JavaOne℠

MIDP 3.0 in-depth: Tutorials and demonstrations

Stanly Kao
Lakshmi Dontamsetti
Aplix Corporation

Roger Riggs
Sun Microsystems

New LCDUI components and functionality

- TabbedPane
- FileSelector
- Notification
- Menu
- Mutable Commands
- CommandLayoutPolicy
- FormLayoutPolicy
- TableLayoutPolicy
- Idle Screen MIDlets

- Application Defined Fonts
- Multiple Display
- DisplayListener
- KeyListener
- Form's Traversal Listener
- ScalableImage
- Text
- TextEditor
- Standardize Displayable's *getWidth()* and *getHeight()* definition.

TabbedPane

- **TabbedPane** presents a series of *lcdui.Screen* to the users and allows them to navigate between screens.
- TabbedPane has two modes:
  - *String mode*: Strings are used to represent each tab's contents.
  - *Image mode*: Icons represent each tab's contents.

FileSelector

- **FileSelector** allows the user to select or create a file or directory from the device's file system.
- The directory and file names passed to and returned by the FileSelector are formatted as fully qualified absolute file URLs [RFC 3986].

Notification

- **Notification** allows an application to post a small informational note to the user without necessarily taking control of the screen.
- The Notifications are managed by the implementation.
- MIDlets can post and remove Notifications.

Menu

- **Menu** is a container of Commands and Menus.
- Menu supports labels, Font, Images and enable/disable states.
- The Image, label, Font and state of the Menu can be changed at any time.

Mutable Commands

- **Commands** supports label, Font, Images and enable/disable states.
- The Image, label, Font and state of the Command can be changed at any time.

New LCDUI components and functionality

- TabbedPane
- FileSelector
- Notification
- Menu
- Mutable Commands
- CommandLayoutPolicy
- FormLayoutPolicy
- TableLayoutPolicy
- Idle Screen MIDlets
- Application Defined Fonts

- Multiple Display
- DisplayListener
- KeyListener
- Form's Traversal Listener
- ScalableImage
- Text
- TextEditor
- Standardize Displayable's *getWidth()* and *getHeight()* definition.

Layout Policies

CommandLayoutPolicy gives control on placing Commands/Menus at the exact placement (softkey).

FormLayoutPolicy gives control on the layout of Items within a Form.

CommandLayoutPolicy

- CommandLayoutPolicy instance can be set to either on a Display or on a Displayable.
- Layout of Command/Menus are done via *onCommandLayout()* method.
- Any Command or Menu not explicitly set at a placement in the CommandLayoutPolicy implementation will be ignored and not displayed.

Abstract FormLayoutPolicy

- The layout algorithm is implemented by the *doLayout* method.
- Position and size of each Item are initially zero. It's the FormLayoutPolicy's responsibility to set valid positions and sizes of each Item.
- Items set beyond the viewport will be clipped.
- If *doLayout* and *getTraverse()* method throws RuntimeException, layout will be revert to default layout policy.

FormLayoutPolicy

Defining FormLayoutPolicy

```
class CustomLayoutPolicy extends FormLayoutPolicy {


 public void doLayout(int viewportX, int viewportY, int viewportWidth, int viewportHeight,
                                int[] totalSize) {
        int x, y, itemWidth, itemHeight, size;
        int totalWidth, totalHeight;
        size = form.size();
        for( int i = 0; i < size; i++ ) {
            Item item = form.get(i);
             // code to calculate item's co-ordinates and size
             setPosition( item, x, y );
             setSize( item, itemWidth, itemHeight );
           }
            // code to calculate the width and height required by all Items
           // and set totalSize array
           totalSize[0] = totalWidth;
           totalSize[1] = totalHeight;
}
```

FormLayoutPolicy

Define Form class

```
Class CustomForm extends Form {
    // append Items into the Form
    append(item1);
    append(item2);
     …….
}
```

Setting FormLayoutPolicy

```
Form customForm = new CustomForm();
FormLayoutPolicy layoutPolicy = new CustomLayoutPolicy(customForm);
customForm.setLayoutPolicy( layoutPolicy );
```

Number of Items in MIDP 3.0 and MIDP 2.x

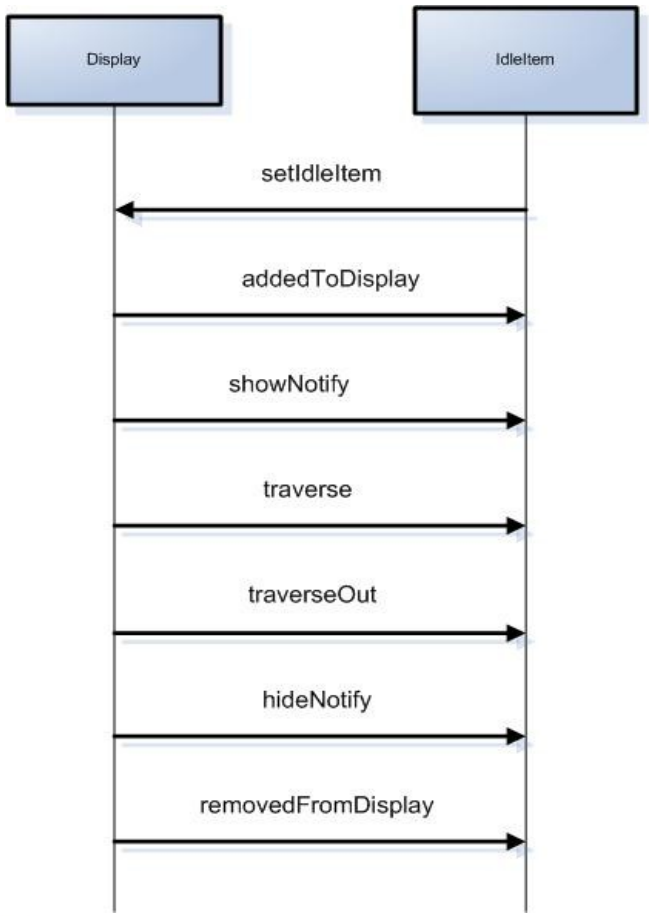
```
void appendItems() {
    append(HOTEL_NAME );
    append(THICKER_LINE );
    append(HOTEL_IMAGE );

    append(ADDRESS );

    append(RATING );
    append(REVIEWS);

    append(RATING_LINK );
    append(REVIEWS_LINK );

    append(HOTEL_INFO );

    append(HOTEL_DETAILS );
    append(BAR);
    append(ROOM_OPTIONS );

    for( int i = 0; i < NUMBER_ICONS ; i++ ) {
        append( ICONS[i] );
    }

    append(ECO_FRIENDLY );
    append(BOOK_NOW );
    append(PRICE );
}
```

```
> void appendItems() {
>     append(SPACER_BTW_ROWS[0]);
>     append(HOTEL_NAME );
>     append(THICKER_LINE_1 );
>     append(SPACER_BTW_ROWS[1]);
>     append(SPACER_LEFT[0]);
>     append(HOTEL_IMAGE );
>
>     append(SPACER_LEFT[1]);
>     append(ADDRESS );
>
>     append(SPACER_BTW_ROWS[3]);
>     append(SPACER_LEFT[2]);
>     append(RATING );
>     append(SPACER_BTW_RR_1);
>     append(REVIEWS);
>
>     append(SPACER_LEFT[3]);
>     append(RATING_LINK );
>     append(SPACER_BTW_RR_2);
>     append(REVIEWS_LINK );
>
>     append(SPACER_BTW_ROWS[2]);
>     append(SPACER_LEFT[4]);
>     append(HOTEL_INFO );
>
>     append(SPACER_BTW_ROWS[4]);
>     append(SPACER_LEFT[5]);
>     append(HOTEL_DETAILS );
>     append(BAR);
>     append(ROOM_OPTIONS );
>
>     append(SPACER_BTW_ROWS[5]);
>     append(SPACER_LEFT[6]);
>     for( int i = 0; i < NUMBER_ICONS; i++ ) {
>         append( ICONS[i] );
>         append( SPACER_BTW_ICONS[i]);
>     }
>
>     append(SPACER_BTW_ROWS[6]);
>     append(SPACER_LEFT[7]);
>     append(ECO_FRIENDLY );
>
>     append(SPACER_BTW_ROWS[7]);
>     append(SPACER_LEFT[8]);
>     append(BOOK_NOW );
>     append(SPACER_LEFT[9]);
```

IdleItem MIDlets

- Idle Screen MIDlet is a normal MIDlet that has an additional UI component (*IdleItem*), which is rendered on the idle screen (home screen).
- As a subclass of CustomItem, it has just as much control over rendering as it has within a Canvas.
- As a subclass of Item it is possible for a MIDlet to add Commands to an IdleItem.

IdleItem MIDlets



- IdleItems are set or cleared from the Display via the *setIdleItem()* method.
- *IdleItem.addedToDisplay*() and *IdleItem.removedFromDisplay*() are called when the IdleItem is added to or removed from the Display.

IdleItem MIDlets

- The platform decides how Idle Screen MIDlets are positioned on the idle screen.
- The IdleItem provides APIs that can be used to match the platform's native look and feel:
  - Drawing focus and highlight
  - Background images/wallpaper
  - Fonts and Colors used on the idle screen
- Idle Screen MIDlets are identified with JAD attribute:

  MIDlet-<n>-Category: idlescreen

IdleItem MIDlets

IdleItem definition:

```
public class TwitterItem extends IdleItem  {
  public void paint( Graphics g, int w, int h ) {
        int y = 0;
        int numFeeds = m_feedVector.size();

        Image image1 = (Image) m_feedVector.elementAt( m_index % numFeeds );
        Image image2 = (Image) m_feedVector.elementAt( Math.abs(m_index - 1) % numFeeds );
        Image image3 = (Image) m_feedVector.elementAt( Math.abs(m_index - 2) % numFeeds );

        g.drawImage(m_titleBarImg, 0, 0, 0);
        g.drawImage(image1, 0, y += m_titleBarImg.getHeight() - 1, 0);
        g.drawImage(image2, 0, y += image1.getHeight() - 1, 0);
        g.drawImage(image3, 0, y += image2.getHeight() - 1, 0);
  }
}
```

Setting IdleItem to the Display

```
public class Twitter extends MIDlet {
    public Twitter() {
        Display.getDisplay(this).setIdleItem( new TwitterItem() );
    }
}
```

LIBlet Basics

- LIBlets are Java classes and resource files packaged in a JAR file that can be shared by multiple MIDlet suites.
- MIDP 3.0 provides a new provisioning mechanism to distribute shared Java code called LIBlets.
- LIBlets do not have their own execution environment. The Java classes can only be run inside a MIDlet suite's execution environment.
- A LIBlet can also depend on another LIBlet.

LIBlet Advantages

- For Developers:
  - Smaller MIDlet code size
  - Decreases development time by using shared Java code
  - Reduces fragmentation

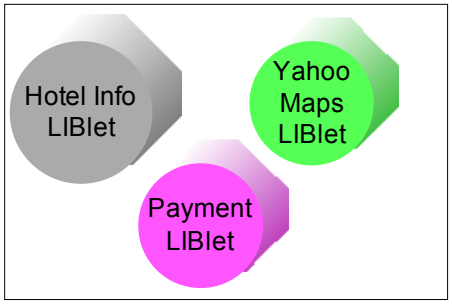
- For Service Providers:
  - Web Services can be made more accessible to J2ME developers
  - Proprietary APIs can be deployed easily

LIBlet Ecosystem

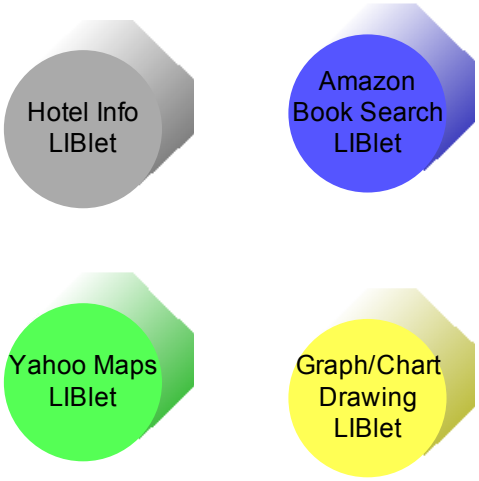
- LIBlets enable a new paradigm in MIDlet development similar to mash-ups in web development.
- Developers can now mix and match different LIBlets to easily create new applications quickly.
- As more providers encapsulate their services with LIBlets, MIDlet developers will have an expanding choice of functionality to choose from.

LIBlet Ecosystem Example

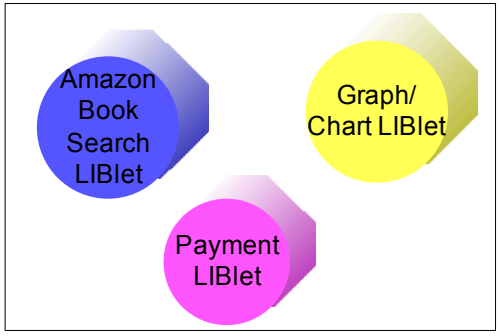
Hotel MIDlet



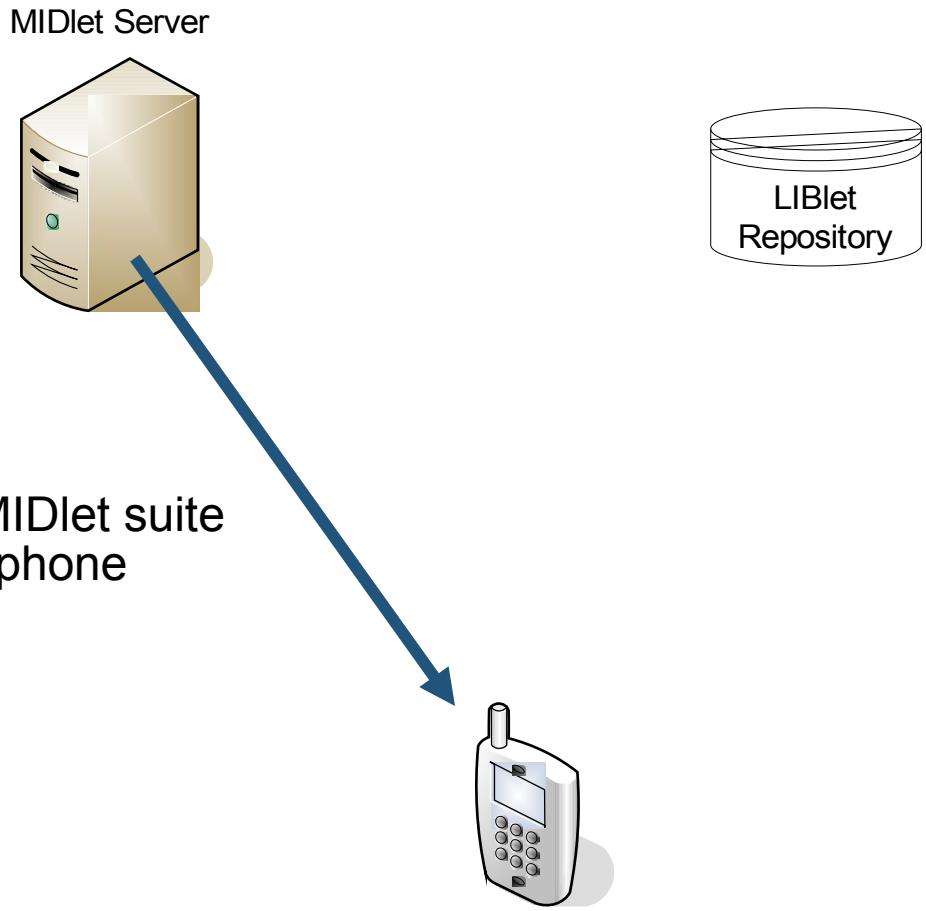
Available LIBlets



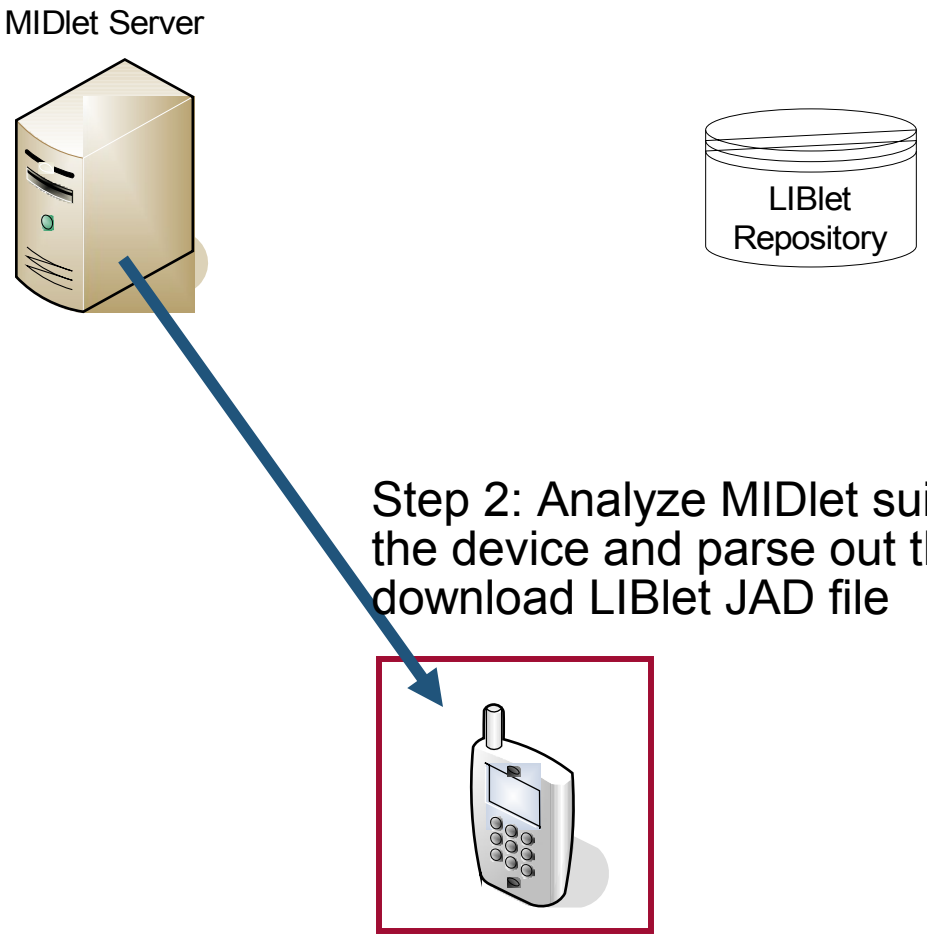
Book Search MIDlet



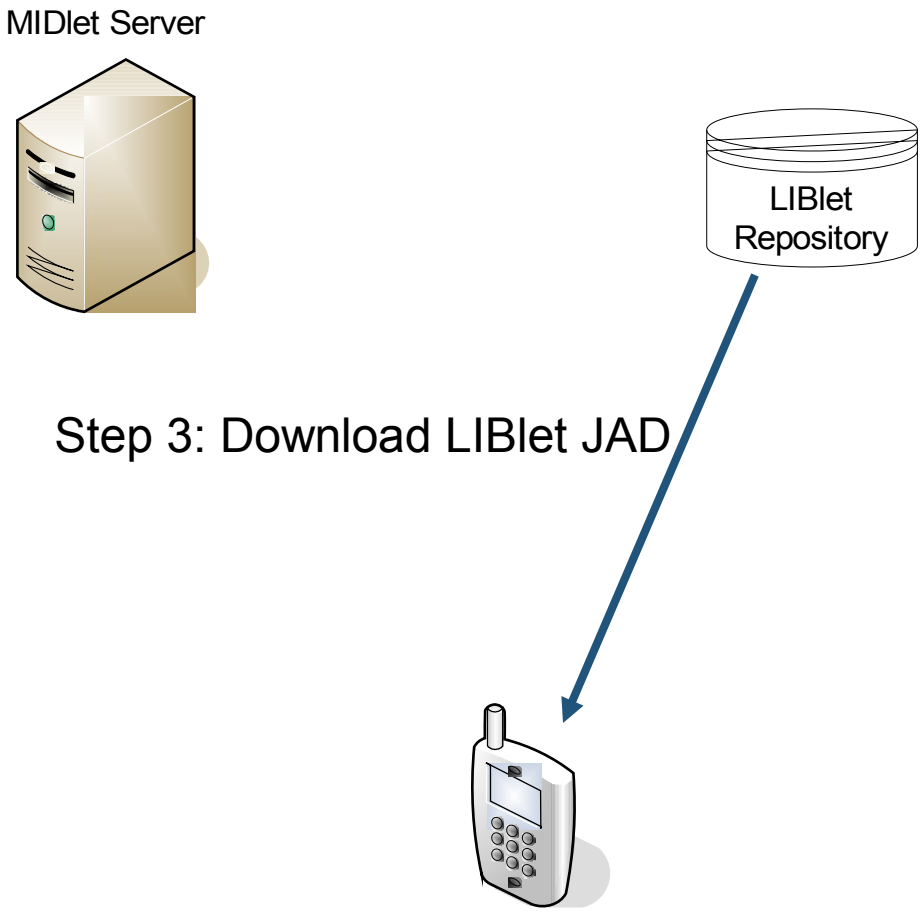
LIBlet Provisioning



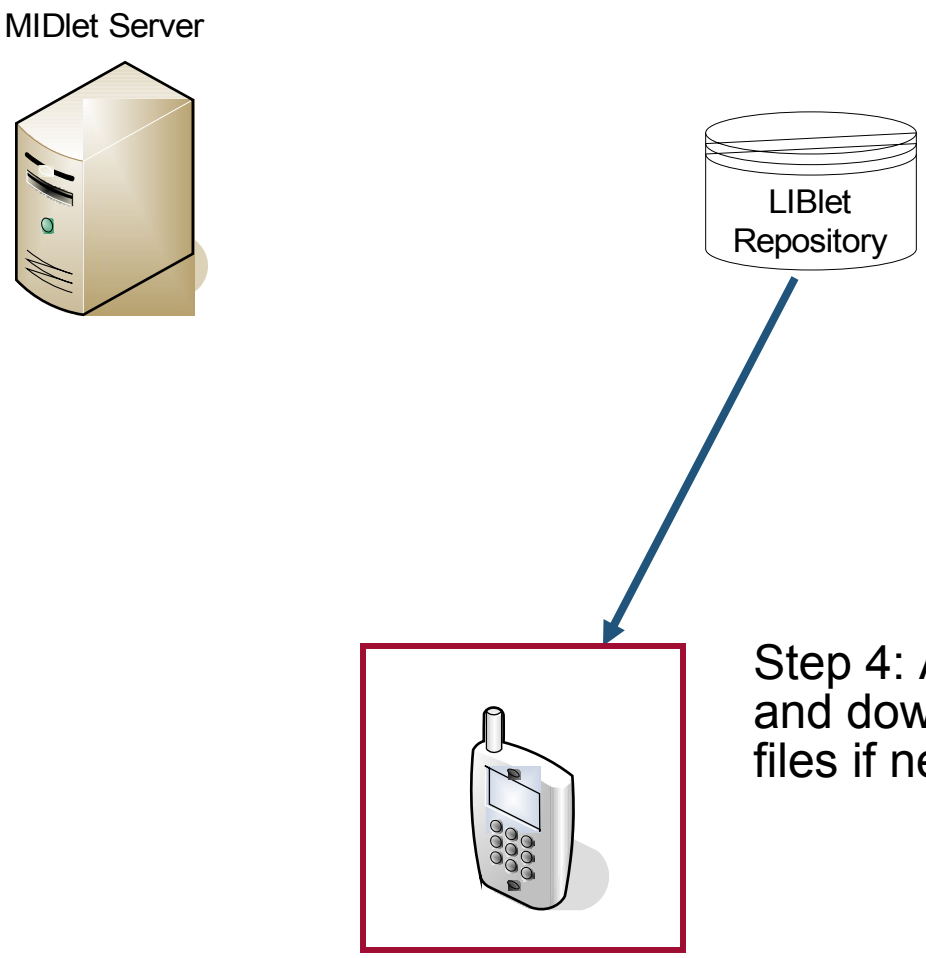
LIBlet Provisioning



LIBlet Provisioning

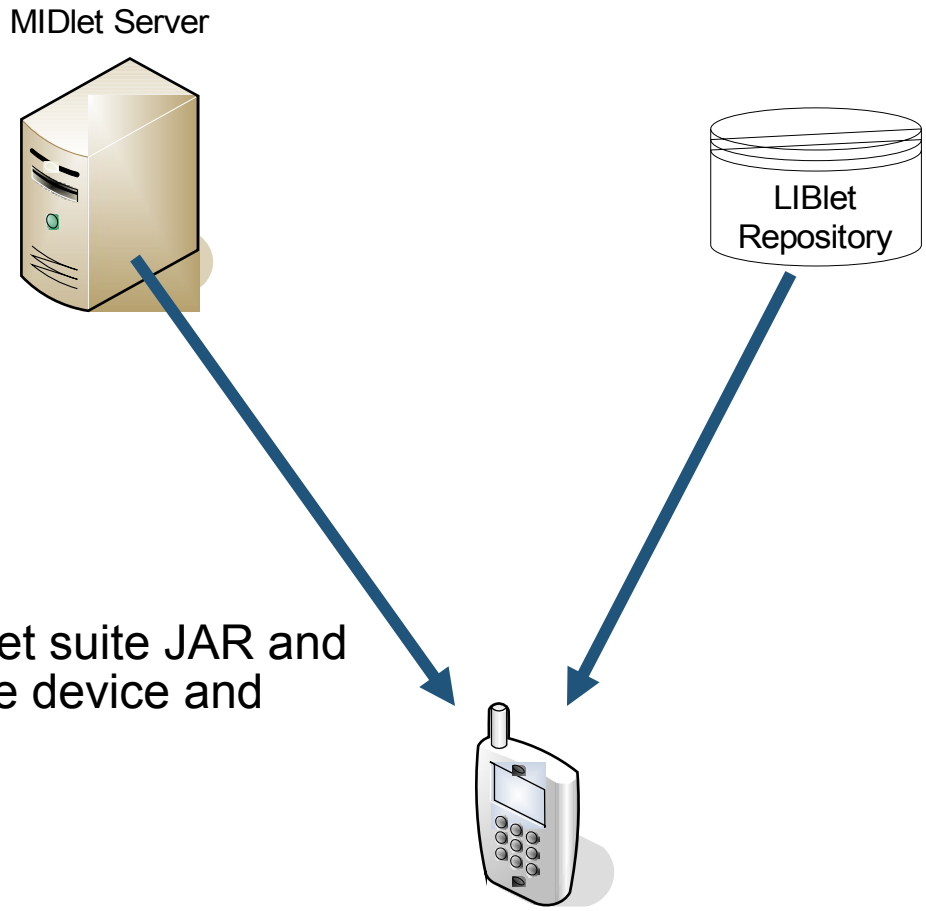


LIBlet Provisioning



Step 4: Analyze LIBlet JAD and download more LIBlet JAD files if necessary

LIBlet Provisioning



Step 5: Download MIDlet suite JAR and LIBlet JAR files onto the device and install the MIDlet suite.

LIBlet Dependency Attributes

- A LIBlet is uniquely identified by its
  - name
  - vendor
  - version
  - hash of the jar file
- The MIDlet-Dependency-<n> attribute specifies the name, vendor, and version.
- Example:
- MIDlet-Dependency-1: liblet; required; mapservice; Aplix; 1.0

LIBlet Attributes Contd.

- Each MIDlet-Dependency-<n> must have two corresponding attributes:
  - MIDlet-Dependency-JAD-URL-<n>
    - Tells the device where to download the LIBlet JAD file
  - MIDlet-Dependency-JAR-SHA1-<n>
    - Hash of the LIBlet JAR file is required. It ensures MIDlet suites will always have to specify an exact match to a specific LIBlet. This is done for security concerns.

LIBlets Demo

- Create a simple MIDlet that uses a Yahoo! MapService LIBlet.
- The JAD files for the demo are the following:

Demo MIDlet JAD

```
MIDlet-1: MIDP30Demo
MIDlet-Jar-Size: 249370
MIDlet-Jar-URL: MIDP30Demo.jar
MIDlet-Name: MIDP30Demo
MIDlet-Vendor: Aplix
MIDlet-Version: 1.0
MIDlet-Dependency-1: liblet; required; mapservice; Aplix; 1.0
MIDlet-Dependency-JAD-URL-1: mapservice.jad
MIDlet-Dependency-JAR-SHA1-1: 2309dglkjetoi33
MicroEdition-Configuration: CLDC-1.1
MicroEdition-Profile: MIDP-3.0
```

LIBlets Demo

MapService LIBlet JAD

LIBlet-Jar-URL: mapservice.jar
LIBlet-Name: mapservice
LIBlet-Vendor: Aplix
LIBlet-Version: 1.0
LIBlet-Jar-Size: 16692
LIBlet-Jar-SHA1: 2309dglkjetoi33

LIBlet Demo Code

- Run Demo here

LIBlet Demo Code

- MapService LIBlet

```
public interface MapServiceOwner {
    public void notifyMapImage(Image i) ;
    public void notifyMapError(String message) ;
}

public class MapServiceLiblet {
     public MapServiceLiblet(MapServiceOwner owner);
     public void setAddress(MapAddress address);
}
```

LIBlet Demo Code

- Demo MIDlet Source Code

```
1 import javax.microedition.lcdui.*;
2 import javax.microedition.lcdui.game.*;
3 import javax.microedition.midlet.MIDlet;
4 import com.aplix.demo.mapservice.* ;
5
6 public class JavaOneLIBletDemo extends MIDlet implements MapServiceOwner, CommandListener {
7
8     Form mainForm;
9     ImageItem mapImageItem;
10    TextField addressTextField;
11    MapServiceLiblet mapService;
12    Display mainDisplay;
13
14    private final static Command CMD_EXIT = new Command("Exit", Command.EXIT, 1);
15    private static final Command CMD_MAP = new Command("MAP IT!!", Command.OK, 1);
16
17    public JavaOneLIBletDemo() {
18          mainForm = new Form("JavaOne LIBlet Demo");
19          addressTextField = new TextField("Address", null, 1024, TextField.ANY);
20          mapService = new MapServiceLiblet(this);
21    }
22    protected void startApp() {
23       mainDisplay = Display.getDisplay(this);
24       mainForm.append(addressTextField);
25       mainForm.setCommandListener(this);
26       mainForm.addCommand(CMD_EXIT);
27       mainForm.addCommand(CMD_MAP);
28       mainForm.setCommandListener(this);
29       mainDisplay.setCurrent(mainForm);
30    }
```

LIBlet Demo Code

```
31 protected void destroyApp(boolean unconditional) {
32 }
33
34 protected void pauseApp() {
35 }
36 public void commandAction(Command c, Displayable d) {
37       if (c == CMD_EXIT) {
38           destroyApp(false);
39           notifyDestroyed();
40       }
41       if (c == CMD_MAP) {
42           MapAddress addr = new MapAddress(addressTextField.getString()," "," ");
43           mapService.setAddress(addr);
44           mainForm.deleteAll();
45           mainForm.append("Loading.....");
46           mainDisplay.setCurrent(mainForm);
47
48       }
49   }
50   public void notifyMapImage(Image i) {
51       int w = mainForm.getWidth() ;
52       int h = mainForm.getHeight() ;
53       int iw = i.getWidth() ;
54       int ih = i.getHeight() ;
55       int x = (iw-w)/2;
56       int y = (ih-h)/2;
57       Image centeredImage = Image.createImage(i, x, y,w, h, Sprite.TRANS_NONE);
58       mainForm.deleteAll();
59       mainForm.append(centeredImage);
60       mainDisplay.setCurrent(mainForm);
61   }
62
63   public void notifyMapError(String message) {
64       Alert a = new Alert("Error", message, null, AlertType.ERROR) ;
65       a.setTimeout(Alert.FOREVER) ;
66       mainDisplay.setCurrent(a) ;
67   }
68 }
```

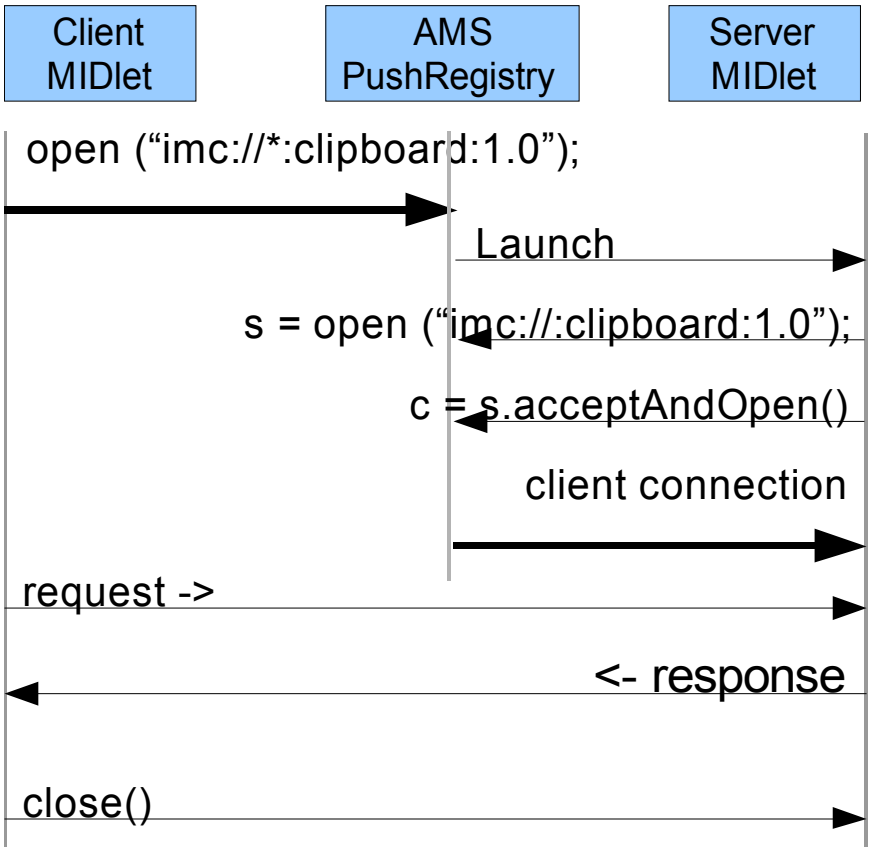
LIBlet Wrap-up

- MIDP3.0 standardizes a way for shared java code to be provisioned on to the device.
- MIDlets specifies dependency in the JAD file using the following three attributes:
  - MIDlet-Dependency-<n>
  - MIDlet-Dependency-JAD-URL-<n>
  - MIDlet-Dependency-JAR-SHA1-<n>
- LIBlets create a new development model that will shorten development time and MIDlet code size.

Communication Between MIDlets

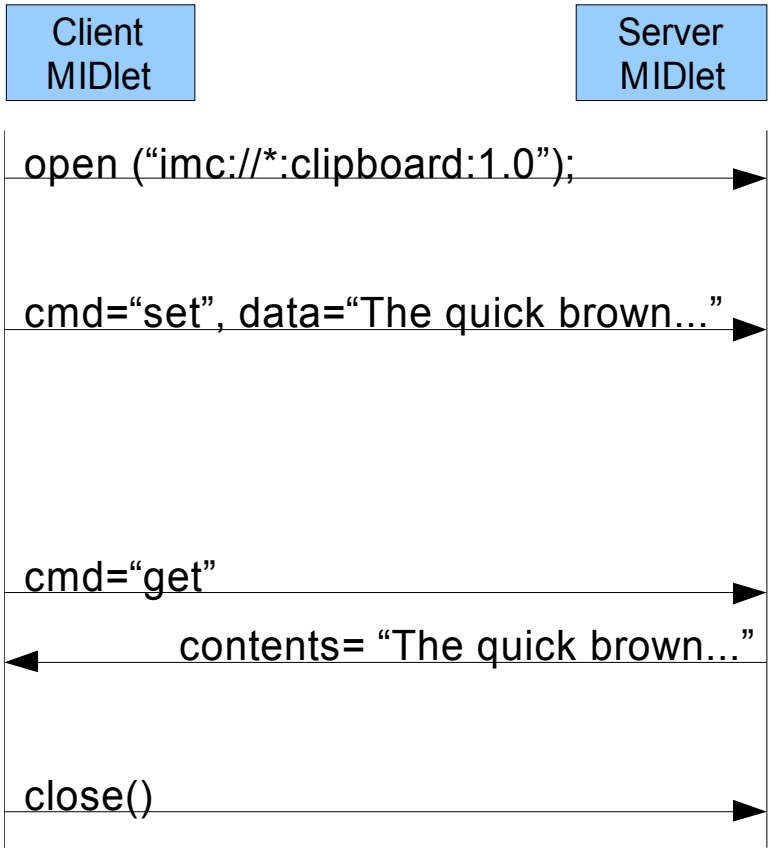
- Communicate between MIDlets on a single Device
- Inter-MIDlet Communication (IMC)
  - IMC Server
  - IMC Client
- Event Communication
  - System Events
  - Application Events
- Tutorial

IMC Server



- IMC Connection is named
- IMC Name is registered with PushRegistry
- Client Opens Named Connection
- If IMC Server is not open
  - MIDlet is launched
  - MIDlet opens server side
- MIDlet accepts IMCConnection from client
- Client to Server protocol is defined by Server

IMC Client



- Typical
  - Open named Connection
  - Writes request to stream
  - Reads result from stream
  - Close Connection
- Any protocol can be used
  - Keep connection open and just repeat for next request
  - Streaming data from Server to client
  - etc.

Tutorial
Client Access to IMC Clipboard Service

```
// Open the Connection and Input and Output Streams
IMCConnection conn = Connector.open("imc://*:clipboard:1.0");
DataInputStream clipInput = conn.openDataInputStream();
DataOutputStream clipOutput = conn.openDataOutputStream();

// Set the clipboard contents
clipOutput.writeUTF(“set”);
clipOutput.writeUTF(“The Quick Brown Fox”);
clipOutput.flush();

// Send the command to ask for the Clipboard contents
clipOutput.writeUTF(“get”);
clipOutput.flush();

// Read the value
String value = clipInput.readUTF();
```

Tutorial
Server Provides Clipboard Service

```
// Open the server connection, for example, "imc://:clipboard:1.0"
public void open(String conn) {
   IMCServerConnection server = Connector.open(conn);
   while (true) {
      IMCConnection conn = server.acceptAndOpen();
      DataInputStream clipInput = conn.openDataInputStream();
      DataOutputStream clipOutput = conn.openDataOutputStream();

      String command = clipInput.readUTF();
      if (“get”.equals(command)) {
         clipOutput.writeUTF(contents);
         clipOutput.flush();
      } else if (“set”.equals(command)) {
         contents = clipInput.readUTF();
      }
   }
}
```

Tutorial
Clipboard Server Packaging

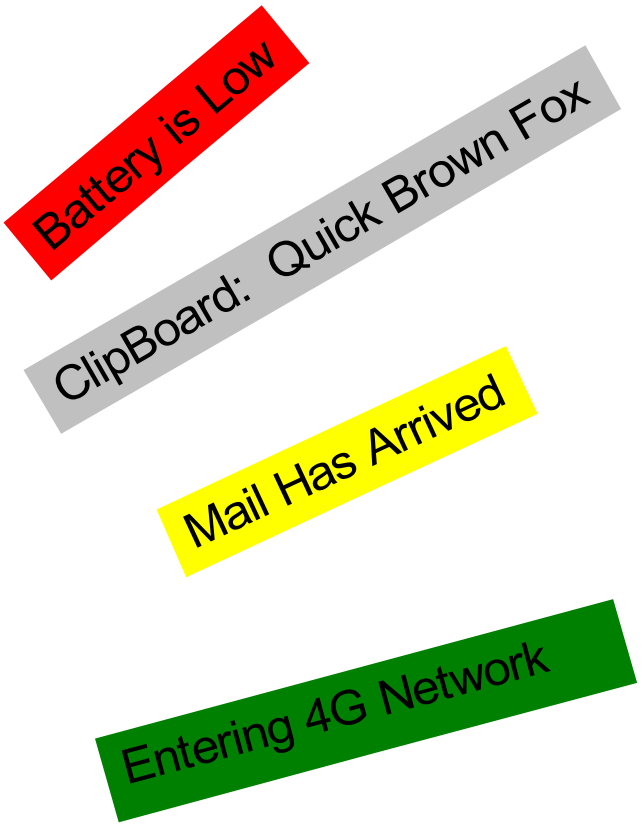
- Push Registration Attributes

```
MIDlet-1: Clipboard,, ClipMIDlet
MIDlet-Push-1: imc://:clipboard:1.0, ClipMIDlet
MIDlet-Name: Clipboard Server
MIDlet-Vendor: ClipsRUS
MIDlet-Version: 1.0
MIDlet-Description: IMC Server supporting a Clipboard
```

Tutorial
Clipboard Startup

```
IMCServer server;
String[] activeConn;
// On startup get the active connection and open it
public void startApp() {
   activeConn = PushRegistry.listConnections(true);
   if (activeConn.length() == 0) {
      notifyDestroyed(); // No active connection; exit
   } else {
      new Thread(this).start();   // start a thread to process
   }
}
// Thread to process requests
public void run() {
   // Open the server connection as in slide above.
   open(activeConn[0]);
}
```

Events
Publish and Subscribe

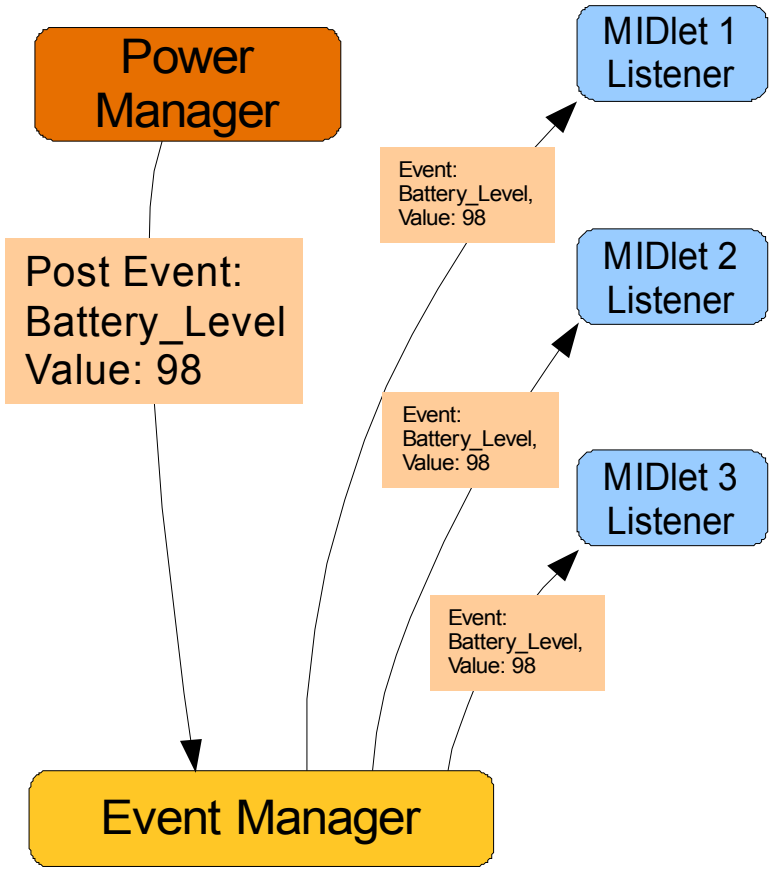


- Event Data
  - Name: string event name
  - Typed value: string, long, boolean, double
  - Message: string
  - Info: an extra typed value, depending on the event
- For example, BATTERY_LEVEL is integer 0..100

Event API

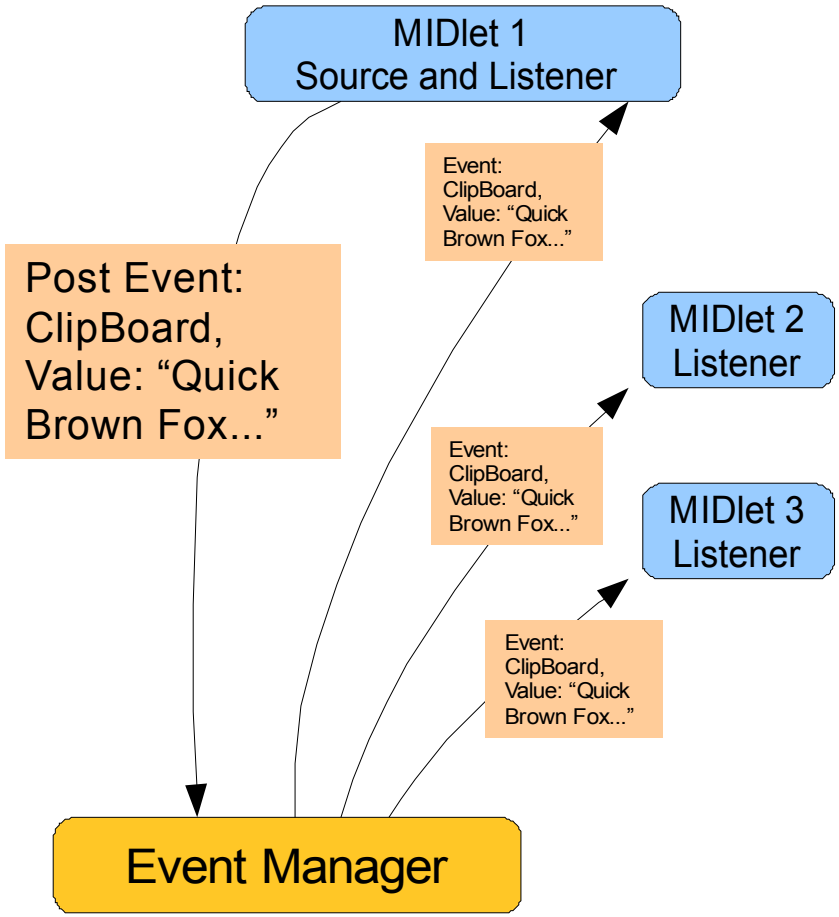
- Events are posted by System or Applications
- Listeners are notified when values are within range
  - Listeners are added for name, type, range/value
  - Every listener is notified on each matching event
  - For example, 0..20 to report low battery
- Register to launch MIDlet when values change
  - For example, when DATA_NETWORK changes
- EventPermission
  - Event functions subject to domain policy

System Events



- Reserved namespace for System events
- Battery level, battery low, charging, external power
- System State – normal, standby, startup, shutdown
- ScreenSaver – activated, deactivated
- Data Network – 3G, EDGE, HSPDA, WiFi, CDMA, etc.
- Voice Call – true, false
- Profile – general, meeting, outdoor, silent, etc.
- Extensible by System.

Application Events



- App event names follow reverse domain convention
- Type, message, and info is defined for each event
- Events are posted via the EventManager
- Posted events are delivered to every listener

MIDP 30 Advantages

- User Interface
  - Giving more control to app developer (layout policies, Font, ScalableImage etc)
  - New components with advanced features
- LIBlets
  - Decreases development time by using shared Java code
  - Enable mash-up in web development
- Events
  - MIDlets can communicate each other in multiple ways

Thank You

Lakshmi Dontamsetti
lakshmi@aplixcorp.com

Stanly Kao
stan@aplixcorp.com

Roger Riggs
roger.riggs@sun.com