JavaOne℠

Mobile Information Device Profile 3.0 (MIDP 3.0): Overview

Paul Su, Aplix Corporation of America
Angus Huang, Aplix Corporation of America
Roger Riggs, Sun Microsystems

MIDP 3.0 overview

- Next evolution of MIDP
- Additional functionality and improved components.
- Retains backward compatibility with MIDP 1.0/2.x applications.

MIDP 3.0 expert group

- JSR 271
- 72 members in the expert group. 23 participant companies, 38 observer companies and 11 individual contributors
- Motorola is the specification lead of MIDP 3.0
- Aplix will provide the RI and TCK
- Final release will be November 2009

Functional Categories

- Concurrency
- Application Management Software (AMS)
- Provisioning
- Security
- LCDUI

Concurrency

- Standardized concurrency behavior
- Events
- Inter-MIDlet Communication (IMC)
- Notification

Concurrency
Standardized behavior

- MIDP 1.0/2.x did not prohibit concurrency, but did not standardize behavior.
- Concurrency implementations exist in the market today, but behavior is fragmented.
- MIDP 3.0 defines expected concurrency behavior
- MIDP 3.0 makes concurrency mandatory
- MIDP 3.0 adds additional functionality that allows MIDlets to run together intelligently

Concurrency
Events

- Mechanism for notifying an application of changes in system state and for application to application communication.
- Two types of events:
  - System Events are predefined events that are sent by the system to registered applications.
  - App-to-App Events are custom events defined and sent by applications. Other applications can register to receive these events.
- Access to these events can be controlled via application-level access authorization.

Concurrency
Events

- Application may be launched automatically in response to events.
- Static registration through a JAD attribute
- Dynamic registration during execution:
  - An application can dynamically register to be launched by calling EventManager.registerApplication()

Concurrency
Inter-MIDlet Communication

- Provides asynchronous bi-directional stream connection between MIDlets.
- Very similar to socket protocol.
- Allows two types of connections:
  - Client connection (IMCConnection)
  - Server connection (IMCServerConnection)
- Access to these events can be controlled via application-level access authorization.

Concurrency
Notification

- A small unobtrusive informational note representing an event that the user would be interested in.
- When the Notification is inspected, the callback methods are called on the NotificationListener registered with the Notification.
- A Notification consists of a label and image.

Concurrency
Notification – Example

- Scenario: A game application is running in the foreground and the mail application is running in the background.
- The mail application posts a new Notification when a new e-mail has arrived.

Application Management Software (AMS)

- Auto Start MIDlets
- User Control of Applications
- Screensaver MIDlets
- Idle Screen MIDlets

AMS
Auto Start MIDlets

- Developers can have their MIDlets automatically launched on device power-up.

- `MIDlet-<n>-Type: autostart`

- Only MIDlets that have the AutoStartPermission are allowed to auto start.

- The AMS MUST attempt to restart Auto Start MIDlets on termination of the MIDlet.

AMS

User Control of Applications

- Developer can determine how much control the user has over the MIDlet's lifecycle.
- The developer can limit the user from:
  - Updating/Deleting a MIDlet Suite
  - Starting/Stopping a MIDlet
- MIDlet: `MIDlet-<n>-UserDenied: [run|stop]`
- MIDlet Suite: `MIDlet-UserDenied: [update|delete]`
- Requires ActionsDeniedPermission.

AMS
Screensaver MIDlets

- Application that can be launched automatically when the device has been inactive for a predetermined amount of time.
- `MIDlet-<n>-Type: screensaver`
- Screen Saver MIDlets receive an Event about the activation and deactivation of the Screen Saver.
- May be deactivated by a key event or implementation specific actions or timeouts.

AMS
Idle Screen MIDlets

- Idle Screen (Home Screen): A default display which is presented to the user when no other activity is taking place on the device.

AMS
Idle Screen MIDlets

- Idle Screen MIDlet is a normal MIDlet that has an additional UI component (IdleItem) rendered on the idle screen.

- `MIDlet-<n>-Type: idlescreen`

- IdleItem extends CustomItem

AMS
Idle Screen MIDlets

- Brings Java applications onto the home screen where it is more accessible to users.
- Zero-click access.
- Enables a seamless user experience.

Provisioning

- LIBlets
- RMS Interchange

Provisioning
LIBlets

- LIBlets are independently packaged sections of code and resources that can be shared between two or more MIDlets.
- LIBlets can depend on other LIBlets.
- All LIBlets run within the execution context of a MIDlet.
- LIBlets cannot be separated from a MIDlet or executed as an independent entity.

Provisioning
LIBlets

- Smaller code footprint.
- Decreased development time.
- Reduced fragmentation.
- Higher quality applications.
- Easier deployment of web services/proprietary APIs.

Provisioning
RMS Interchange

- MIDP 3.0 introduces support for provisioning RMS data to a device.
- `MIDlet-Persistent-Data-URL-<n>: <dataURL>[overwrite][encryptLocally]`
- RMS Interchange File Format
- RecordStore class supports serialization / deserialization of RMS data into this file format.

Provisioning
RMS Sharing

- MIDP 3.0 introduces more fine-grained control over sharing of RMS.
- A MIDlet may control access to the shared record store using the Application-Level Access Authorization.
- Authorization Modes
  - AUTHMODE_ANY
  - AUTHMODE_APPLEVEL
  - AUTHMODE_PRIVATE

Security

- Class-based permissions
- Application Access Authorization

Security
Class-based Permissions

- MIDP 2.x Security Review
  - Based on protection domains
  - Uses named permissions (e.g. “javax.microedition.io.CommConnection”)
  - Protection domains consist of a set of Allowed and User permissions
  - MIDlet Suites request permissions using MIDlet-Permissions / MIDlet-Permissions-Opt attributes

Security
Class-based Permissions – Overview

- Based on protection domains
- Adopts class-based permissions from JavaSE
- Protection domains consist of a set of Allowed/User permissions
- MIDlet Suites request permissions using `MIDlet-Permission-<n>` / `MIDlet-Permission-Opt-<n>` attributes

Security
Class-based Permissions

- Permission classes defined for each API requiring protection
- Subclassed from java.security.Permission
- May specify the name of a resource to protect and an associated action
- Checked by implementation during execution of protected function

Security
Class-based Permissions – Advantages

- Can specify permissions for property, resource, or function for a particular API
  - MIDP 2.x did not provide that level of granularity; only had boolean permissions
- Unifies permission definitions
  - Do not need to redefine named permissions for different configurations (CLDC, CDC, SE)

Security

Class-based Permissions – Protection Domain

- Each MIDlet Suite is bound to a protection domain when installed.
- Protection Domain defines set of Permissions that may be granted to a MIDlet Suite.
  - **Allowed** – granted (applies to requested Permissions)
  - **User** – granted by user (applies to requested Permissions)
- When Permission is checked, must check in the order User > Allowed to ensure same behavior when more than one Permission in the Policy could grant access

Security

Class-based Permissions – LIBlet Permissions

- LIBlet is executed inside protection domain of the MIDlet it is bound to.
- Permission declared with LIBlet-Permission-<n> and LIBlet-Permission-Opt-<n> are informative only
- All Permissions MUST still be declared in dependent MIDlet Suite's JAD/Manifest.

Security
Application Access Authorization

- Allows MIDlet Suites to restrict access to its shared resources from other MIDlet Suites
- Application Access Authorization is separate from the protection domain authorization model.
- Functionality using App Access Authorization:
  - RMS : Record store access
  - IMC : Establishing IMC connection
  - Events : Posting/receiving App-to-App Events

Security
Application Access Authorization

- Access can be restricted based on:
  - domain
  - vendor
  - signer
- Application Access Authorization is in Manifest:

```
MIDlet-Access-Auth-Type-<n>: domain=SELF|ANY;vendor=<vendorname>|
ANY;signer=<certalias>|ANY
```

- If <certalias> is used, it must match exactly one <alias> in…

```
MIDlet-Access-Auth-Cert-<n>: <alias> <base-64 encoded signing cert>
```

- A MIDlet Suite B has access to MIDlet Suite A's resources if it satisfies the combined domain/vendor/signer constraint in at least one of MIDlet Suite A's MIDlet-Access-Auth-Type-<n> attributes.

Security
Application Access Authorization – Example

- MIDlet Suite A Manifest

```
MIDlet-Access-Auth-Type-1:domain=SELF;vendor=ANY;signer=coolcompanycert
MIDlet-Access-Auth-Type-2:domain=ANY;vendor="Cool Company Inc.";signer=ANY
```

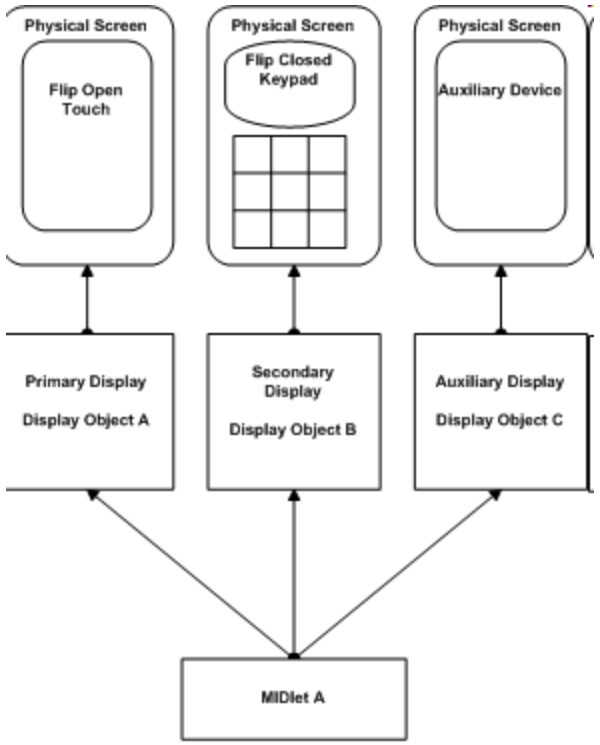
- A MIDlet Suite will be granted access if it EITHER…
  - belongs to same domain as MIDlet Suite A AND is signed by (certificate corresponding to) “coolcompanycert”, OR
  - has MIDlet-Vendor of “Cool Company Inc.”

LCDUI

- TabbedPane
- Menu
- Multiple screen support
- Animated/Scalable Images
- Mandatory PNG, GIF, JPEG, & SVG support
- Application-defined fonts
- Layout policies
- TextEditor input

LCDUI
Multiple Displays

- Previous versions of MIDP allowed for only one Display per MIDlet
- MIDP 3.0 allows for multiple Displays
- Each MIDlet will have a *primary* Display
- Each MIDlet may have any number of *secondary* Displays



LCDUI
Multi-Windows

- Two aspects of a Display resource:
  - *exclusive* – one MIDlet at a time (e.g. key events)
  - *non-exclusive* – can be shared (e.g. display)
- Three states for sharing Display resources:
  - *foreground* – access to exclusive aspects; priority on non-exclusive aspects
  - *background* – no access to any aspects
  - *visible* – at least one pixel; no access to exclusive aspects
- These definitions allow for a multi-windowed environment

LCDUI

Application-defined fonts

- MIDP 3.0 adds functionality for importing fonts packaged with the application.
- Imported fonts must be OpenType fonts with TrueType outlines
- Application-specific font packages can be used to deliver languages and look-and-feel not available on a device

LCDUI
TextEditor

- Editable text component
- Sits as a separate layer on top of Canvas or CustomItem; not part of the Graphics object
- Provides access to device's predictive text input mechanisms
- Supports the same *input constraints* and *input modes* as TextField/TextBox
- TextEditorChangeListener – events are sent on change in text, caret position, or input mode

Summary
New Features in MIDP 3.0

- Required concurrency
- Inter-MIDlet communication mechanisms
- Shared libraries (LIBlets)
- Updated Security mechanism for finer-grained policy
- New LCDUI elements and layouts
- Improvements to all APIs

Summary
More Info

- "MIDP 3.0 In Depth: Tutorials and Demonstrations"

  ID#: TS-6816
  Date: 02-JUN-09
  Time: 03:20 PM-04:20 PM
  Room: Esplanade 303

- "Mobile Information Device Profile 3" Specification

  http://www.jcp.org/en/jsr/detail?id=271

JavaOne℠

Thank You

Paul Su
paulsu@aplixcorp.com

Angus Huang
angus@aplixcorp.com

Roger Riggs
roger.riggs@sun.com

Sun microsystems