess indicated by absence of "Error"
    ix = stdErr.IndexOf(error);

    if (ix>-1)
    {
        commandSuccess = false;
    }
    else
    {
        commandSuccess = true;
    }
    return commandSuccess;
}

```

---

### Step 3: Create an interface with the commands you want to support

---

The “BESUserAdminClient.exe -?” option displays the commands and supported command syntax and options. We will only show the 'SetPassword' command in this article:

SetPassword

- -set\_password <password>
- -u <user name> : unique user name
- -b <BES instance> : source BlackBerry Enterprise Server

We will create an interface for the command where the required parameters are listed.

Note the use of the XmlTypeAttribute attribute (enclosed in [], below) to declare the object serializable, to be used in a Web Service. When we look at the Web Service and BlackBerry MDS Studio application, we will see the BRKSetPassword object as part of their definitions because of the attribute having being used:

```

namespace BBResourceKitLibrary
{
    /// <summary>
    /// Create a class for the command and parameters and make it serializable
    /// for use in a Web Service if you wish to mobilize to MDS Studio
    /// </summary>

    [System.Xml.Serialization.XmlTypeAttribute(Namespace="http://rim.net")]
    public class BRKSetPassword
    {
        public BRKSetPassword() {}
    }
}

```

```

public BRKSetPassword(
    String network,
    String password,
    String bes,
    String user,
    String activation
)
{
    this.network = network;
    this.password = password;
    this.bes = bes;
    this.user = user;
    this.activation = activation;
}

public String network;
public String password;
public String bes;
public String user;
public String activation;
}
}

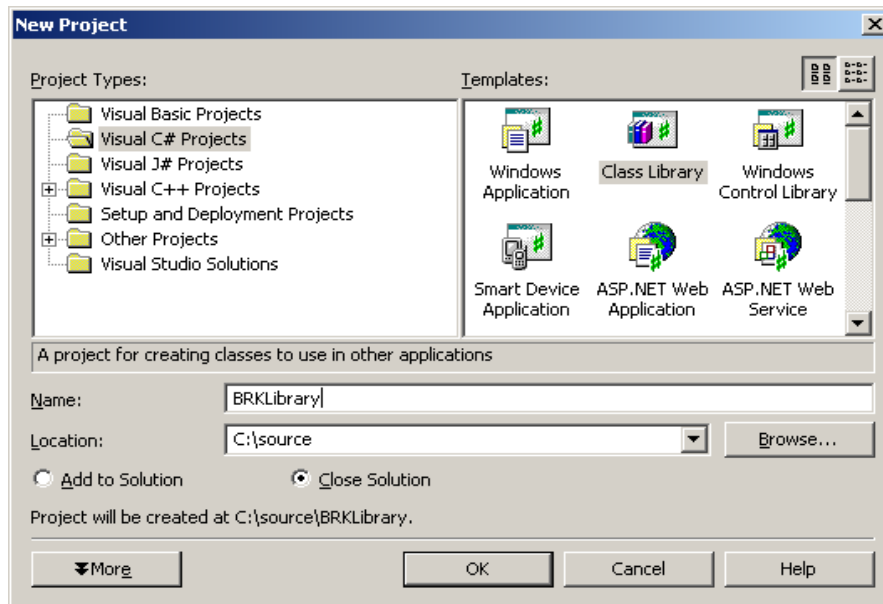
```

---

#### Step 4: Create a DLL Library to support the command invocation and results

---

Create a new Project using the Project Wizard and select a Class Library:



Note that the SetPassword method is also in the BRKLibrary class.

```

namespace BBResourceKitLibrary
{
    /// <summary>
    /// BRKLibrary also defines the commands we support.
    /// </summary>
    public class BRKLibrary
    {
        /// <summary>
        /// Set Activation Password
        /// </summary>
        /// <param name="obj"></param>
        /// <returns></returns>
        public void SetPassword(BRKSetPassword obj)
        {
            String cmd = BRKCommand.BuildCommand(
                BRKCommand.CMD_SETPASSWORD,
                obj.network,
                obj.password,
                obj.bes,
                obj.user,
                obj.activation,
                ""
            );
            results = ProcessCommand(cmd);
        }

        /// <summary>
        /// Contains the command result
        /// </summary>
        private string commandResult;

        /// <summary>
        /// Contains the command success
        /// </summary>
        private bool commandSuccess = false;

        /// <summary>
        /// CommandSuccess
        /// </summary>
        /// <returns>bool</returns>
        public bool GetCommandSuccess()
        {
            return commandSuccess;
        }

        /// <summary>
        /// Return command result.
        /// </summary>
        /// <returns>string</returns>
        public string GetCommandResult()
        {
            return commandResult;
        }

        /// <summary>
        /// Return a CommandResponse
        /// </summary>
        /// <returns>string</returns>
        public CommandResponse GetCommandResponse()
        {
            CommandResponse response = new CommandResponse();

```

```

        response.success = GetCommandSuccess();
        response.response = GetCommandResult();
        return response;
    }
}

```

Create another class that contains the command success and stderr response. This is serializable so that it can be used in a Web Service and BlackBerry MDS Studio:

```

/// <summary>
/// CommandResponse is a composite. It is serializable.
/// </summary>
[System.Xml.Serialization.XmlTypeAttribute(Namespace="http://rim.net")]
public class CommandResponse
{
    public bool success;
    public string response;
}

```

This example shows the use of `BRKCommand.BuildCommand` (not included in this article), which is a static method used to build the command from the parameters. It creates the correct command format based on the `BRKCommand.CMD_SETPASSWORD` constant. This relates to step 1 and step 3, where you need to construct the command(s) you want to support.

The following shows the command syntax and successful command execution:

```

C:\Software\BESResourceKit>BESUserAdminClient.exe -n 10.64.21.195 -p password -b WAABES1 -set_password abcd1234 -u "Waa Test2"
BlackBerry User Administration Client Version 4.0.3.7
Copyright (c) Research In Motion, Ltd. 1997-2005. All rights reserved.
Modification date: Dec 21 2005

BESUserAdminClient.exe -n 10.64.21.195 -p password -b WAABES1 -set_password abcd1234 -u "Waa Test2"

BESUserAdmin::main - Log Start
BESUserAdmin::DoSetPassword - ITAdmin SET_PASSWORD...
BESUserAdmin::DoSetPassword - ...done.
BESUserAdmin::main - Log End

C:\Software\BESResourceKit>

```

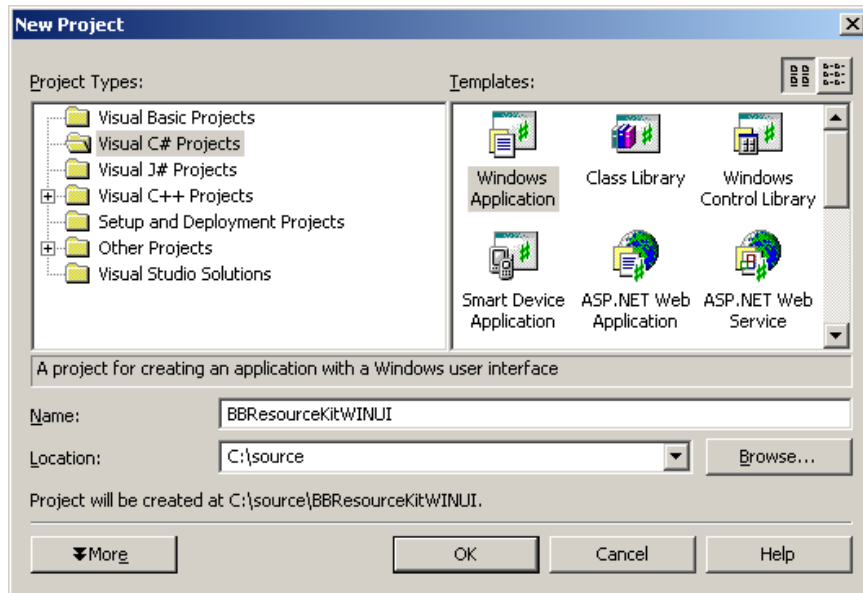
---

### Step 5: Create an implementation using the DLL: a Windows UI or Web UI

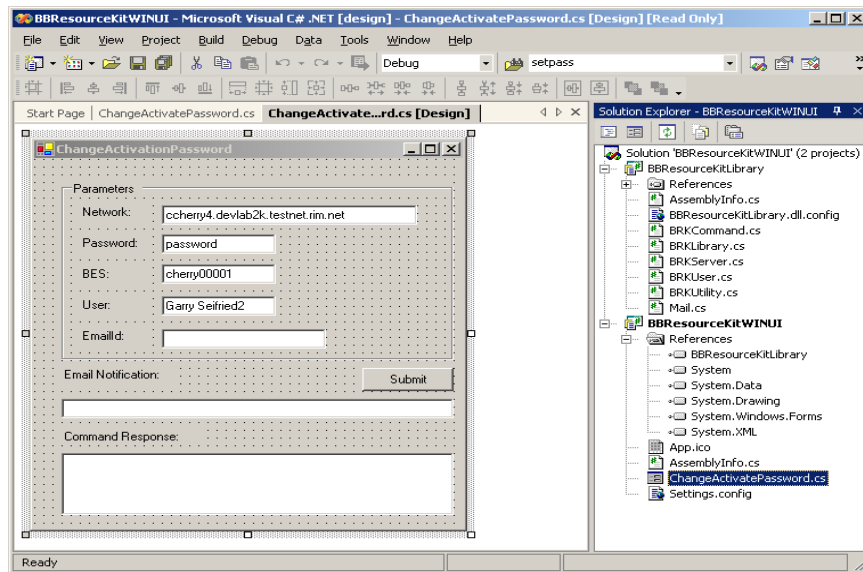
---

This step is optional, its purpose is to test the DLL library created above.

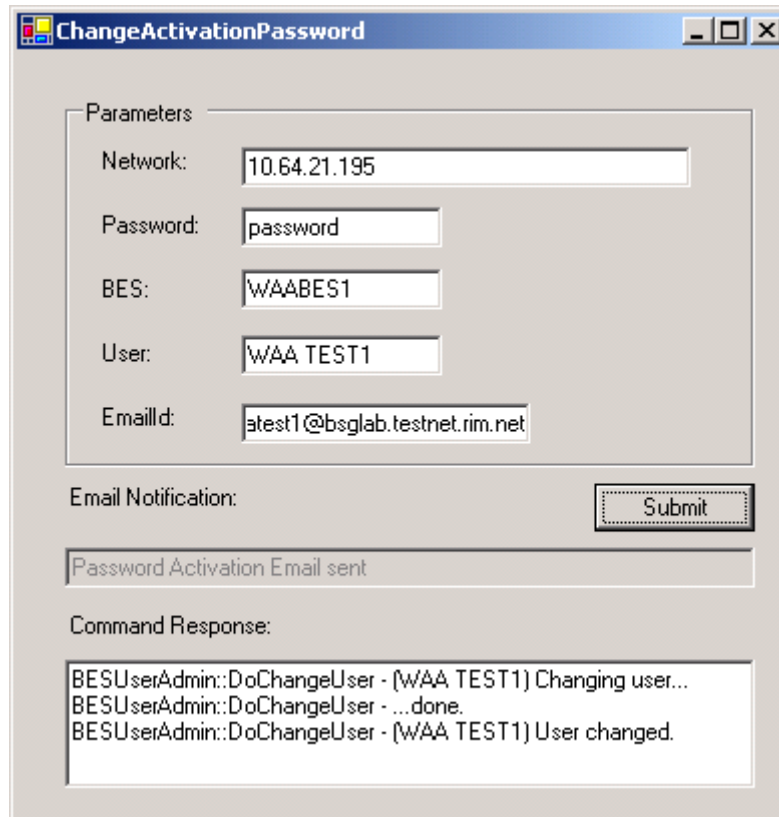
This `BBResourceKitWINUI` project is a Windows Application which presents a form for the collection of the parameters needed for the Set Password operation.



The project references the BBResourceKitLibrary which contains the interface for the commands it supports as well as the ProcessCommand method that is used to invoke the BESUserAdminClient.exe program.

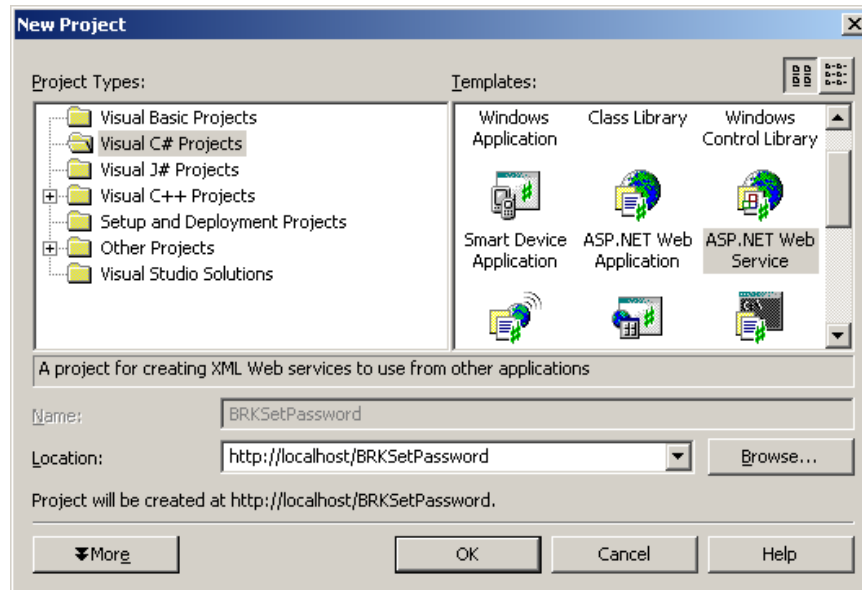


This UI also reports the stderr response from the command:

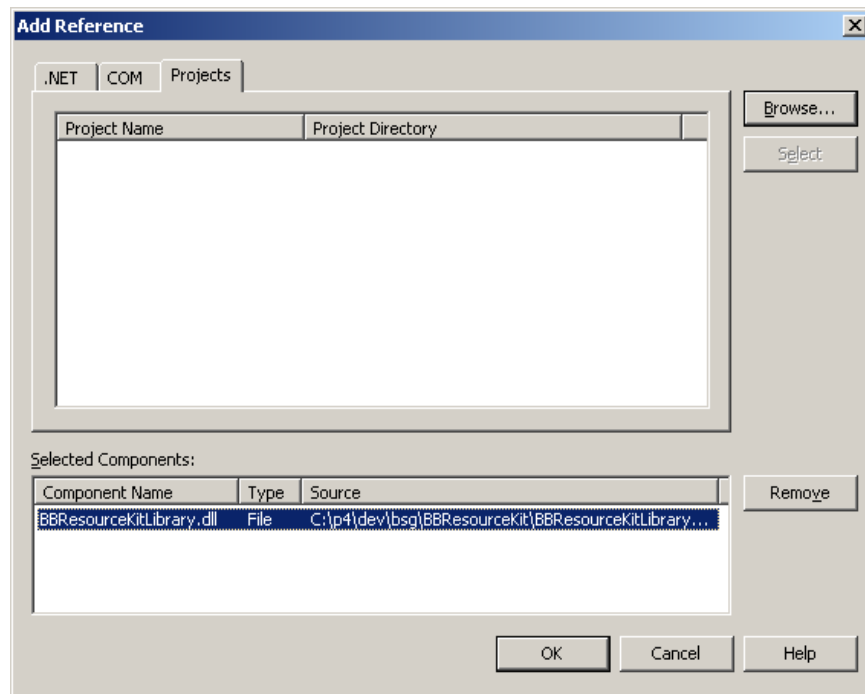


### Step 6: Create a Web Service for the DLL Library

Create a new Project using the Project Wizard and select a Class Library:



Add the DLL as a reference to the Web Service project:



The following is code to provide an implementation for the Web Service operation, specifying the BRKLibrary SetPassword method that receives a BRKSetPassword object as a parameter. Note use of the [WebService] and [WebMethod] attributes. These are provided when a Web Service is created and must be present:

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Web;
using System.Web.Services;

using BBResourceKitLibrary;

namespace BRKSetPasswordWebService {

    [WebService(
        Namespace="http://rim.net/",
        Name="BRK Set Password Command as a Web Service in C#",
        Description="This web service provides BESUserAdminClient commands.")
    ]

    /// <summary>
    /// Summary description for Service1.
    /// </summary>
    public class Service1 : System.Web.Services.WebService {
        public Service1() {
            //CODEGEN: This is required by the ASP.NET Web Services Designer
            InitializeComponent();
        }

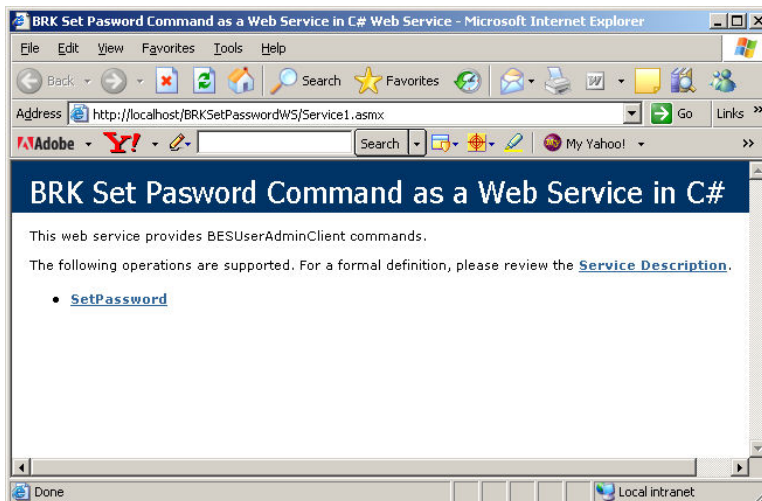
        [WebMethod]
        public CommandResponse SetPassword(BRKSetPassword obj)
```

```

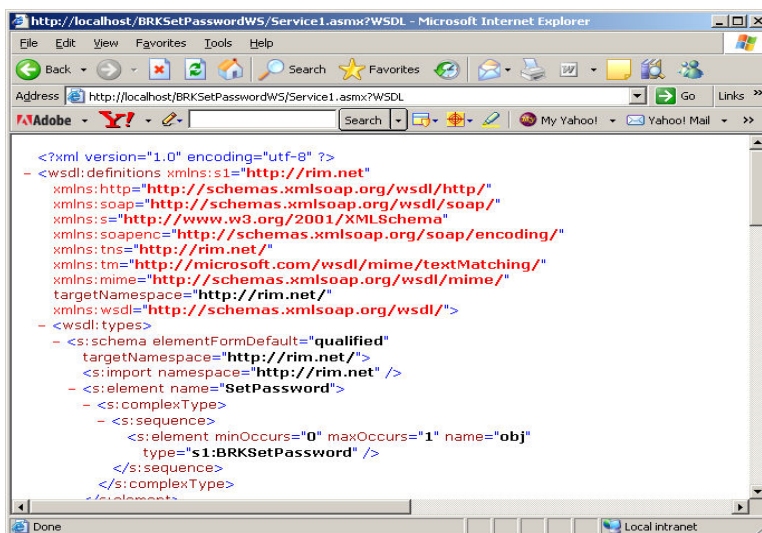
    {
        BRKLibrary lib = new BRKLibrary();
        lib.SetPassword(obj);
        return lib.GetCommandResponse();
    }
}

```

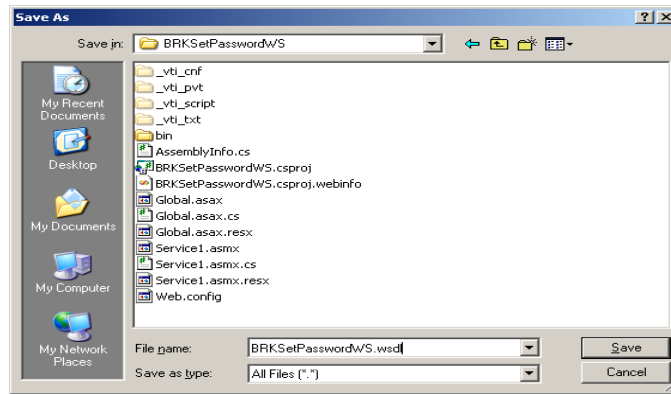
Review the Web Service: invoke with Debug | Start:



Review the Service Description:



Save the Service Description for use in MDS Studio:

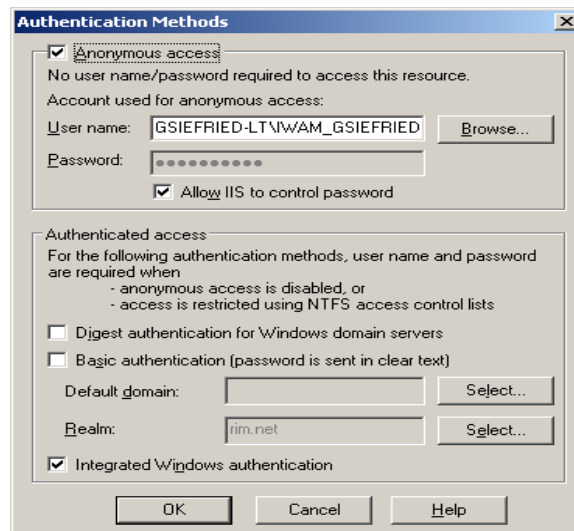


---

### Step 6b: Configure the Web Service to Execute a Process

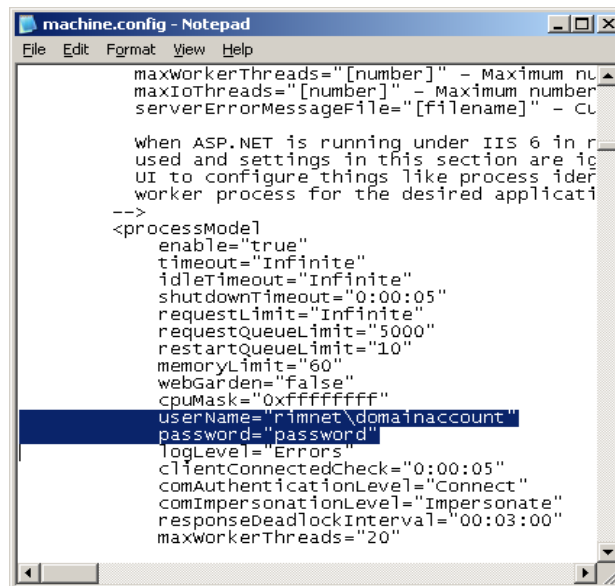
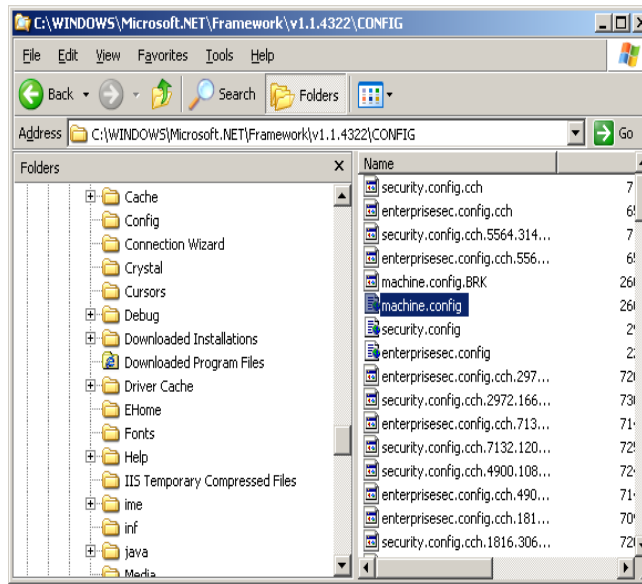
---

View the properties of the BRKSetPassword web service to set the Directory Security and Authentication methods for Anonymous access:



The process must be run with an ASP.NET worker process using a domain account to execute the BESUserAdminClient program which uses RMI to access the server computer. This would involve assigning the Account used for anonymous access to a domain account.

Alternatively, modify the machine.config as shown below, or configure IIS to accomplish the same outcome:

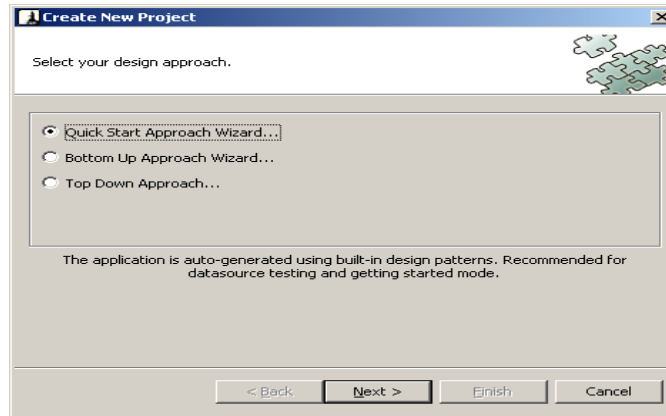


---

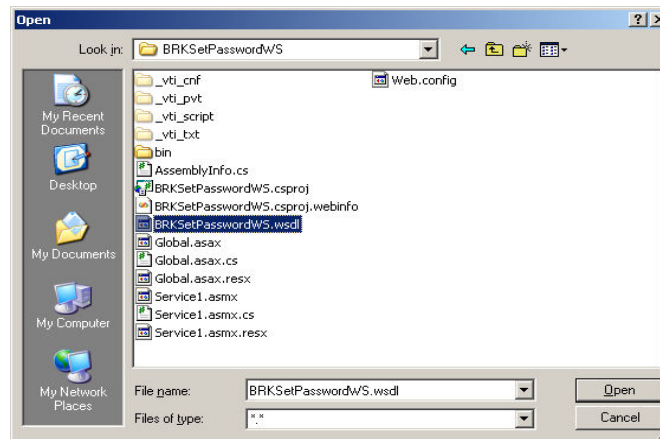
## Step 7: Create a MDS Studio application

---

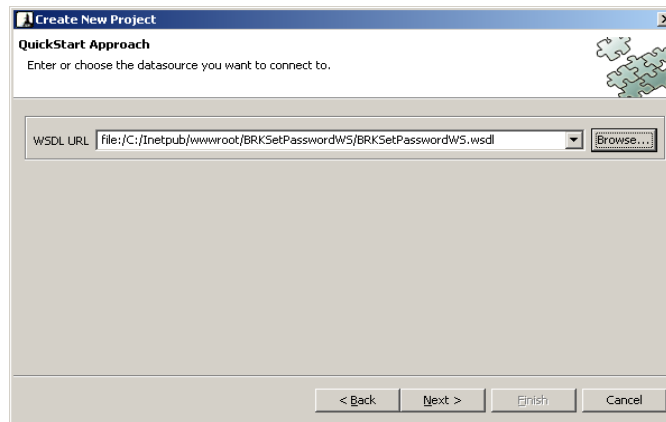
Use the saved WSDL file created in Step 6 to create a BlackBerry MDS Studio Quick Start application:



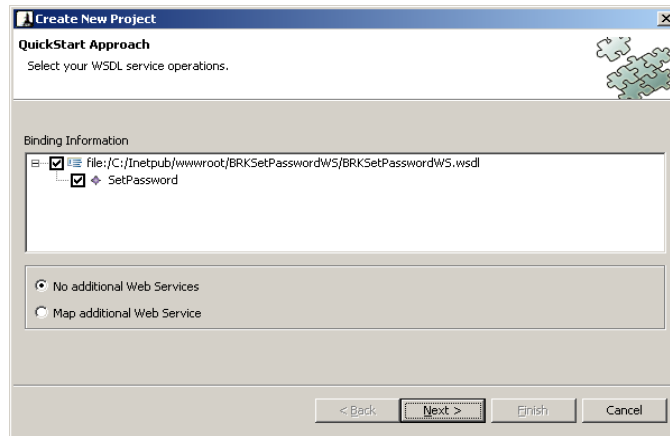
Locate the saved WSDL on the local file system:



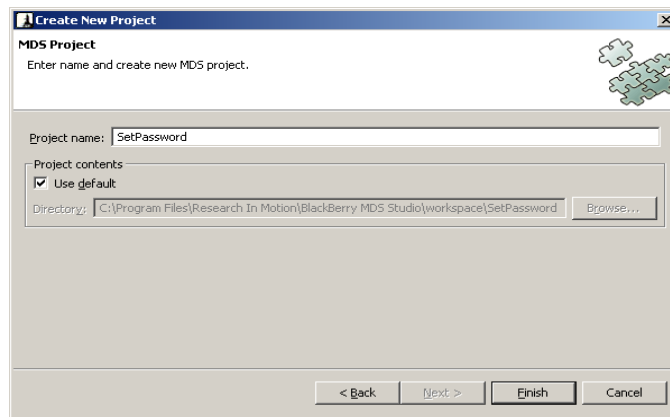
Continue with the QuickStart Approach:



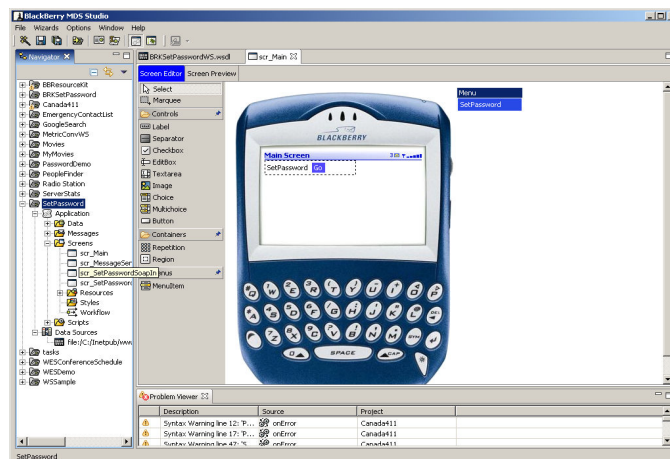
Select the service operations for this application:



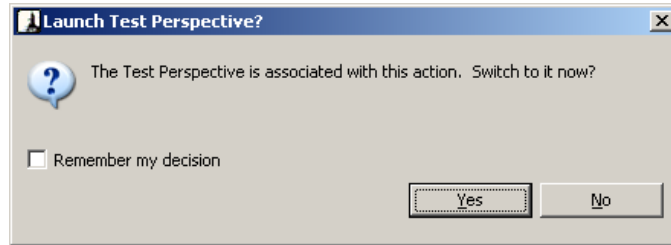
Name the Project:



Test the generated application:



From within BlackBerry MDS Studio, select the Project and right-click to get the Test option:



Select the SetPassword application from the ribbon:

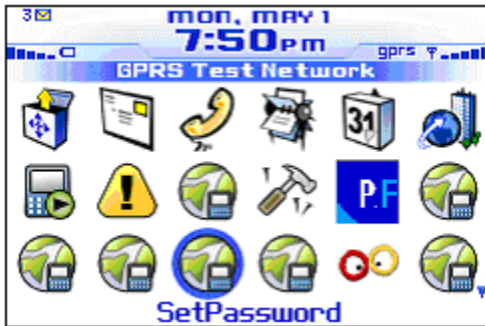


Figure 1: SetPassword MDS application

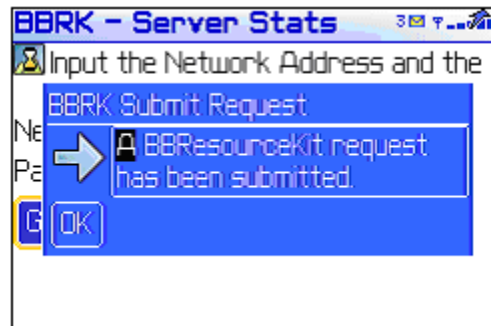


Figure 4: Submit screen

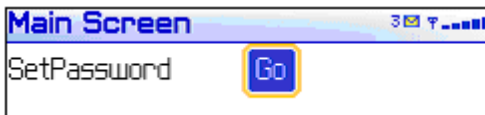


Figure 2: QuickStart Main Screen

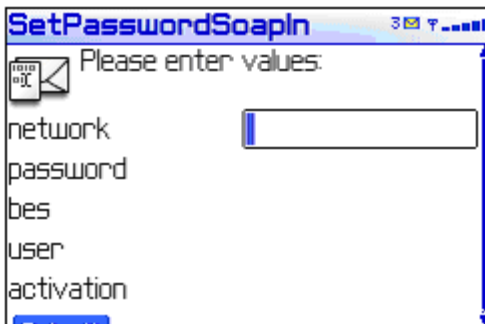


Figure 3: Input screen

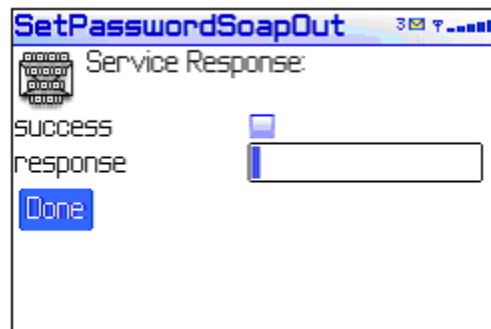


Figure 5: Response screen

# The BlackBerry Browser does JavaScript™

Richard Evers, Editor

JavaScript's roots date back to early 1995 when Brendan Eich was hired by Netscape® to take charge of a project to design and implement a new language. In this capacity he created a scripting language that eventually became known as JavaScript.

JavaScript was publicly announced by Netscape and Sun® on December 4, 1995 and was included with the release of Netscape 2 in early 1996.

Netscape 3 with JavaScript 1.1 was released in 1996. This release marked completion of Document Object Model (DOM) level 0. DOM is a model that describes how all elements in an HTML page relate to the document itself, which is the upper-most structure. DOM level 0 offered access to a few HTML elements such as forms, and later provided access to images. Modern browsers still provide backward support for DOM level 0.

In 1996 Netscape passed the JavaScript language to the European Computer Manufacturers Association (ECMA) for standardization. In turn, the ECMA created the ECMA Script standard based on the core syntax of JavaScript but failed to include all aspects of DOM level 0.

Over time, ECMA Script kept evolving in line with many of the changes made to JavaScript. This included DHTML support and enhanced syntax. Unfortunately, they largely avoided standardizing the DOM.

## Incompatibilities

By 1997, difficulties with JavaScript were starting to become apparent largely because of changes made by Netscape and Microsoft® to enhance the language and browser capabilities. While the intent of both companies were honorable in most ways, their efforts resulted in problems with incompatibility that have plagued the community since. Most incompatibilities are not JavaScript issues but are specific to DOM. The JavaScript implementations of most popular web browsers usually follow the ECMAScript standard. The incompatibilities come about by trying to make a browser better than competing browsers. Where the Netscape and Mozilla™ based browser such as Firefox® and Flock™ support JavaScript, Microsoft® Internet Explorer supports JScript®. With that in mind, a safer way to develop JavaScript applications for conventional desktop browsers is to remain as compatible as possible to ECMA Script and stay clear of browser-specific DOMs.

## BlackBerry Browser and JavaScript

The BlackBerry® Browser supports JavaScript 1.0, 1.1, 1.2 and 1.3, and small subsets of JavaScript 1.4 and 1.5. It also supports the ECMA-262 ECMA Script Language Specifications.

The BlackBerry Browser has limited support for style sheets, does not support Dynamic HTML style sheets, and will ignore requests for special effects such as pop up menus.

Support is provided for the following JavaScript objects:

- blackberry
- blackberry.location (starting in BlackBerry Browser v4.1)
- navigator
- document
- form
- screen
- window

---

### Object: blackberry

---

#### blackberry.network

Contains a network identifier as shown below:

- GPRS
- CDMA
- iDEN™
- WLAN
- 3G

#### Example:

```
document.write("This BlackBerry device is  
communicating on the " + blackberry.network  
+ " wireless network.");
```

#### blackberry.location

The location object is only available on GPS-enabled BlackBerry devices.

The **blackberry.location.GPSSupported** property will return true if GPS is supported.

Use the **blackberry.location.latitude** property to get the current latitude, and the **blackberry.location.longitude** property to get the current longitude. Both properties are set to zero if GPS is unavailable.

Call **blackberry.location.refreshLocation()** before retrieving either property to retrieve the most current results. This method returns true on success.

Call **blackberry.location.onLocationUpdate()** to register a callback method that will be called when Location has been updated. Pass the callback method as the parameter. The method returns true on success.

Call **blackberry.location.setAidMode()** to set the GPS aid mode to be used by passing 0 for Cellsite, 1 for Assisted or 2 for Autonomous.

**Example:**

```
if ( blackberry.location.GPSSupported ) {
    // register callback
    blackberry.location.onLocationUpdate
        ( locationCB );

    setModeCellsite();
//setModeAssisted();
//setModeAutonomous();

} else {
    document.write("GPS not supported");
}

void setModeCellsite()
{
    // set to cellsite mode
    blackberry.location.setAidMode(0);

    // refresh properties
    blackberry.location.refreshLocation();

    document.write("<p>Cellsite Mode</p>" );
}

void setModeAssisted()
{
    // set to assisted mode
    blackberry.location.setAidMode(1);

    // refresh properties
    blackberry.location.refreshLocation();

    document.write("<p>Assisted Mode</p>" );
}

void setModeAutonomous()
{
    // set to autonomous mode
    blackberry.location.setAidMode(2);

    // refresh properties
    blackberry.location.refreshLocation();

    document.write("<p>Autonomous Mode</p>" );
}

// called when location object changes
void locationCB()
{
    document.write("<p>Latitude " +
        blackberry.location.latitude + "</p>");
    document.write("<p>Longitude " +
        blackberry.location.longitude + "</p>");
}
```

---

## Object: navigator

---

This object provides information such as name, version, language, mime types, plug ins, and whether Java™ is supported.

### Properties

**navigator.appCodeName** is set to the browser code name, which is currently "Mozilla".

**navigator.appName** is set to the browser name, "BlackBerry."

**navigator.appVersion** is set to the current browser revision. This takes the form of "4.1.0".

**navigator.language** is set to the two-letter language code of the default language set on the BlackBerry device.

**navigator.mimeTypes** is an array of supported MIME types that includes the following properties:

- navigator.mimeTypes.length
- navigator.mimeTypes[index]
- navigator.mimeTypes[index].type
- navigator.mimeTypes[index].desc
- navigator.mimeTypes[index].suffixes

The "index" range is 0 to length-1.

"length" is an Integer in the range of 1 through n.

"type", "desc" and "suffixes" are Strings.

### The following code fragment demonstrates use:

```
var numTypes = navigator.mimeTypes.length;
for ( el = 0; el < numTypes; el++ )
{
    document.write("Element " + el + "<br />");
    document.write("MIME Type " +
        navigator.mimeTypes[el].type + "<br />");
    document.write("Description " +
        navigator.mimeTypes[el].desc + "<br />");
    document.write("Suffixes " +
        navigator.mimeTypes[el].suffixes);
    document.write("<p></p>");
}
```

**navigator.platform** is set to "BlackBerry"

**navigator.plugins** is set to an array of installed plug-ins.

**navigator.userAgent** provides an extract of the user agent from the user-agent header of the HTTP header. This is used by servers to identify the client browser. I tested it with my BlackBerry® 8700 Series handheld with the following result:

```
Mozilla /4.0 BlackBerry8700/4.2.0
Profile/MIDP-2.0 Configuration/CLDC-1.1
VendorID/-1
```

## Methods

There are five navigator methods:

Call **navigator.javaEnabled()** to test if Java is enabled. This always returns true.

**navigator.plugins.refresh()** has not been implemented. The intent would be to make newly installed plugins available and then update the “plugins” array. When implemented, you would pass true to reload all open documents that contain embedded objects.

Call **navigator.preferences()** to read BlackBerry device preferences based on the passed parameter. The syntax is preferences(parameter) where the following list details all possible parameters:

general.always_load_images	
security.enable_java	returns true
javascript.enabled	
browser.enable_style_sheets	
autoupdate.enabled	returns false
network.cookie.cookieBehavior	
network.cookie.warnAboutCookies	returns true

**navigator.savePreferences()** not been implemented.

Call **navigator.taintEnabled()** to test if data tainting is enabled. Note that this returns false.

If data tainting had been enabled, JavaScript in one window could access properties of another window even when it was loaded from another server. Property values or other data can be tainted (marked) to be either secure or private. If this is done, and if taint has been enabled, then JavaScript cannot pass the tainted values on to any server without the author’s permission.

---

## Object: document

---

The document object provides access to elements within an HTML page such as properties of forms, links and colors.

### Properties

The following properties are not currently supported:

- **document.anchors**  
An array of entries for each anchor in the document.
- **document.applets**  
An array of entries for each applet in the document.
- **document.classes**  
Creates a Style object used to set the styles of HTML tags with a specific CLASS attribute.
- **document.embeds**  
An array of entries for each plug-in in the document.

- **document.ids**  
Creates a Style object used to set the style of individual HTML tags.
- **document.images**  
An array of entries for each image in the document.
- **document.plugins**  
An array of entries for each plug-in in the document.
- **document.tags**  
Creates a Style object used to set the styles of HTML tags.

The following are supported properties:

**document.alinkColor** is a read-only property used to determine the color of links within the current page. Colors are returned as either text values (such as red), or as hexadecimal triplets in RGB format (such as #FF0000). The return format depends on how the link color is represented within the page.

**document.bgColor** is a read-only property used to determine the background color of the current page. Colors can be returned as text (such as blue) or hexadecimal triplets (such as #0000FF)

**document.cookie** is used to create or update cookies associated with the current page, or view visible and expired cookies associated with the current page.

#### To view cookies:

```
document.write(document.cookie);
```

#### To create a cookie:

```
function createCookie(name,value,days) {
  if (days) {
    var date = new Date();
    date.setTime(date.getTime()+
      (days*24*60*60*1000));
    var expires = ";expires=" +
      date.toGMTString();
  } else var expires = "";
  document.cookie = name + "=" +
    value + expires + ";path=/";
}
```

**document.domain** is a read-only property used to determine the domain that served the current page.

**document.fgColor** is a read-only property used to determine the text color within the current page.

**document.formName** is used to access and control a specific <form> within the active page. The following code sample shows how to use this property by submitting a form:

```
document.loginForm.submit();
```

**document.forms** is a read-only property that provides access to any <form> object in the active document. Use **document.forms.length**; to determine the length of the index. Accessing a specific form elements could take the form of **document.forms[0].submit()**;

**document.height** returns the height of the BlackBerry device screen, in pixels.

Use **document.lastModified** to extract the date from the HTTP header when the current document was last modified.

**document.linkColor** provides the color used by the active page to identify hyperlinks. The color is either represented by a text string or the hexadecimal RGB triplet depending on how the page has been coded.

Use **document.referrer** to get the URL of the referrer through which the user arrived at the current page. This will be set to null if the user arrived at the current page without use of a referrer.

**document.title** is used to read or set the title of the current page.

**document.URL** is a read-only property that is set to the URL of the current document.

**document.vlinkColor** provides the color used by the active page to identify previously visited links.

**document.width** is set to the width of the BlackBerry device screen, in pixels.

## Methods

The following methods are not currently supported:

- **document.captureEvents()**  
Sets document to capture all events of the specified type
- **document.contextual()**  
Uses contextual selection criteria to set a Style object that can set the style of individual HTML tags.
- **document.getSelection()**  
Returns a string set to the text of the current selection.
- **document.handleEvent()**  
Invokes the handler for the specified event.
- **document.releaseEventst()**  
Passes a captured event to the normal event hierarchy.
- **document.routeEvent()**  
Passes a captured event to the normal event hierarchy.

The following are supported methods:

**document.close()** closes an output stream and forces data to display.

**document.open()** opens a stream to collect the output of **document.write()** or **document.writeln()** methods.

**document.write()** writes one or more HTML expressions to a document in the specified window.

**document.writeln()** writes one or more HTML expressions to a document in the specified window then appends a new-line character.

### Example:

```
function doThis() {
```

```
    var string1 = "Hi mom!";  
    var string2 = "Bye mom!";  
    document.open();  
    document.write(string1 + "<P>" + string2);  
    document.close();  
}
```

---

## Object: form

---

The form object provides access to elements of an HTML form. This object takes the name of the <form> within the current page to access elements from that form. In the following descriptive text, replace the name, "formName" with the name of the <form> that you want to access.

## Properties

**formName.action** is used to either read or set the URL where the <form> will be submitted.

### Examples:

```
var theAction = myForm.action;  
myForm.action = "http://www.rim.com/form.php";  
myForm.action = "mailto:nospam@rim.com";
```

**formName.elements** is used to retrieve a list of elements within the <form>.

Use **formName.elements.length** or **formName.length** to determine the size of the list, and **formName.elements[index]** to access specific elements in the <form>.

### Example:

```
for (el=0; el<myForm.elements.length; el++ )  
    document.write(myForm.elements[el]);
```

**formName.method** returns the value of the METHOD attribute of the HTML <form> element. This will either be POST or GET. It will be set to the method used to submit the form data to the server.

**formName.name** returns the NAME attribute of the HTML <form>. This can also be accomplished by using the attribute **document.forms[ELEMENT].name**.

**formName.target** returns or sets the target window that <form> responses are sent to after a <form> has been submitted. Setting the target doesn't make a great deal of sense for BlackBerry devices because the BlackBerry Browser is a single document interface that never displays more than one browser window at the same time. Therefore, setting the target will have no effect.

## Methods

**formName.handleEvent()** has not been implemented. If it was then the method would invoke the event handler for the specified event.

**formName.reset()** will reset the default values of any elements in the specified <form>.

**formName.submit()** will submit the <form> to the server.

## Event Handlers

**formName.onClick** specifies the event that will be triggered when an “on click” event occurs. You can use `onClick` to determine the current event handler or set the event handler.

### Example:

```
<html>
  <body>
    <script type="text/javascript"
      language="JavaScript">
<!--
// Show field data when button is selected
function showData(){
  var fName = document.myForm.first.value;
  var lName = document.myForm.last.value;
  alert("The name entered is " +
    fName + " " + lName);
}
// -->
</script>
<form name="myForm">
  <p>
    Name - First:
    <input type="text" name="first"
      size="25">
    &nbsp;  Last:
    <input type="text" name="last"
      size="25">
  </p>
  <p>
    Select button to view entry
    <input type="button" value="isokay"
      name="showdata" onClick='showData()' '>
  </p>
</form>
</body>
</html>
```

The **formName.onReset** event handler will execute a specified section of JavaScript code when the user resets a form by selecting a Reset button. You can either read or write the setting using this handler.

### Complex Example:

```
<html>
  <body>
    <script type="text/javascript"
      language="JavaScript">
<!--
// Complex method to display a message when
// the onReset event handler is called
function resetCalled(){
  document.msgForm.message.value =
    "Field data has been cleared";
}
// -->
</script>
<form name="dataForm"
  onReset='resetCalled()' '>
```

```
<p>
  Field: <input type="text"
    name="myField" size="30">
</p>
<p>
  <input type="reset" value="Reset"
    name="Reset"
  </p>
</form>
<form name="msgForm">
  <p>
    Status: <input type="text"
      name="message" size="30">
  </p>
</form>
</body>
</html>
```

### Less Complex Example:

```
<form name="myForm"
  onReset="alert('Defaults restored')">
  Country:
  <p>
    <input type="text" name="country"
      value="USA" size="10">
  </p>
  <p>
    <input type="reset" value="Reset"
      name="doReset">
  </p>
</form>
```

The **formName.onSubmit** event handler will execute the specified JavaScript code when the user submits a form.

### Example:

```
<html>
  <body>
    <script type="text/javascript"
      language="JavaScript">
<!--
function isDone() {
  alert("Form has been submitted");
}
// -->
</script>
<form name="mainForm" onSubmit='isDone()' '>
  <p>
    <b>Suggestions, please!</b>
  </p>
  <p>
    <textarea name="suggest" rows="10"
      cols="30">
  </textarea>
  </p>
  <p>
    <input type="submit" value="Submit">
  </p>
</form>
</body>
</html>
```

---

## Object: screen

---

The screen object returns information about the BlackBerry device display such as height, width, color depth and more.

### Properties

**availHeight** provides the screen height in pixels.

**availWidth** provides the screen width in pixels.

**colourDepth** provides the bit depth of the color palette. If no palette is used, this will reflect the value of **pixelDepth**.

**height** provides the screen height in pixels.

**pixelDepth** provides the color resolution, in bits per pixel, of the BlackBerry device screen.

**width** provides the screen width in pixels.

---

## Object: window

---

The window object is automatically created when the BlackBerry Browser parses a <body> or <frame> tag and will return a wide variety of information.

### Properties

The following properties are not currently supported:

crypto	personalbar
locationbar	screenX
menubar	screenY
name	scrollbars
opener	status
pageXOffset	statusbar
pageYOffset	toolbar

The following properties are supported:

**window.blackberry** is used to access the blackberry object. As previously discussed, **window.blackberry.network** contains an identifier for the network used by the BlackBerry device, where **window.blackberry.location** is an object used to get navigational information on GPS-enabled BlackBerry devices.

**window.closed** always returns false to indicate that the window instance has not been closed.

**window.defaultStatus** is used to set the default message displayed in the status bar.

The syntax is:

```
window.defaultStatus = "your message";
```

**window.document** provides access to the document object contained within the window. Please refer to the section in this article that describes the document object.

**window.frames** provides an array of each frame instance created with the <frame> tag within the current document. Array entries can be referenced by index number or the name assigned by the name attribute of the <frame> tag. **window.frames.length** is used to determine the size of the index.

---

*Note: The BlackBerry Browser does not support frames. Instead, a list of frames within the document are displayed for the user to select one frame to open.*

---

### Example:

```
// display the name of each frame
for (var el=0; el<=window.frames.length; el++)
    document.write("<p>Frame #" + el + ": " +
        window.frames[el].name + "</p>");
```

**window.history** is a core JavaScript object. **window.history[index]** provides access to an array containing the URLs and names of visited pages. Array entries are referenced by index number, with **window.history.length** used to determine the size of the index. The URL of the active document is accessed via **window.history.current**. Use **window.history.next** to determine the next URL as it relates to the current URL. Use **window.history.previous** to determine the previous URL in relation to the current URL. There are also three related methods, **window.history.back()**, **window.history.forward()** and **window.history.go()**, will be discussed later in this section.

**window.innerHeight** and **window.innerWidth** return the screen height and width respectively in pixels.

**window.length** returns the number of frames in the parent window. See **window.history** for more information.

**window.location** retrieves or sets the URL of the active window.

### Example:

```
<html>
  <head>
    <title>Redirection</title>
  </head>
  <body>
    <script language="JavaScript">
      window.location = "http://www.rim.com/";
    </script>
  </body>
</html>
```

**window.offscreenBuffering** returns false to indicate that the BlackBerry Browser does not buffer data offscreen before display.

**window.outerHeight** and **window.outerWidth** provides the screen height and width in pixels.

**window.pageXOffset** and **window.pageYOffset** are both set to zero (0). They reflect the current horizontal and vertical pixel location of the top-left corner of the document within the window.

**window.parent** provides a reference to the parent window. If no parent window exists then it will point to the current active window.

The syntax to use this is either **window.parent.property** or **window.parent.method** where the “**property**” and “**method**” is any window object property and method. Note that this is a writable property.

**window.screenX** and **window.screenY** both return zero (0).

**window.self** provides a reference to the current active window. The syntax to use this is either **window.self.property** or **window.self.method** where the “**property**” and “**method**” is any window object property and method. Note that this is a writable property.

**window.top** contains a reference to the top window, which is the current window when using a single document interface (SDI) browser such as the BlackBerry Browser. When writing to this property, the syntax is:

```
window.top.propertyName
window.top.methodName
```

... where *propertyName* is any property associated with the Window object, and *methodName* is any method associated with the Window object.

## Methods

The following methods are not currently supported:

blur()	releaseEvents()
captureEvents()	resizeBy()
disableExternalCapture()	resizeTo()
enableExternalCapture()	routeEvent()
find()	scroll()
focus()	scrollBy()
handleEvent()	scrollTo()
moveBy()	setHotKeys()
moveTo()	setZOptions()
print()	

The following methods are supported:

Call **window.alert()** (“*message*”) to display a standard alert dialog box. Note that *message* can be a string or a property of an existing object.

**window.atob()** and **window.btoa()** are related methods.

Call **window.btoa()** to encode potentially problematic data such as data within the ASCII range of 0 through 31 into

Base64 format. Afterwards, transmit the encoded data and then use **window.atob()** to decode it into original form. A simple example is shown below:

```
// encode passed string into Base64 format
base64Data = btoa("frumious Bandersnatch!");
// decodeBase64 data into original format
sourceData = atob(base64Data);
```

Call **window.back()** to return to the previous URL in the history list.

Call **window.clearInterval()** to clear the interval that is passed to the method. The passed interval has to be previously defined using the **window.setInterval()** method. A working example is shown below:

```
<html>
<head>
  <script language="JavaScript"
    type="text/javascript">
<!--
var countValue = 1;

// display the count
function displayCount() {
  document.theForm.data.value = countValue++;
}

// stop the count
function endCount() {
  window.clearInterval(intervalID);
}

// set interval to call the display
// function every second
var intervalID =
window.setInterval("displayCount()", 1000)
// -->
  </script>
</head>
<body onload="displayCount()">
  <form name="theForm">
    <input type="text" size="3" value=""
      name="data">
    <input type="button" value="Stop"
      onclick="endCount()">
  </form>
</body>
</html>
```

**window.clearTimeout()** and **window.setTimeout()** are related methods.

**window.setTimeout()** sets a timeout period in milliseconds. The expression or function passed to the method is called after the timeout period elapses. This timeout period can be cleared by calling the **clearTimeout()** method. A simple example is shown below:

```
<html>
<head>
  <script language="JavaScript"
    type="text/javascript">
<!--
```

```

// show the time
function showTime() {
    toShow = new Date();
    toShow = toShow.getHours() + ":" +
        toShow.getMinutes() + ":" +
        toShow.getSeconds();
    document.theForm.data.value = toShow;
}

// clear timeout to stop display of the time
function stopTime() {
    window.clearTimeout(timeoutID);
}

// set timeout period to 2 seconds
var timeoutID =
    window.setTimeout("showTime()", 2000);
// -->
</script>
</head>
<body>
    <form name="theForm">
        <input type="text" size="10" value=""
            name="data">
        <input type="button" value="Stop"
            onclick="stopTime()">
        <input type="button" value="Show Time"
            onclick="showTime()">
    </form>
</body>
</html>

```

Calling **window.close()** will return to the previous URL in the history list.

Call **window.confirm("message")** to display a confirmation dialog box with the "message" passed displayed in the box. The dialog box contains an OK and a Cancel button, and will return true if OK is clicked and false if Cancel is clicked.

Call **window.home()** to return the user to the default BlackBerry Browser home page.

Call **window.open("URL")** to open the passed URL in a new window. This method returns the name of the new window. For example:

```

theWindow = window.open("http://www.rim.com");
window.close(theWindow);

```

Call **window.prompt()** to display a prompt dialog box. You must pass a text message to display, and can also pass an optional parameter to set the default value for the user. For example:

```

<html>
    <body>
        <form>
            <input type="button" value="Prompt"
                onclick="window.prompt('Number:', 25)">
        </form>
    </body>
</html>

```

Call **window.stop()** to stop the current download.

# Profiling BlackBerry Devices for Microsoft .NET Mobile Controls

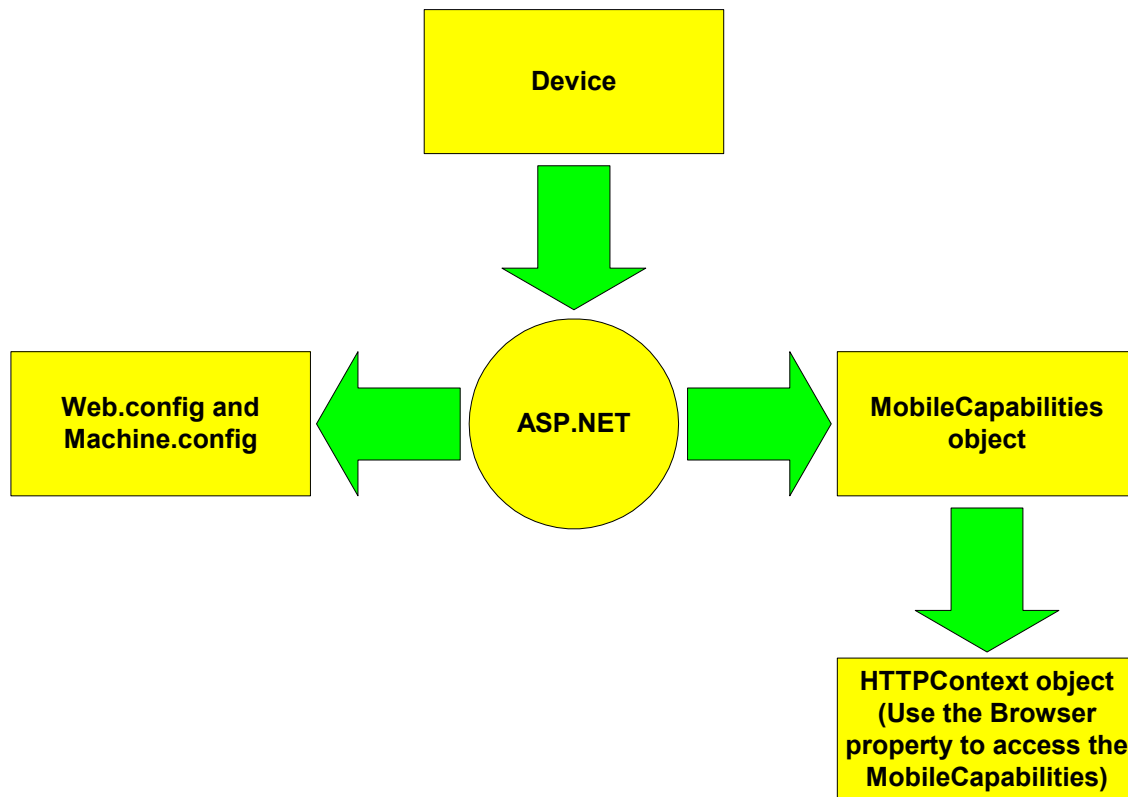
Garry Seifried, Research In Motion

In this article I will describe the process of creating a device profile for BlackBerry® devices to work with the Microsoft® .NET Mobile Controls framework.

I will use the Microsoft ASP.NET Device Profiling Tool to perform a series of tests using a BlackBerry device to create a profile. This will be used by the Mobile Controls framework to create Mobile Controls that work according to the capabilities of the BlackBerry device.

## How Device Profiling Works

To add support for a new device, you must understand how the ASP.NET runtime identifies a device and renders the most appropriate markup for the device. The following diagram shows the process that the ASP.Net runtime follows to determine the device type.



The diagram above shows a device request for a mobile web forms page. An instance of the `HttpRequest` class is created after the ASP.NET runtime receives the request. The `HttpRequest` object exposes a `MobileCapabilities` object through the `browser` property that the ASP.NET runtime uses to store information about the capabilities of the requesting device. The ASP.NET runtime then checks for device attributes in the application configuration file (`Web.config`) and the system configuration file (`Machine.config`) for `<browserCaps>` sections.

The configuration sections contain several `<case>` sections, one for each supported device browser.

The `<case>` sections use a regular expression to match the `HTTP_USER_AGENT` environment variable, which the requesting browser passes as an HTTP request header. If a match occurs, then the ASP.NET runtime has found a device type configuration. The runtime then uses the information in the `<case>` section of the configuration file to populate the `MobileCapabilities` object.

If the ASP.NET runtime does not find a device configuration in one of the <case> sections, it will populate the object, MobileCapabilities, with the default configuration section values. Note that by default, it identifies the requesting device as an HTML 3.2 browser of type Unknown.

### Mobile Controls QuickStart

The ASP.NET Mobile Controls QuickStart contains a series of samples and supporting commentary designed to quickly acquaint you with the ASP.NET mobile controls. The QuickStart samples are designed to be short, easy-to-understand illustrations of features of the mobile controls. Mobile Controls QuickStart are available at:

[http://msdn2.microsoft.com/en-us/library/c55eyak3\(VS.80\).aspx](http://msdn2.microsoft.com/en-us/library/c55eyak3(VS.80).aspx)

### Roadmap for Mobile Web Development with ASP.NET

Before adopting Mobile Web development with ASP.NET, you should review the roadmap located at:

<http://www.asp.net/mobile/mobileroadmap.aspx>

### Microsoft ASP.NET Device Profiling Tool

The ASP.NET device profiling tool was created for device users to create device profiles as additional devices come to market. By design, the Mobile Controls framework is extensible so that additional device support is not dependent on support from Microsoft.

The device profiling tool is available at:

<http://www.asp.net/mobile/profile/default.aspx>

Device Profile Tool Instructions are available at:

<http://www.asp.net/mobile/profile/help.htm>

The first page has a heading of “ASP.NET Device Profiling.” Read the text then select the “Click to Begin” button.

The next “Log In” page requires that you either log in, get a replacement password, or become a registered user. If you chose to become a registered user, a simple form is provided that requires you to provide your name, an alias and an email address.

After creation of your account, you will automatically be signed in and sent an email confirming your account. At this stage, return to the device profile page and log in:

<http://www.asp.net/mobile/profile/default.aspx>

On first time use, a notice will appear stating that you have no profiled devices and should click through the link, “Start Profiling a New Device.”

On the next “Device Profiling” page, you will need to select what markup language is best supported on your device. Follow the steps to create a profile for your BlackBerry device. Note that this process will be detailed later in this article.

After you have completed a device profile, each time you log in, you will be presented with the existing device profiles and the ability to profile a new device. Each device profile can be viewed, modified or downloaded for your web.config on an application level or machine.config on a machine level.

Ideally, you should use the BlackBerry device simulator to do the work of the device profile although a live BlackBerry device can be used. Make sure to start the BlackBerry MDS Simulator to allow the BlackBerry Browser to perform the device profile tests. Selecting a BlackBerry JDE device simulator allows you to create a device profile for the desired device and version of device software. Once you have created a device profile, you will have a better understanding of the relationships between the device capabilities and the questions presented by the device profiling tool. This may make it easier for you to modify an existing device profile (after it has been downloaded) and modify the device specific properties known to be different between devices. This is especially true for BlackBerry devices running the same BlackBerry device software versions, where the physical capabilities of the BlackBerry device vary by screen size as opposed to whether cascading-style sheets are supported, for example.

The device profile can be viewed after creation.

### Downloading the completed Profile

The download page displays instructions for incorporating the new profile into the .NET Framework 1.1. The file is named according to the Device ID assigned at the start of device profiling.

### ASP.NET Mobile Control Device Profile for BlackBerry 8700 Series Handheld 4.1.0

The following is an example of a configuration file for a BlackBerry® 8700 Series handheld emulator profiled using the Device Profiling tool. The generated device configuration file has a regular expression in the <case> element that matches with the HTTP\_USER\_AGENT string of the device to allow the ASP.Net runtime to identify the device.

```
<?xml version="1.0" encoding="utf-8"?>
<browserCaps>
  <use var="HTTP_USER_AGENT" />
  <filter>
    <case match="BlackBerry8700\4\1\0 Profile\MIDP-2\0 Configuration\CLDC-1\1 VendorID\-1">
      breaksOnInlineElements = "False"
      browser = "BlackBerry Browser"
      canInitiateVoiceCall = "True"
      canSendMail = "True"
      cookies = "True"
      inputType = "keyboard"
    </case>
  </filter>
</browserCaps>
```

```

isColor = "True"
javascript = "True"
maximumHrefLength = "5000"
maximumRenderedPageSize = "256000"
mobileDeviceManufacturer = "Research In Motion"
mobileDeviceModel = "8700"
preferredImageMime = "image/jpeg"
preferredRenderingMime = "application/vnd.wap.xhtml+xml"
preferredRenderingType = "xhtml-mp"
requiresAbsolutePostbackUrl = "False"
requiresCommentInStyleElement = "False"
requiresFullyQualifiedRedirectUrl = "False"
requiresHiddenFieldValues = "False"
requiresHtmlAdaptiveErrorReporting = "False"
requiresOnEnterForwardForCheckboxLists = "False"
requiresPostRedirectionHandling = "False"
requiresXhtmlCssSuppression = "False"
screenBitDepth = "16"
screenCharactersHeight = "10"
screenCharactersWidth = "35"
screenPixelsHeight = "240"
screenPixelsWidth = "320"
supportsAccessKeyAttribute = "False"
supportsBodyClassAttribute = "True"
supportsBodyColor = "True"
supportsBold = "True"
supportsCss = "True"
supportsDivAlign = "True"
supportsDivNoWrap = "False"
supportsEmptyStringInCookieValue = "True"
supportsFontColor = "True"
supportsFontName = "True"
supportsFontSize = "True"
supportsItalic = "True"
supportsNoWrapStyle = "False"
supportsQueryStringInFormAction = "True"
supportsRedirectWithCookie = "True"
supportsSelectFollowingTable = "True"
supportsStyleElement = "True"
supportsTitleElement = "True"
supportsUrlAttributeEncoding = "True"
tables = "True"
</case>
</filter>
</browserCaps>

```

## Adding a Device Entry

The entry defined for each particular browser is shown as a 'case' section in the <browserCaps> section of the config file:

```

<case match=
"BlackBerry((?'model'\d+)/('version'((?'major'
\d+).(?'minor'\d+).(?'extra'\d+)))">

```

This will define an entry for a BlackBerry device and create variables for model, version, version major, version minor and extra text.

Using these variables and a <filter> construct allows a single BlackBerry definition to be created that permits variations between devices. This is defined in the Filter Match section.

## Adding Filter Matches for device differences

The following are filters for the screen size and color capabilities for the BlackBerry® 6200 Series, BlackBerry® 7200 Series, BlackBerry® 7500 Series, BlackBerry® 7100 Series, BlackBerry® 7700 Series, and BlackBerry® 8700 Series of devices:

```

<filter match="62???" with="{model}">
  isColor = "false"
  screenPixelsWidth = 160
  screenPixelsHeight = 100
  screenBitDepth = 1
</filter>
<filter match="72???" with="{model}">
  isColor = "true"
  screenPixelsWidth = 240
  screenPixelsHeight = 160
  screenBitDepth = 16
</filter>
<filter match="75???" with="{model}">
  isColor = "true"

```

```

screenPixelsWidth = 240
screenPixelsHeight = 160
screenBitDepth = 16
</filter>
<filter match="71??" with="{model}">
  isColor = "true"
  screenPixelsWidth = 240
  screenPixelsHeight = 260
  screenBitDepth = 16
</filter>
<filter match="77??" with="{model}">
  isColor = "true"
  screenPixelsWidth = 240
  screenPixelsHeight = 240
  screenBitDepth = 16
</filter>
<filter match="87??" with="{model}">
  isColor = "true"
  screenPixelsWidth = 320
  screenPixelsHeight = 240
  screenBitDepth = 16
</filter>

```

## Profiling a New Device

### Device Headers

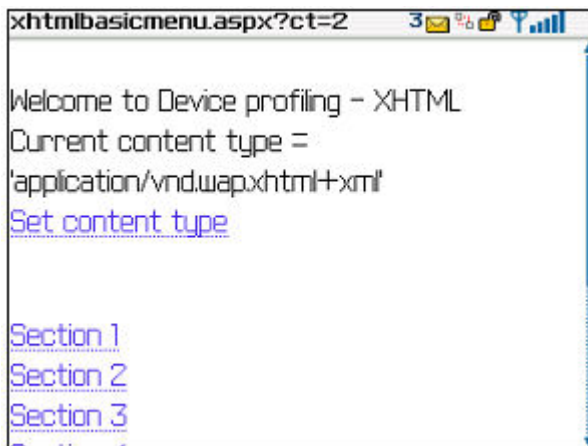
For the device headers, you navigate to the following URL from the BlackBerry Browser.

```
http://www.asp.net/Mobile/Profile/
xhtmlbasicmenu.aspx?ct=<0,1,2>.
```

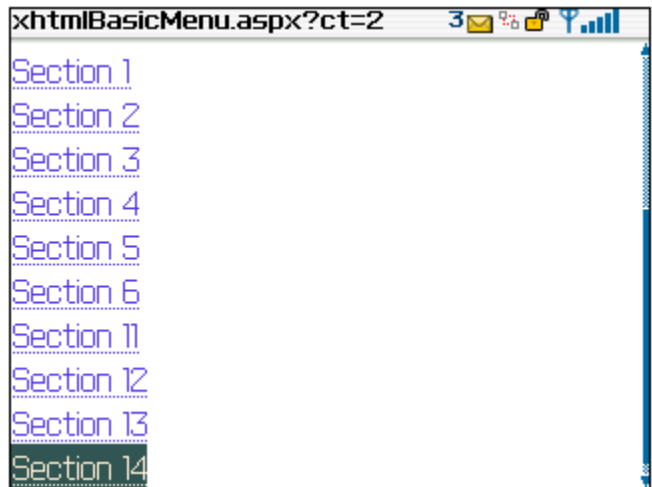
The parameter to the “ct” variable is one of **0**, **1** or **2** according to the device content-type header:

- 0 - application/xhtml+xml (default)
- 1 - text/html
- 2 - application/vnd.wap.xhtml+xml

Navigating to this URL using a BlackBerry 8700 Series simulator displays the content type and Sections 1 through 13 that contain the profile tests to be performed using the BlackBerry Browser:



The Device Profiling tool presents a number of sections 1-14 that are used to create the device profile. You continue through the instructions until all sections have been completed.



Following the instructions to section 1.0, test 1.0 will result in the header ID being shown. Entered this value in your desktop browser session to name the device profile.



## BlackBerry Device Mobile Profile

### HTTP Headers

A program that displays the HTTP headers from a BlackBerry Browser request can be used to get the values in the request header to obtain the BlackBerry device profile.

This sample code needs to be served by a Java Application Server from a URL such as:

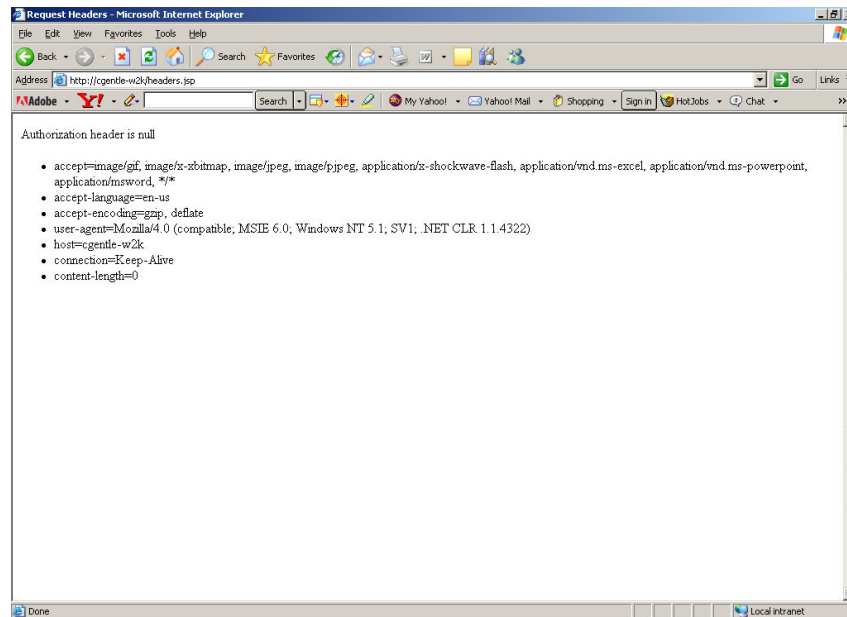
<http://localhost:8888/HDR/headers.jsp>:

```

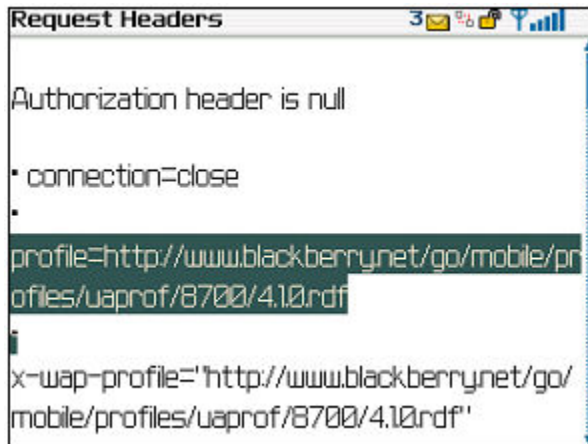
<%@ page contentType="text/html" %>
<%@ page import="java.util.*" %>
<%@ page import="javax.servlet.http.*" %>
<html>
  <head>
    <title>Request Headers</title>
  </head>
  <body>
    <ul>
<% Enumeration headerNames =
request.getHeaderNames();
while (headerNames.hasMoreElements()) {
  String headerName =
    (String)headerNames.nextElement();
  String headerValue =
    request.getHeader(headerName); %>
    <li><%=headerName%>=<%=headerValue%></li>
<% } %>
    </ul>
  </body>
</html>

```

Accessing this page from a desktop browser would show information relevant to the desktop browser:



However, the BlackBerry Browser includes additional header information that shows the profile:



### Download the device profile

Access the profile URL from a desktop browser to view the BlackBerry Device definition. This may help you to understand the device capabilities and complete the profile using the ASP.NET Device Profiling Tool.

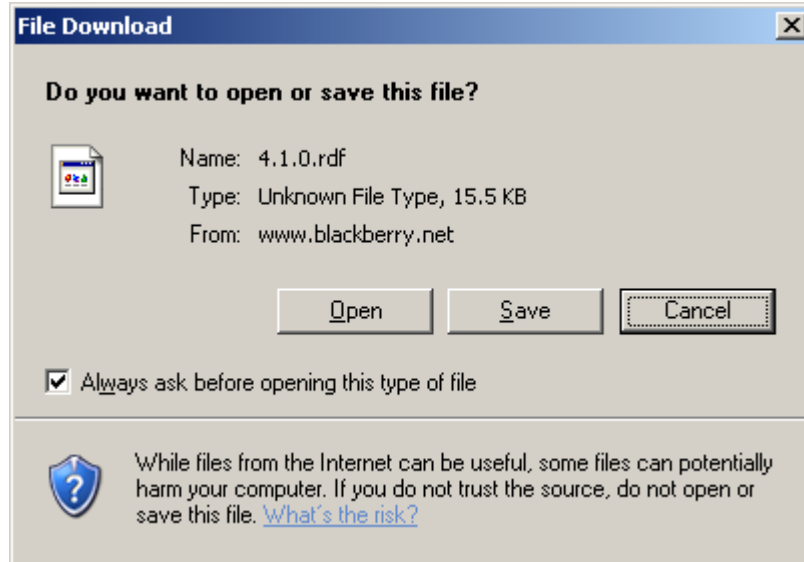
The profile URL is:

<http://www.blackberry.net/go/mobile/profiles/uaprof/8700/4.1.0.rdf>

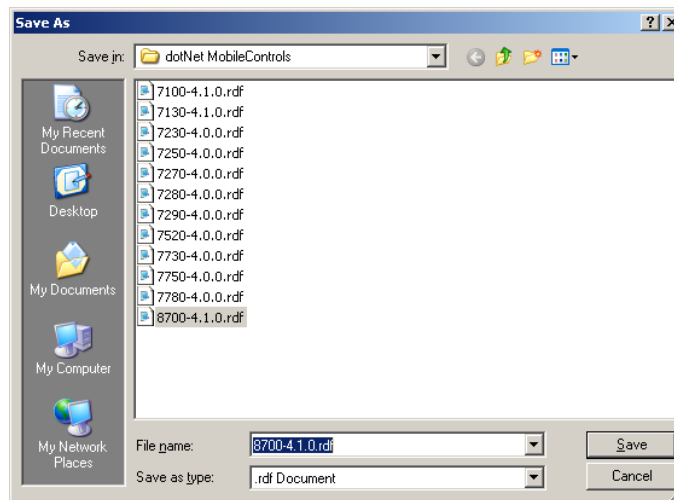
BlackBerry profiles exist for other devices and can be accessed using the same format for the URL:

- 8700 – for the BlackBerry 8700 Series handheld
- 4.1.0 – for the BlackBerry® Device Software version

The file is named using the software version and should have the BlackBerry device version added to the filename:



The naming convention is shown below:



### Device Profile – 8700-4.1.0.rdf

The device profile includes information that is needed to create a new device profile. For example, screen size and screen size character:

```
<prf:PixelAspectRatio>1x1</prf:PixelAspectRatio>  
<prf:ScreenSize>320x240</prf:ScreenSize>  
<prf:ScreenSizeChar>32x16</prf:ScreenSizeChar>
```

The 8700-4.1.0.rdf profile is shown here:

```
<?xml version="1.0" encoding="utf-8"?>  
<!--  
*  
* (c) 2006 Research In Motion Ltd.,  
* 295 Phillip Street  
* Waterloo, Ontario, Canada  
* N2L 3W8  
*  
* Version: 2
```

```

* Created: Mon May 08 14:33:45 EDT 2006
*
* Notes:
*     Elements that are not required by RIM Browser have been omitted from this template.
*     Any elements that are defined in the UAProf specification may be added to this document.
*
*
* Revision History:
* 1: Initial Version
* 2: XSLT'd Files
* 3: New models
*
*
-->
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:prf="http://www.openmobilealliance.org/tech/profiles/UAPROF/ccppschem-20021212#"
xmlns:mms="http://www.openmobilealliance.org/tech/profiles/MMS/ccppschem-20050301-MMS1.2#"
xmlns:date="java.util.Date">
  <rdf:Description rdf:ID="DeviceProfile">
<!-- Hardware Platform Description -->
  <prf:component>
    <rdf:Description rdf:ID="HardwarePlatform">
      <rdf:type rdf:resource="http://www.openmobilealliance.org/tech/profiles/UAPROF/
ccppschem-20021212#HardwarePlatform"/>
      <prf:BluetoothProfile>
        <rdf:Bag>
          <rdf:li>Headset Profile</rdf:li>
          <rdf:li>Handsfree Profile</rdf:li>
          <rdf:li>Serial Port Profile</rdf:li>
        </rdf:Bag>
      </prf:BluetoothProfile>
      <prf:BitsPerPixel>16</prf:BitsPerPixel>
      <prf:ColorCapable>Yes</prf:ColorCapable>
      <prf:CPU>XScale</prf:CPU>
      <prf:InputCharSet>
        <rdf:Bag>
          <rdf:li>ISO-8859-1</rdf:li>
          <rdf:li>UTF-8</rdf:li>
          <rdf:li>UTF-16be</rdf:li>
          <rdf:li>Windows-1252</rdf:li>
          <rdf:li>US-ASCII</rdf:li>
        </rdf:Bag>
      </prf:InputCharSet>
      <prf:ImageCapable>Yes</prf:ImageCapable>
      <prf:Keyboard>Qwerty</prf:Keyboard>
      <prf:Model>BlackBerry 8700</prf:Model>
      <prf:NumberOfSoftKeys>0</prf:NumberOfSoftKeys>
      <prf:OutputCharSet>
        <rdf:Bag>
          <rdf:li>ISO-8859-1</rdf:li>
          <rdf:li>UTF-8</rdf:li>
          <rdf:li>UTF-16be</rdf:li>
          <rdf:li>Windows-1252</rdf:li>
          <rdf:li>US-ASCII</rdf:li>
        </rdf:Bag>
      </prf:OutputCharSet>
      <prf:PointingResolution>Character</prf:PointingResolution>
      <prf:PixelAspectRatio>1x1</prf:PixelAspectRatio>
      <prf:ScreenSize>320x240</prf:ScreenSize>
      <prf:ScreenSizeChar>32x16</prf:ScreenSizeChar>
      <prf:SoundOutputCapable>Yes</prf:SoundOutputCapable>
      <prf:StandardFontProportional>Yes</prf:StandardFontProportional>
      <prf:TextInputCapable>Yes</prf:TextInputCapable>
      <prf:Vendor>Research In Motion Ltd.</prf:Vendor>
    </rdf:Description>
  </prf:component>
</rdf:Description>

```

```

        <prf:VoiceInputCapable>Yes</prf:VoiceInputCapable>
    </rdf:Description>
</prf:component>
<!-- Software Platform Description -->
<prf:component>
    <rdf:Description rdf:ID="SoftwarePlatform">
        <rdf:type rdf:resource="http://www.openmobilealliance.org/tech/profiles/UAPROF/
            ccppschemata-20021212#SoftwarePlatform"/>
        <prf:AcceptDownloadableSoftware>Yes</prf:AcceptDownloadableSoftware>
        <prf:CcppAccept>
            <rdf:Bag>
                <rdf:li>text/plain</rdf:li>
                <rdf:li>text/html</rdf:li>
                <rdf:li>text/vnd.wap.wml</rdf:li>
                <rdf:li>text/vnd.wap.wmlscript</rdf:li>
                <rdf:li>application/vnd.wap.wmlc</rdf:li>
                <rdf:li>application/vnd.wap.wmlscriptc</rdf:li>
                <rdf:li>application/vnd.rim.html</rdf:li>
                <rdf:li>application/xhtml+xml</rdf:li>
                <rdf:li>application/vnd.wap.xhtml+xml</rdf:li>
                <rdf:li>application/vnd.wap.mms-message</rdf:li>
                <rdf:li>application/vnd.oma.drm.message</rdf:li>
                <rdf:li>application/smil</rdf:li>
                <rdf:li>image/vnd.wap.wbmp</rdf:li>
                <rdf:li>image/png</rdf:li>
                <rdf:li>image/gif</rdf:li>
                <rdf:li>image/jpeg</rdf:li>
                <rdf:li>image/jpeg</rdf:li>
                <rdf:li>image/vnd.rim.png</rdf:li>
                <rdf:li>audio/midi</rdf:li>
                <rdf:li>audio/sp-midi</rdf:li>
                <rdf:li>audio/x-mid</rdf:li>
                <rdf:li>audio/x-midi</rdf:li>
                <rdf:li>audio/mid</rdf:li>
                <rdf:li>audio/x-wav</rdf:li>
                <rdf:li>audio/wav</rdf:li>
                <rdf:li>audio/mpeg</rdf:li>
                <rdf:li>audio/x-mpeg</rdf:li>
                <rdf:li>audio/amr</rdf:li>
                <rdf:li>application/vnd.wap.multipart.mixed</rdf:li>
                <rdf:li>application/vnd.wap.multipart.alternative</rdf:li>
                <rdf:li>application/vnd.wap.multipart.related</rdf:li>
                <rdf:li>multipart/mixed</rdf:li>
                <rdf:li>multipart/alternative</rdf:li>
                <rdf:li>multipart/related</rdf:li>
                <rdf:li>text/vnd.sun.j2me.app-descriptor</rdf:li>
                <rdf:li>application/java-archive</rdf:li>
                <rdf:li>application/vnd.rim.cod</rdf:li>
                <rdf:li>application/vnd.wap.sic</rdf:li>
                <rdf:li>application/vnd.wap.slc</rdf:li>
                <rdf:li>application/vnd.wap.coc</rdf:li>
                <rdf:li>application/vnd.wap.sia</rdf:li>
                <rdf:li>application/x-vnd.rim.pme.b</rdf:li>
                <rdf:li>application/x-vnd.rim.pme</rdf:li>
                <rdf:li>image/pme</rdf:li>
            </rdf:Bag>
        </prf:CcppAccept>
        <prf:CcppAccept-Charset>
            <rdf:Bag>
                <rdf:li>ISO-8859-1</rdf:li>
                <rdf:li>UTF-8</rdf:li>
                <rdf:li>UTF-16be</rdf:li>
                <rdf:li>Windows-1252</rdf:li>
                <rdf:li>US-ASCII</rdf:li>
            </rdf:Bag>
        </prf:CcppAccept-Charset>
    </rdf:Description>
</prf:component>

```

```

</prf:CcppAccept-Charset>
<prf:CcppAccept-Language>
  <rdf:Seq>
    <rdf:li>de</rdf:li>
    <rdf:li>en</rdf:li>
    <rdf:li>en_GB</rdf:li>
    <rdf:li>es</rdf:li>
    <rdf:li>fr</rdf:li>
    <rdf:li>it</rdf:li>
    <rdf:li>pt</rdf:li>
  </rdf:Seq>
</prf:CcppAccept-Language>
<prf:DownloadableSoftwareSupport>
  <rdf:Bag>
    <rdf:li>application/vnd.rim.cod</rdf:li>
    <rdf:li>text/vnd.sun.j2me.app-descriptor</rdf:li>
  </rdf:Bag>
</prf:DownloadableSoftwareSupport>
<prf:JavaEnabled>Yes</prf:JavaEnabled>
<prf:JavaPlatform>
  <rdf:Bag>
    <rdf:li>CLDC</rdf:li>
    <rdf:li>MIDP</rdf:li>
    <rdf:li>MIDP/1.0-compatible</rdf:li>
    <rdf:li>Profile/MIDP-2.0</rdf:li>
    <rdf:li>Configuration/CLDC-1.0</rdf:li>
    <rdf:li>Configuration/CLDC-1.1</rdf:li>
  </rdf:Bag>
</prf:JavaPlatform>
<prf:JVMVersion>
  <rdf:Bag>
    <rdf:li>RIM JVM 4.1</rdf:li>
  </rdf:Bag>
</prf:JVMVersion>
<prf:Mexeclassmarks>
  <rdf:Bag>
    <rdf:li>1</rdf:li>
    <rdf:li>3</rdf:li>
  </rdf:Bag>
</prf:Mexeclassmarks>
<prf:MexecureDomains>No</prf:MexecureDomains>
<prf:OSName>RIM OS</prf:OSName>
<prf:OSVendor>RIM</prf:OSVendor>
<prf:OSVersion>2.0</prf:OSVersion>
<prf:RecipientAppAgent>RIM Browser</prf:RecipientAppAgent>
<prf:SoftwareNumber>4.1.0</prf:SoftwareNumber>
</rdf:Description>
</prf:component>
<!-- Browser UA Description -->
<prf:component>
  <rdf:Description rdf:ID="BrowserUA">
    <rdf:type rdf:resource="http://www.openmobilealliance.org/tech/profiles/UAPROF/
      ccppschemata-20021212#BrowserUA"/>
    <prf:BrowserName>RIM Browser</prf:BrowserName>
    <prf:BrowserVersion>4.1.0</prf:BrowserVersion>
    <prf:FramesCapable>No</prf:FramesCapable>
    <prf:HtmlVersion>4.0.1</prf:HtmlVersion>
    <prf:PreferenceForFrames>No</prf:PreferenceForFrames>
    <prf:TablesCapable>Yes</prf:TablesCapable>
    <prf:JavaAppletEnabled>no</prf:JavaAppletEnabled>
    <prf:JavaScriptEnabled>yes</prf:JavaScriptEnabled>
    <prf:JavaScriptVersion>1.3</prf:JavaScriptVersion>
    <prf:XhtmlVersion>1.0</prf:XhtmlVersion>
    <prf:XhtmlModules>
      <rdf:Bag>

```

```

        <rdf:li>XHTML1-struct</rdf:li>
        <rdf:li>xhtml-basic10</rdf:li>
    </rdf:Bag>
</prf:XhtmlModules>
</rdf:Description>
</prf:component>
<!-- Network Characteristics Description -->
<prf:component>
    <rdf:Description rdf:ID="NetworkCharacteristics">
        <rdf:type rdf:resource="http://www.openmobilealliance.org/tech/profiles/UAPROF/
            ccpschema-20021212#NetworkCharacteristics"/>
        <prf:CurrentBearerService>GPRS/EDGE</prf:CurrentBearerService>
        <prf:SecuritySupport>
            <rdf:Bag>
                <rdf:li>WTLS-1</rdf:li>
                <rdf:li>WTLS-2</rdf:li>
                <rdf:li>TLS</rdf:li>
                <rdf:li>SSL</rdf:li>
            </rdf:Bag>
        </prf:SecuritySupport>
        <prf:SupportedBearers>
            <rdf:Bag>
                <rdf:li>850MHz GPRS</rdf:li>
                <rdf:li>900MHz GPRS</rdf:li>
                <rdf:li>1800MHz GPRS</rdf:li>
                <rdf:li>1900MHz GPRS</rdf:li>
                <rdf:li>EGPRS</rdf:li>
                <rdf:li>EDGE</rdf:li>
            </rdf:Bag>
        </prf:SupportedBearers>
    </rdf:Description>
</prf:component>
<!-- WAP Characteristics Description -->
<prf:component>
    <rdf:Description rdf:ID="WapCharacteristics">
        <rdf:type rdf:resource="http://www.openmobilealliance.org/tech/profiles/UAPROF/
            ccpschema-20021212#WapCharacteristics"/>
        <prf:WapDeviceClass>C</prf:WapDeviceClass>
        <prf:WapVersion>2.0</prf:WapVersion>
        <prf:WmlDeckSize>32768</prf:WmlDeckSize>
        <prf:WmlScriptLibraries>
            <rdf:Bag>
                <rdf:li>Lang</rdf:li>
                <rdf:li>String</rdf:li>
                <rdf:li>URL</rdf:li>
                <rdf:li>WMLBrowser</rdf:li>
                <rdf:li>Dialogs</rdf:li>
            </rdf:Bag>
        </prf:WmlScriptLibraries>
        <prf:WmlScriptVersion>
            <rdf:Bag>
                <rdf:li>1.0</rdf:li>
            </rdf:Bag>
        </prf:WmlScriptVersion>
        <prf:WmlVersion>
            <rdf:Bag>
                <rdf:li>1.0</rdf:li>
                <rdf:li>1.1</rdf:li>
                <rdf:li>1.2</rdf:li>
                <rdf:li>1.3</rdf:li>
            </rdf:Bag>
        </prf:WmlVersion>
        <prf:WtaVersion>1.1</prf:WtaVersion>
        <prf:DrmClass>
            <rdf:Bag>

```

```

        <rdf:li>ForwardLock</rdf:li>
    </rdf:Bag>
</prf:DrmClass>
<prf:OmaDownload>No</prf:OmaDownload>
</rdf:Description>
</prf:component>
<!-- MMS Characteristics Description -->
<prf:component>
    <rdf:Description rdf:ID="MmsCharacteristics">
        <rdf:type rdf:resource="http://www.openmobilealliance.org/tech/profiles/MMS/
            ccppschemata-20050301-MMS1.2#MmsCharacteristics"/>
        <mms:MmsMaxMessageSize>307200</mms:MmsMaxMessageSize>
        <mms:MmsMaxImageResolution>640x480</mms:MmsMaxImageResolution>
        <mms:MmsVersion>
            <rdf:Bag>
                <rdf:li>1.2</rdf:li>
            </rdf:Bag>
        </mms:MmsVersion>
        <mms:MmsCcppAccept>
            <rdf:Bag>
                <rdf:li>text/plain</rdf:li>
                <rdf:li>text/html</rdf:li>
                <rdf:li>text/vnd.wap.wml</rdf:li>
                <rdf:li>text/vnd.wap.wmlscript</rdf:li>
                <rdf:li>application/vnd.wap.wmlc</rdf:li>
                <rdf:li>application/vnd.wap.wmlscriptc</rdf:li>
                <rdf:li>application/vnd.rim.html</rdf:li>
                <rdf:li>application/xhtml+xml</rdf:li>
                <rdf:li>application/vnd.wap.xhtml+xml</rdf:li>
                <rdf:li>application/vnd.wap.mms-message</rdf:li>
                <rdf:li>application/vnd.oma.drm.message</rdf:li>
                <rdf:li>application/smil</rdf:li>
                <rdf:li>image/vnd.wap.wbmp</rdf:li>
                <rdf:li>image/png</rdf:li>
                <rdf:li>image/gif</rdf:li>
                <rdf:li>image/jpeg</rdf:li>
                <rdf:li>image/jpeg</rdf:li>
                <rdf:li>image/vnd.rim.png</rdf:li>
                <rdf:li>audio/midi</rdf:li>
                <rdf:li>audio/sp-midi</rdf:li>
                <rdf:li>audio/x-mid</rdf:li>
                <rdf:li>audio/x-midi</rdf:li>
                <rdf:li>audio/mid</rdf:li>
                <rdf:li>audio/x-wav</rdf:li>
                <rdf:li>audio/wav</rdf:li>
                <rdf:li>audio/mpeg</rdf:li>
                <rdf:li>audio/x-mpeg</rdf:li>
                <rdf:li>audio/amr</rdf:li>
                <rdf:li>application/vnd.wap.multipart.mixed</rdf:li>
                <rdf:li>application/vnd.wap.multipart.alternative</rdf:li>
                <rdf:li>application/vnd.wap.multipart.related</rdf:li>
                <rdf:li>multipart/mixed</rdf:li>
                <rdf:li>multipart/alternative</rdf:li>
                <rdf:li>multipart/related</rdf:li>
                <rdf:li>text/vnd.sun.j2me.app-descriptor</rdf:li>
                <rdf:li>application/java-archive</rdf:li>
                <rdf:li>application/vnd.rim.cod</rdf:li>
                <rdf:li>application/vnd.wap.sic</rdf:li>
                <rdf:li>application/vnd.wap.slc</rdf:li>
                <rdf:li>application/vnd.wap.coc</rdf:li>
                <rdf:li>application/vnd.wap.sia</rdf:li>
                <rdf:li>application/x-vnd.rim.pme.b</rdf:li>
                <rdf:li>application/x-vnd.rim.pme</rdf:li>
                <rdf:li>image/pme</rdf:li>
                <rdf:li>text/x-vCard</rdf:li>
            </rdf:Bag>
        </mms:MmsCcppAccept>
    </rdf:Description>
</prf:component>

```

```
        <rdf:li>text/x-vCalendar</rdf:li>
    </rdf:Bag>
</mms:MmsCcppAccept>
<mms:MmsCcppAcceptCharSet>
    <rdf:Bag>
        <rdf:li>ISO-8859-1</rdf:li>
        <rdf:li>UTF-8</rdf:li>
        <rdf:li>UTF-16be</rdf:li>
        <rdf:li>Windows-1252</rdf:li>
        <rdf:li>US-ASCII</rdf:li>
    </rdf:Bag>
</mms:MmsCcppAcceptCharSet>
<mms:MmsContentClass>
    <rdf:Bag>
        <rdf:li>TX</rdf:li>
        <rdf:li>IB</rdf:li>
        <rdf:li>IR</rdf:li>
    </rdf:Bag>
</mms:MmsContentClass>
</rdf:Description>
</prf:component>
</rdf:Description>
</rdf:RDF>
```

# Using String Pattern Matching to Add Dynamic Menu Items

Mike Kirkup, Research In Motion

With BlackBerry® Device Software v4.0, several classes were exposed which provide the ability for third party applications to add dynamic menu items based on the matching of text in active text fields with string patterns.

Until now, there was no sample provided on how a third party application would be able to leverage this capability in their own application. Possible uses include:

- Providing phone number lookup by automatically recognizing phone numbers and looking them up in a reverse phone number lookup database.
- Providing additional integration between BlackBerry device applications and third party applications by enabling users to seamlessly transition between applications based on string patterns.

The following sample code highlights a use case of matching zip codes and looking up the zip code to show a list of cities covered by the zip code. For example, when this sample is

loaded, a user could type 90210 into an email message and the zip code would be underlined. The user selects the underlined zip code and clicks the trackwheel to bring up the default menu item of “Lookup ZipCode”. A zip code look up is performed and the results are displayed in the browser showing Beverly Hills, California as the city covered by that zip code.

There are five classes provided as part of the sample although many of the classes or functionality could be combined into one class. The classes were kept separate to make the sample as easy as possible to understand.

The key to the sample code is the registration of the StringPattern in the StringPatternRepository and the registration of the Factory implementation in the RuntimeStore. These two critical pieces allow the automatic handling of the string pattern matching and dynamic menu item to seamlessly show up for the user.

```
/*
 * PostalCodeLookupSample.java
 *
 * © Research In Motion Limited, 2006
 */

import net.rim.device.api.system.*;
import net.rim.device.api.util.*;

/**
 * This class represents the main class for the application. Note that this
 * application has been set up to run on startup for initialization purposes
 * and to run as a system module. It should not need to be run by the user
 * or show on the ribbon.
 */
public class ZipCodeLookupSample
{
    public static final long FACTORY_ID = 0xe840b1f1c2939e66L; // ZipCodeLookupSample.FACTORY_ID
    private static final long INSTANCE_ID = 0xa66adeb6a8301cbcL; // ZipCodeLookupSample.INSTANCE_ID

    // Boolean indicating whether initialization has occurred and to protect
    // against multiple registration.
    private boolean _initialized;

    /**
     * Entry point for the application.
     */
    public static void main( String[] args )
    {
        ZipCodeLookupSample app = ZipCodeLookupSample.getInstance();
        app.initialize();
    }
}
```

```

// Mark as private so it cannot be instantiated outside this class.
private ZipCodeLookupSample() {}

/**
 * Returns the one instance of the ZipCodeLookupSample on the system.
 */
public static ZipCodeLookupSample getInstance()
{
    RuntimeStore rs = RuntimeStore.getRuntimeStore();
    ZipCodeLookupSample sample = (ZipCodeLookupSample)rs.get( INSTANCE_ID );
    if( sample == null ) {
        sample = new ZipCodeLookupSample();
        rs.put( INSTANCE_ID, sample );
    }
    return sample;
}

/**
 * The initialize method will register the application
 * with all of the appropriate factories. It is important
 * to note that this method can be invoked multiple times
 * but it will only initialize the structures once.
 */
public void initialize()
{
    if( !_initialized ) {

        ZipCodeLookupFactory factory = new ZipCodeLookupFactory();
        RuntimeStore.getRuntimeStore().put( FACTORY_ID, factory );

        ZipCodeLookupStringPattern pattern = new ZipCodeLookupStringPattern();
        StringPatternRepository.addPattern( pattern );

        _initialized = true;
    }
}
}
/*
 * ZipCodeLookupFactory.java
 *
 * © Research In Motion Limited, 2006
 */

import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.util.*;

/**
 * The ZipCodeLookupFactory provides the necessary piece
 * which allows the StringPattern.Match class to indicate
 * how it should move forward when a match is found.
 * One will note that the ID for the Factory is the same
 * ID that is passed back from the findMatch method in the
 * StringPattern.
 */
public class ZipCodeLookupFactory implements Factory
{
    public ZipCodeLookupFactory()
    {
        // Do nothing for now
    }
}

```

```

/**
 * Create a new instance of the ActiveFieldCookie
 * using the information passed into this method.
 * @param initialData An ActiveFieldContext that contains
 * the necessary information from the StringPattern class.
 * @return An ActiveFieldCookie embodying the matched pattern.
 */
public Object createInstance( Object initialData )
{
    if ( initialData instanceof ActiveFieldContext ) {
        ActiveFieldContext afc = (ActiveFieldContext)initialData;
        String stringData = (String)afc.getData();
        return new ZipCodeLookupActiveFieldCookie( stringData );
    }
    return null;
}
}
/**
 * ZipCodeLookupMenuItem.java
 *
 * © Research In Motion Limited, 2006
 */

import java.io.*;
import javax.microedition.io.*;

import net.rim.blackberry.api.browser.*;
import net.rim.device.api.io.*;
import net.rim.device.api.system.*;
import net.rim.device.api.ui.*;
import net.rim.device.api.util.*;

/**
 * The ZipCodeLookupMenuItem class provides the MenuItem implementation
 * that is displayed in the appropriate application when the pattern has
 * been found. In this particular implementation, the MenuItem will
 * send the ZipCode to a website asking for the appropriate cities in that
 * ZipCode. The results are displayed in the browser.
 */
public class ZipCodeLookupMenuItem extends MenuItem
{
    private static final long GUID = 0x663468deaaa15552L; // ZipCodeLookupMenuItem.GUID
    private String URL = "http://www.getzips.com/CGI-BIN/ziplook.exe?What=1&Zip=";
    private String DEVICE_SIDE = ";deviceside=true";
    private String _zipCode;

    /**
     * Create the ZipCodeLookupMenuItem
     * @param zipCode the zip code to lookup
     */
    public ZipCodeLookupMenuItem( String zipCode )
    {
        super( "Lookup ZipCode", 5, 5 );
        _zipCode = zipCode;
    }

    /**
     * In the run method, simply start the HTTP thread that will
     * lookup the zip code.
     */
    public void run()
    {
        ZipCodeLookupThread thread = new ZipCodeLookupThread();
        thread.start();
    }
}

```

```

/**
 * The ZipCodeLookupThread will open up the HTTP Connection and communicate
 * with the server to obtain the information from looking up the zipcode.
 * After capturing all of the information, it will be passed into the browser
 * in such a way that doesn't require the browser to connect to the network.
 */
private class ZipCodeLookupThread extends Thread
{
    public void run()
    {

        DataBuffer dbuffer = new DataBuffer();

        try {
            // Use a StringBuffer to piece together the URL.
            StringBuffer sbuffer = new StringBuffer();
            sbuffer.append( URL ).append( _zipCode ).append( DEVICE_SIDE );

            HttpConnection connection = (HttpConnection)Connector.open( sbuffer.toString() );
            InputStream input = connection.openInputStream();

            // Read in the data from the InputStream
            byte[] temp = new byte[ 1024 ];

            for( ;; ) {
                int bytesRead = input.read( temp );
                if( bytesRead == -1 ) {
                    break;
                }
                dbuffer.write( temp, 0, bytesRead );
            }

            input.close();
            connection.close();
        } catch( IOException e ) {
            EventLogger.logEvent( GUID, e.toString().getBytes() );
        }

        try {
            // Create the ByteArrayOutputStream that will be used to capture
            // the Base64 encoded data.
            ByteArrayOutputStream output = new ByteArrayOutputStream();
            Base64OutputStream boutput = new Base64OutputStream( output );

            output.write( "data:text/html;base64,".getBytes() );
            boutput.write( dbuffer.toArray() );
            boutput.flush();
            boutput.close();

            output.flush();
            output.close();

            // Launch the browser using the base64 encoded data above.
            BrowserSession bSession = Browser.getDefaultSession();
            String data = output.toString();
            bSession.displayPage( data );
        } catch( IOException e ) {
            EventLogger.logEvent( GUID, e.toString().getBytes() );
        }
    }
}

```

```

/*
 * ZipCodeLookupStringPattern.java
 *
 * © Research In Motion Limited, 2006
 */

import net.rim.device.api.util.*;

/**
 * This class provides an implementation of the StringPattern.
 * It is important to note that every single time a character
 * is entered into an active field that this method will be executed.
 * As such, it is incredibly important to stress that the findMatch method
 * must be meticulously written for speed and efficiency so as to not
 * alter the user experience.
 */
public class ZipCodeLookupStringPattern extends StringPattern
{
    public ZipCodeLookupStringPattern() {}

    /**
     * For the purposes of this implementation, findMatch will match any five digit continuous
     * number (no spaces, etc) that it finds.
     * It is better to err on the side of caution and accept a zip code than exclude it.
     * @param str the AbstractString to search through.
     * @param beginIndex the beginning index in the string.
     * @param maxIndex the ending index in the string.
     * @param match the holder for information on the match if applicable
     * @return true if a match was found and false otherwise
     */
    public boolean findMatch
    ( AbstractString str, int beginIndex, int maxIndex, StringPattern.Match match )
    {
        // Because we are matching against a zip code which is a minimum of 5 numbers, we can
        // perform a quick check here to ensure that it is worth evaluating the actual characters.
        if( maxIndex - beginIndex < 5 ) {
            return false;
        }

        int numCounter = 0;

        for( int i = beginIndex; i < maxIndex; i++ ) {
            if( CharacterUtilities.isDigit( str.charAt( i ) ) ) {
                numCounter++;
                if( numCounter == 5 ) {
                    match.beginIndex = i - 4; // Begin Index is zero based
                    // End Index is expecting the end index after the last matching character
                    match.endIndex = i + 1;
                    match.id = ZipCodeLookupSample.FACTORY_ID;
                    match.prefixLength = 0;
                    return true;
                }
            } else {
                numCounter = 0;
            }
        }

        return false;
    }
}

```

```

/*
 * ZipCodeLookupActiveFieldCookie.java
 *
 * © Research In Motion Limited, 2006
 */

import java.util.*;

import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;

/**
 * The ActiveFieldCookie implementation allows for the factory to
 * provide a container for the zipcode information when moving
 * between the StringPattern and the MenuItem.
 */
public class ZipCodeLookupActiveFieldCookie implements ActiveFieldCookie
{
    private String _zipCode; // The ZipCode that matched the pattern.

    /**
     * Create the ZipCodeLookupActiveFieldCookie by passing in the zip code
     * @param data the zip code to lookup
     */
    public ZipCodeLookupActiveFieldCookie( String data )
    {
        // Save the zipcode for later use.
        _zipCode = data;
    }

    /**
     * This is an abstract method for the ActiveFieldCookie and must be implemented.
     * @return always return false.
     */
    public boolean invokeApplicationKeyVerb()
    {
        return false;
    }

    /**
     * In this case, we are under-utilizing the getFocusVerbs method by simply adding
     * our menu item to the list of items and returning. Refer to the javadocs for
     * all of the different options here.
     */
    public MenuItem getFocusVerbs(CookieProvider provider, Object context, Vector items)
    {
        items.addElement( new ZipCodeLookupMenuItem( _zipCode ) );
        return (MenuItem)items.elementAt(0);
    }
}

```

#### NOTE

This document is provided for informational and non-commercial or personal use only and must not be copied, disclosed, or posted on any network computer or broadcast in any media or otherwise distributed, in whole or in part. Any such copying, distribution, disclosure, posting, or broadcast is a violation of copyright laws. This document must not be modified. Use for any other purpose is expressly prohibited by law, and may result in severe civil and criminal penalties. Violators will be prosecuted to the maximum extent possible.

No Research In Motion Limited or BlackBerry logo, graphic, sound or image may be copied or retransmitted unless expressly permitted in writing by Research In Motion Limited.

The opinions expressed herein are strictly those of the writer(s) and do not necessarily reflect the opinion of Research In Motion Limited

© 2007 Research In Motion Limited. All Rights Reserved. The BlackBerry and RIM families of related marks, images and symbols are the exclusive properties of Research In Motion Limited. RIM, Research In Motion, 'Always On, Always Connected', BlackBerry are registered with the U.S. Patent and Trademark Office and may be pending or registered in other countries.

Sun, Java, J2ME, Java Card, JVM and JavaScript are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. The Bluetooth word mark is owned by Bluetooth SIG, Inc. and any use of such marks by Research In Motion Limited is under license. Adobe and Photoshop are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries. Windows is either a registered trademark or trademark of Microsoft Corporation in the United States and/or other countries. GSM is a registered and owned by the GSM Association. CDMA2000 is a registered trademark and certification mark of the Telecommunications Industry Association (TIA-USA). All other brands, product names, company names, trademarks and service marks are the properties of their respective owners.

The BlackBerry handheld and/or associated software are protected by copyright, international treaties and various patents, including one or more of the following U.S. patents: 6,278,442; 6,271,605; 6,219,694; 6,075,470; 6,073,318; D,445,428; D,433,460; D,416,256. Other patents are registered or pending in various countries around the world. Please visit [www.rim.net/patents.shtml](http://www.rim.net/patents.shtml) for a current listing of applicable patents.

This document is provided "as is" and Research In Motion Limited (RIM) assumes no responsibility for any typographical, technical or other inaccuracies in this document. RIM reserves the right to periodically change information that is contained in this document; however, RIM makes no commitment to provide any such changes, updates, enhancements or other additions to this document to you in a timely manner or at all. RIM MAKES NO REPRESENTATIONS, WARRANTIES, CONDITIONS OR COVENANTS, EITHER EXPRESS OR IMPLIED (INCLUDING WITHOUT LIMITATION, ANY EXPRESS OR IMPLIED WARRANTIES OR CONDITIONS OF FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, MERCHANTABILITY, DURABILITY, TITLE, OR RELATED TO THE PERFORMANCE OR NON-PERFORMANCE OF ANY SOFTWARE REFERENCED HEREIN OR PERFORMANCE OF ANY SERVICES REFERENCED HEREIN). IN CONNECTION WITH YOUR USE OF THIS DOCUMENTATION, NEITHER RIM NOR ITS AFFILIATED COMPANIES AND THEIR RESPECTIVE DIRECTORS, OFFICERS, EMPLOYEES OR CONSULTANTS SHALL BE LIABLE TO YOU FOR ANY DAMAGES WHATSOEVER BE THEY DIRECT, ECONOMIC, COMMERCIAL, SPECIAL, CONSEQUENTIAL, INCIDENTAL, EXEMPLARY OR INDIRECT DAMAGES, EVEN IF RIM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, INCLUDING WITHOUT LIMITATION, LOSS OF BUSINESS REVENUE OR EARNINGS, LOST DATA, DAMAGES CAUSED BY DELAYS, LOST PROFITS, OR A FAILURE TO REALIZE EXPECTED SAVINGS.

This document might contain references to third party sources of information, software and/or third party web sites ("Third-Party Information"). RIM does not control, and is not responsible for, any Third-Party Information, including, without limitation the content, accuracy, copyright compliance, legality, decency, links, or any other aspect of Third-Party Information. The inclusion of Third-Party Information in this document does not imply endorsement by RIM of the third party in any way. Any dealings with third parties, including, without limitation, compliance with applicable licenses and terms and conditions, are solely between you and the third party. RIM shall not be responsible or liable for any part of such dealings.

Prior to subscribing to or implementing any third party products or services, it is your responsibility to ensure that the airtime service provider you are working with has agreed to support all of the features of the third party products and services. Installation and use of third party products and services with RIM's products and services may require one or more patent, trademark or copyright licenses in order to avoid infringement of the intellectual property rights of others. You are solely responsible for determining whether such third party licenses are required and are responsible for acquiring any such licenses. To the extent that such intellectual property licenses may be required, RIM expressly recommends that you do not install or use these products and services until all such applicable licenses have been acquired by you or on your behalf. Your use of third party software shall be governed by and subject to you agreeing to the terms of separate software licenses, if any, for those products or services. Any third party products or services that are provided with RIM's products and services are provided "as is". RIM makes no representation, warranty or guarantee whatsoever in relation to the third party products and services and RIM assumes no liability whatsoever in relation to the third party products and services even if RIM has been advised of the possibility of such damages or can anticipate such damages.

RIM does not claim ownership of the materials you provide to RIM in any way. However, by posting, uploading, inputting, providing or submitting any such materials, you are granting RIM and any necessary sublicensees permission to use your submitted materials in connection with the operation of its business, including, without limitation, the rights to: copy, distribute, transmit, publicly display, publicly perform, reproduce, edit, translate and reformat your submitted material; and to publish your name and any contact information you provide in connection with your submitted material. No compensation will be paid with respect to the use by RIM or any necessary licensee of your submitted material, as provided herein. RIM is under no obligation to post or use any material you may provide and RIM may remove any such material at any time in its sole discretion. By posting, uploading, inputting, providing or submitting your material you warrant and represent that you own or otherwise control all of the rights to your material as described in this section including, without limitation, all the rights necessary for you to provide, post, upload, input or submit the material. Notwithstanding the foregoing, any suggestions, improvements or modifications to RIM products and services ("Enhancements") made by you or anyone acting on your behalf, including your employees, will be the property of RIM without any further consideration to you, whether or not such Enhancements are incorporated into RIM products and services.

By posting, uploading, inputting, providing or submitting any information to RIM, you consent to RIM's collection of such information, and you grant RIM, its affiliated companies and necessary sublicensees permission to use such submitted information in connection with the operation of this journal and its business, including, without limitation, the worldwide, royalty-free rights to: copy, distribute, transmit, publicly display, publicly perform, reproduce, edit, translate and reformat your submitted information; and to publish your name and any contact information you provide of in connection with your submitted information.

If you submit personal information to RIM, you consent to the collection, use, and disclosure of such information by RIM in accordance with RIM's privacy policy which can be found at <http://www.blackberry.com/legal/privacy.shtml>.