MIDP 3.0 overview

MIDP 3.0 specification status

Specification started in Q1 2005.

72 members in the expert group. 23 participant companies, 38 observer companies and 11 individual contributors.

Public review was completed March 24th, 2008.

JSR was finalized Dec 1st 2009.

New MIDP 3.0 functionality

- CLDC 1.1.1
- Mandatory JSR 135 v1.1 compliance
- Mandatory concurrency
- LIBlets/Shared Libraries
- Events
- Inter-MIDlet communication
- Notification manager
- Application Access Authorization
- Extended Permission model
- Auto-start MIDlets
- Screensaver MIDlets
- Idle screen MIDlets
- User restrictions on application life-cycle control
- IPv6 support
- Record store interchange format
- Record store encryption
- Mandatory PNG, GIF, JPEG, & SVG support
- Multiple-screen support
- Mutable Images
- Animated Images
- Application-defined fonts
- TabbedPane
- Menu
- Layout policies for Forms
- Application-specified splash screens

CLDC 1.1.1

- Maintenance release for CLDC 1.1
- Mandatory requirement for MIDP 3.0 running on CLDC
- It includes permission classes to support new class based permission model
- New Math methods that were left out the CLDC 1.1 spec
- Standardize the class verification behavior

Mandatory JSR 135 v1.1 compliance

- MIDP 2.0 shared some classes with JSR 135
- MIDP 3.0 makes JSR135 mandatory

Concurrency

MIDP 1.0/2.x did not prohibit concurrency, but did not standardize behavior

There are concurrency implementations in the market today, but behavior is fragmented

MIDP 3.0 not only defines expected concurrency behavior but makes it mandatory and adds additional functionality that allows MIDlets to run together intelligently, not just at the same time

Events

Events are a 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 constrained through the application level access mechanism.

Events

Application may be launched automatically in response to events.

- Static registration through a JAD attribute:

```
MIDlet-Event-Launch-<n>
:<Classname>;<AuthorizationMode>;<Launch-Condition>
```

`Where:`

<Classname> = The fully-qualified name of the class to launch.

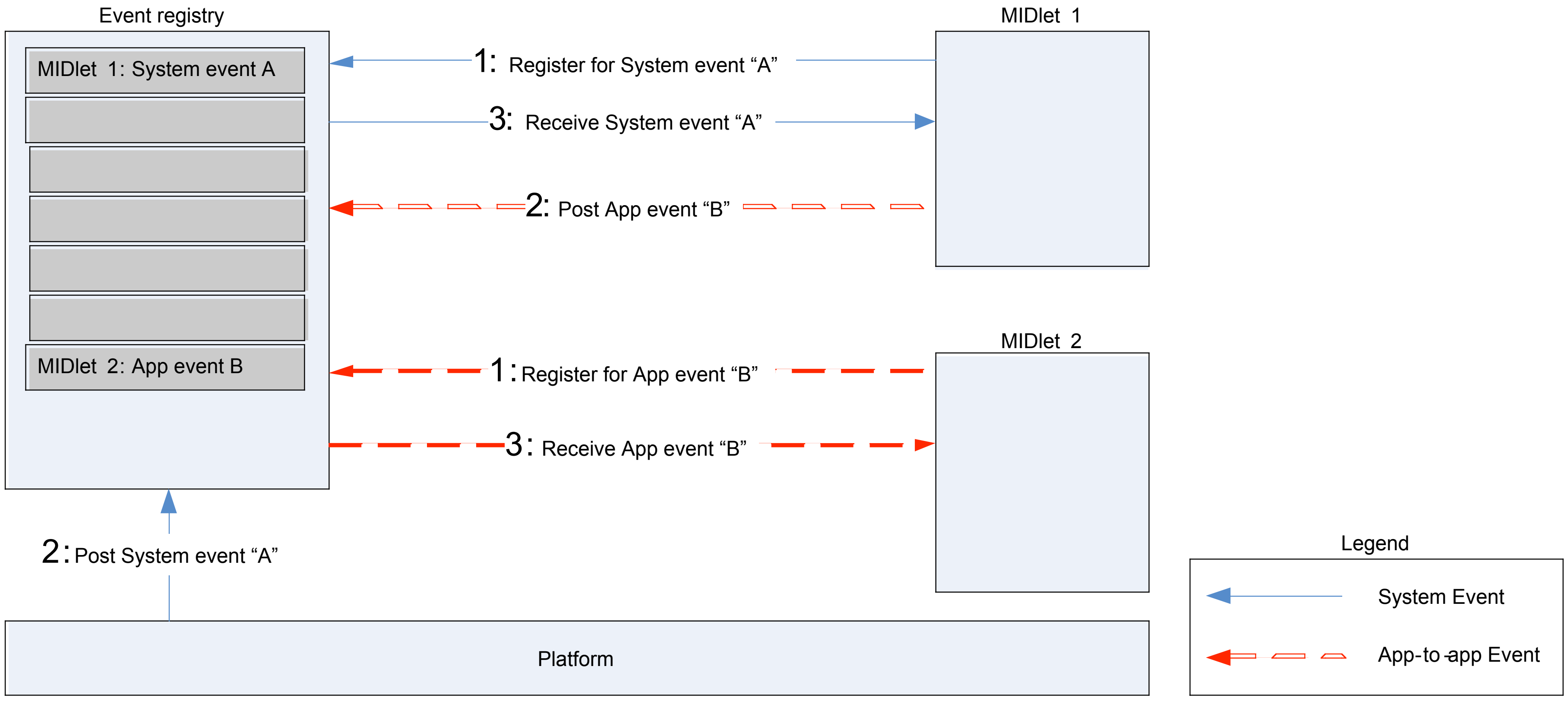
<AuthorizationMode> = true|false.

<Launch-Condition> = The condition that should cause the application to be launch.

- Dynamic registration during execution

An application can dynamically register to be launched by calling EventManager.registerApplication()

Events – transaction diagram



Events - Tutorial

Registering a Listener for an event

IMClient.java

```
EventDataListener edl = new EventDataListenerImpl();
EventManager em = EventManager.getInstance();
em.addEventListener("midp30.demo.AddContact", edl, false);
```

Creating and posting an Application-to-Application event

AddressBook.java

```
EventManager mgr = EventManager.getInstance();
EventData eventData = new EventData("midp30.demo.AddContact", 10,
    eventString, null);
try {
  mgr.post(eventData, false);
} catch (Exception e) {
  e.printStackTrace();
}
```

Inter-MIDlet Communication (IMC)

IMC APIs can be used to establish a bi-directional stream connection between two MIDlets.

Two types of IMC connections:

- Client connection
- Server connection

Access to these connections can be constrained through the application level access mechanism.

IMC clients

IMC client MIDlets can open a connection to a server MIDlet by using Connector.open() with a IMC client URI:

imc:// <MIDlet UID>/"*" : <server name> : <server version> ; [<authmode>]

Client MIDlets can specify a specific server MIDlet to connect to or can use the "*" wildcard to connect to any server MIDlet that provides the same IMC service.

IMC servers

A MIDlet acting as an IMC server can use Push register to be automatically launched when a matching client connection request is made.

```
MIDlet-Push-<n>: imc://:<server name>:<version>;<authmode>, <MIDlet
   UID>, *
```

IMC servers can use application level access authorization to restrict which client MIDlets can connect to it

LIBlets/Shared libraries

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 deployed separately from a MIDlet or executed as an independent entity. They are not dynamic services and cannot be updated independently from the dependent MIDlet.

LIBlets/Shared libraries - Dependencies

Dependencies can be declared on LIBlets, standards, or proprietary extensions.

```
MIDlet-Dependency-<n>: <type>; <level>; <name>; <vendor>;
  <version>
LIBlet-Dependency-<n>: <type>; <level>; <name>; <vendor>;
  <version>
```

Dependencies must also include the exact hash of the LIBlet and the URL.

```
MIDlet-Dependency-Jar-SHA1-<n>
MIDlet-Dependency-JAD-URL-<n>
LIBlet-Dependency-Jar-SHA1-<n>
LIBlet-Dependency-JAD-URL-<n>
```

LIBlets - Tutorial

LIBlet dependency declaration in the application JAD file

Showtimes.jad

```
MIDlet-Dependency-1: liblet; required; mapservice; Aplix Corporation Inc.; 1.0
MIDlet-Dependency-Jar-SHA-1:
    601FD2020CF43821D33F85BA9021E273C4B22D26
MIDlet-Dependency-JAD-URL-1: mapservice
MIDlet-1: Showtimes, Showtimes.png, Showtimes
MIDlet-Jar-Size: 7654
MIDlet-Jar-URL: Showtimes.jar
MIDlet-Name: Showtimes
MIDlet-Vendor: Unknown
MIDlet-Version: 1.0
```

LIBlet JAD file

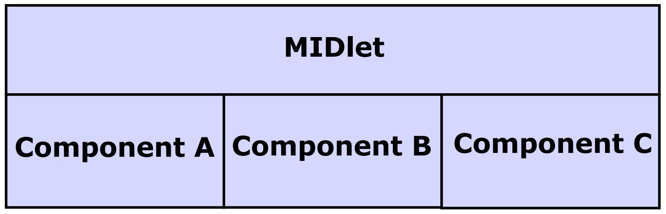
Mapservice.jad

```
LIBlet-Jar-URL: mapservice.jar
LIBlet-Name: mapservice
LIBlet-Vendor: Aplix
LIBlet-Version: 1.0
```

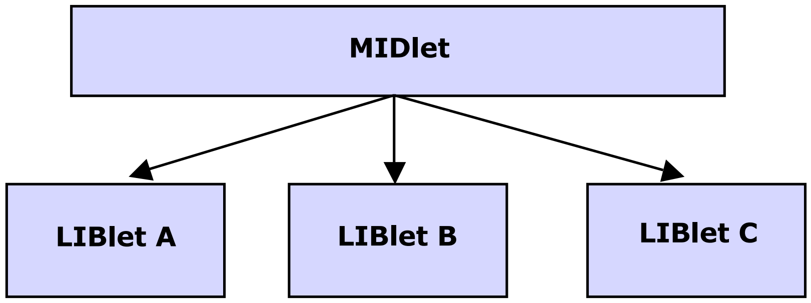
LIBlets - Advantages

- Decreased application development cost.
- Decreased application development time.
- Increased Average Revenue Per Unit.

Old MIDlet Model



New MIDlet Model w/ LIBlets



In the "Old MIDlet Model", Components A, B, C must be developed and supported by the MIDlet owner, who may or may not have expertise in those components.

The "New MIDlet Model" can make use of reusable software components – LIBlets A, B, C – that are developed by 3rd party experts who are more efficient in creating them.

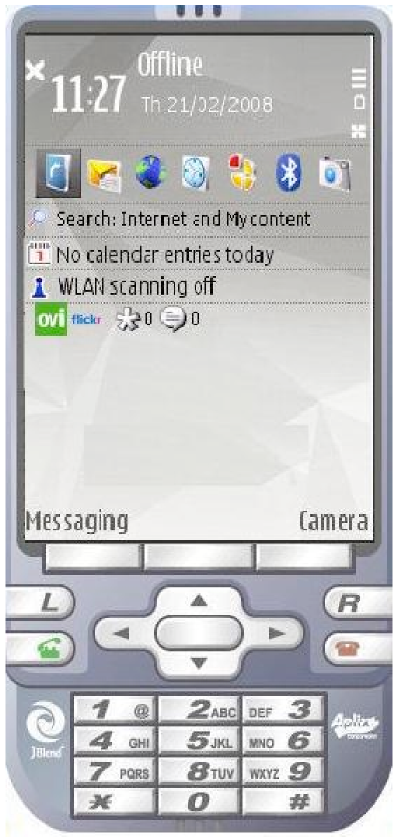
Auto-Start MIDlets

MIDP3.0 allows developers to have greater control over the lifecycle management of the MIDlet. With the Auto-Start feature, developers can have their MIDlets automatically launched at power-up of the phone

To protect the phone from malicious auto-start MIDlets, only MIDlets that have the javax.microedition.midlet.AutoStartPermission is allowed to auto-start.

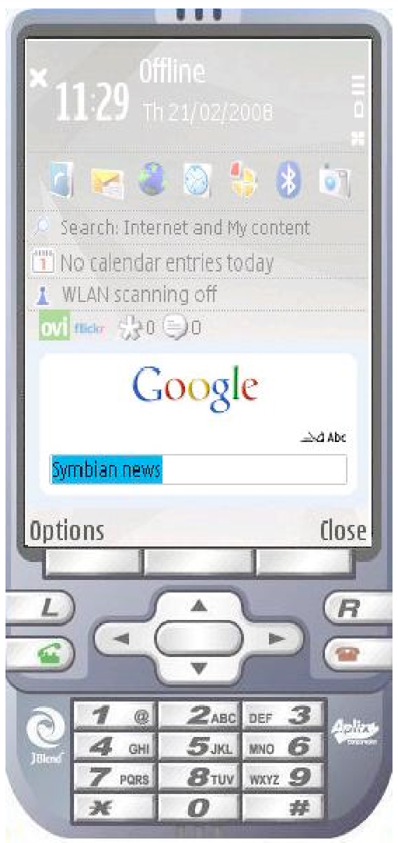
AMS MUST attempt to restart Auto Start MIDlets on exit/termination of the MIDlet.

Idle Screen


Offline
11:27
Th 21/02/2008
Search: Internet and My content
No calendar entries today
WLAN scanning off

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

Idle Screen - Importance



- The idle screen is the screen that is most often visible.
- It enables operators and manufacturers to push information or advertising to the user.
- Idle screen applications require fewer clicks to access.

Example:

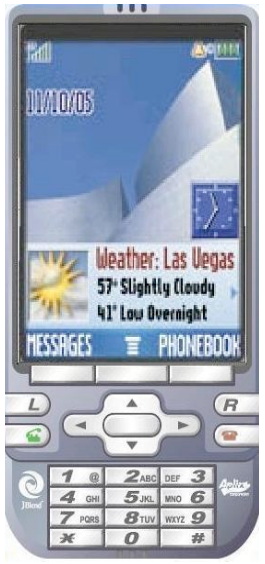
- Google's native search application has increased mobile phone searches by 20 percent[1].

- Searching through the idle screen is on average 40 percent quicker from Google's native search application than from the device's web browser[1].

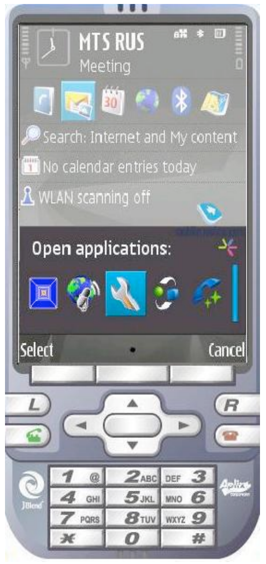
[1] Mikael Ricknäs, IDG News Service
"Google Reports Jump in Mobile Search", March 19, 2008 <http://www.pcworld.com/article/id,143590-c,google/article.html>

Idle Screen - Existing Implementations

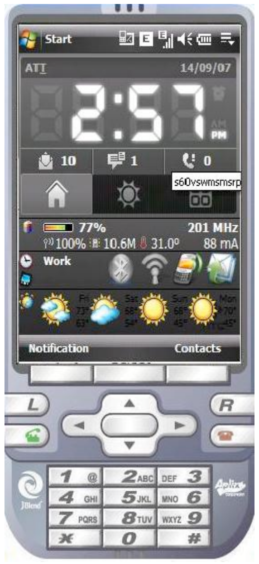
SCREEN3



Nokia Active idle



Windows Mobile



Motorola's SCREEN3 technology

– Content is categorized into channels and then delivered to subscribers via their mobile devices

Nokia Active Idle

– Application shortcuts are placed on the idle screen for faster access.

Windows Mobile

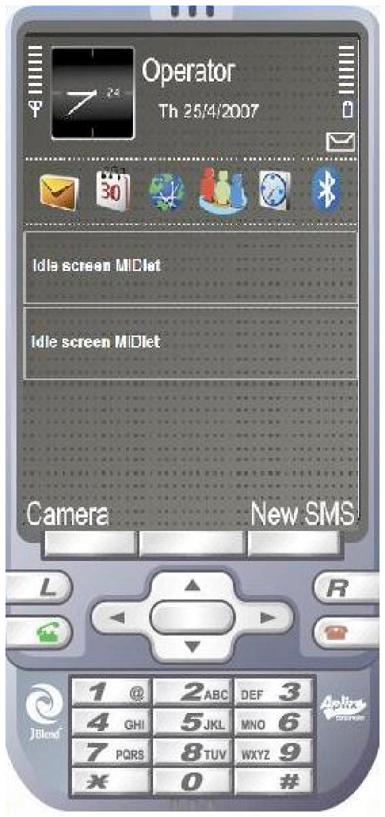
– Applications can be placed on the idle screen for faster access.

Idle Screen MIDlets – Why in MIDP 3.0?

MIDlets can access greater functionality via existing JSRs on the device.

MIDP has a wider addressable market than platform-specific idle screen implementations.

Idle Screen MIDlets



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

The platform decides how Idle Screen MIDlets are positioned on the idle screen.

Access to the idle screen is controlled by the security policy of the device.

Idle Screen MIDlets

Idle Screen MIDlet is identified with JAD/JAR attribute

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

Idle Screen MIDlets add or remove the content on the idle screen via **Display.setIdleItem(IdleItem)** method.

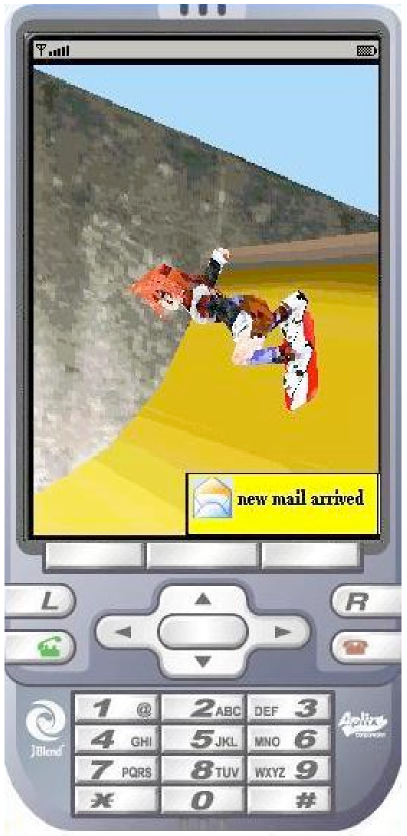
Notification

Notification: A small unobtrusive informational note to be shown to the user, without necessarily taking control of the screen.

When the **Notification** is inspected, the callback methods are called on the **NotificationListener** registered with the **Notification.**

A **Notification** is created with **NotificationType,** which represents the type of the Notification. A **Notification** can have a **String** label and an **Image.**

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 Notification when a new e-mail has arrived.

Splash Screen

Splash Screen is visible to the user after the MIDlet is launched.

MIDlet can specify Splash Screen with JAD/JAR attribute:

```
MIDlet-Splash-Screen-Image: /cardgamesplash.png
```

The Splash Screen will remain visible until one of the following events occur:

- The MIDlet shows a Displayable by calling setCurrent on its primary Display
- The MIDlet's startApp method returns
- The MIDlet is no longer in the foreground

Splash Screen

<u>SubmenuCommand.jad</u>

```
MIDlet-1: SubmenuCommand, , SubmenuCommand
MIDlet-Name: SubmenuCommand
MIDlet-Vendor: Aplix Corp
MIDlet-Version: 1.0
MIDlet-Jar-URL: SubmenuCommand.jar
MIDlet-Jar-Size: 60587
MicroEdition-Configuration: CLDC-1.0
MicroEdition-Profile: MIDP-3.0
MIDlet-Splash-Screen-Image: /aplixlogo.png
```

IPv6

Certain classes must now be able to support parsing addresses in either IPv4 or IPv6 format.

- Connector
- HttpConnection
- HttpsConnection
- SecureConnection
- SocketConnection
- UDPDatagramConnection

Implementations are not required to support IPv6 but must be able to support the address and recognize the format

Applications can specify which version they require or allow both through the `MIDlet-Required-IP-Version` JAD attribute

User Restrictions on MIDlet Suite Lifecycle

MIDP3.0 allows the developer to determine how much control the user has over the lifecycle of a MIDlet Suite

The developer can limit the user from

- Deleting a MIDlet Suite
- Updating a MIDlet Suite
- Starting a MIDlet
- Stopping a MIDlet

This functionality can only be used if the MIDlet has the javax.microedition.midlet.ActionsDeniedPermission

Application Level Access Authorization

Application Level Access Authorization allows MIDlet suites to restrict access to its shared resources from other MIDlet Suites

Application Level Access Authorization is *separate* from the **protection domain** authorization model.

Functionality using App Level Access Authorization

1) **RMS** : RecordStore access
2) **IMC** : Establishing IMC Connection
3) **Events** : Posting and Receiving App-to-App Event

Application Level Access Authorization

<u>Access can be restricted based on…</u>

1) **domain**
2) **vendor**
3) **signer**

Application Level Access Authorization is declared 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.

Application Level 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...

a) belongs to same domain as MIDlet Suite A AND is signed by (certificate corresponding to) "coolcompanycert“, OR…

b) has MIDlet-Vendor of "Cool Company Inc."

Permissions Model

MIDP 2.X Security Review

- Based on protection domains
- Uses named permissions (e.g. "javax.microedition.io.CommConnection")
- MIDlet Suites request permissions using `MIDlet-Permissions` / `MIDlet-Permissions-Opt` attributes

MIDP 3.0 Security Overview

- Based on protection domains
- Adopts class-based permissions from JavaSE
- MIDlet Suites request permissions using `MIDlet-Permission-<n>` / `MIDlet-Permission-Opt-<n>` attributes

Permissions Model

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

Advantages of new Permissions Model

- 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

- Unified permissions definitions

  Do not need to redefine named permissions for different configurations (CLDC, CDC, SE)

Permissions Model

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.

1) **Allowed** - granted (applies to requested permissions)
2) **User** - granted by user (applies to requested permissions)

Permissions Model

Requesting Permissions

MIDlet Suites that require access to a protected API MUST request the corresponding permission using…

```
MIDlet-Permission-<n>: <classname>, <resource>, <action>
```

Non-critical APIs should be requested using…

```
MIDlet-Permission-Opt-<n>: <classname>, <resource>, <action>
```

The `<resource>` and `<action>` values should only exist if the associated Permission class constructor has corresponding parameters.

```
MIDlet-Permission-1: javax.microedition.io.HttpProtocolPermission http://
MIDlet-Permission-2: javax.microedition.io.SocketProtocolPermission socket://:4321
```

Permissions Model

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.

If LIBlet declared permissions do not exist in dependent MIDlet Suite requested permissions, installation MUST fail.

Screen Saver MIDlets

A Screen Saver application can be launched automatically when the device has been inactive for a predetermined amount of time.

Screen Saver MIDlets are identified with JAD/JAR attribute:

```
        MIDlet-<n>-Category: screensaver
```

Screen Saver MIDlets receive an **Event** about the activation and deactivation of the Screen Saver.

Events:

Name	*Value*
"SCREENSAVER_MODE"	"SCREENSAVER_MODE_ACTIVATED
"SCREENSAVER_MODE"	"SCREENSAVER_MODE_DEACTIVATED"

Screen saver MIDlets

Screen Saver MIDlet may be deactivated by a key event or implementation specific actions or timeouts.

If the screen saver MIDlet uses functionality that is protected with the security policy, then the security prompts SHOULD NOT deactivate the screen saver.

Like a normal MIDlet, if the screen saver MIDlet satisfies security requirements for specific APIs, the APIs MUST be accessible.