MOTODEV

The Motorola developer network

MIDLET LIFECYCLE ON MOTOROLA LINUX OS DEVICES

V 1.1

TECHNICAL ARTICLE

MIDlet Lifecycle on Motorola Linux OS Devices

Version 1.1

June 2007

For the latest version of this document, visit http://developer.motorola.com

Motorola, Inc.

http://www.motorola.com

Contents

Figures

Introduction

The Mobile Device Information Profile (MIDP) version 2.0 for Java Micro Edition (JavaME) specification defines the architecture and associated APIs required to enable development of JavaME applications (MIDlets) that can be run on Motorola Mobile Devices. The MIDP profile was developed with the flexibility to allow device manufacturers and carriers the ability to customize their application environment based on the needs and capabilities of their devices.

In continuing our efforts to improve the experience on Motorola mobile devices, the Linux-Java platform utilizes the flexibility allowed by the MIDP specification to provide new Java capabilities that are not available on devices running the Motorola Proprietary Operating System. One of the key, functional differences is the support for background execution of MIDlets. This new capability does, however, require MIDlet developers to understand the implementation of the feature and thus MIDlet developers must take the steps necessary to ensure the MIDlet produces the desired experience.

This document outlines the changes in the Java KVM architecture which allows background running capabilities of MIDlets on the Linux-Java platform and provides information to help developers create MIDlets for the platform.

MIDlet Lifecycle Behavior and Background Execution Support

The latest Motorola Operating System (OS) device range does not support “Background” execution.

On this family of devices, a MIDlet can only have 4 states:

1. Not running
2. Active
3. Suspended (run state saved with MIDlet dormant)
4. Destroyed

A running MIDlet can have the following three states: started, suspended, or destroyed.

Started

Started from the “Games” menu in the device UI.

Suspended

- Suspended from a running state by the user pressing the red end key followed by selecting the “Suspend” option from the AMS (Application Management Software) menu.
- Suspended from a running state by the user closing the handset flip (clamshell style devices only).
- Suspended from a running state by an interruption event such as an incoming phone call.

Destroyed

- Destroyed from a running state via pressing the red end key followed by selecting “End” from the AMS menu.
- Destroyed from a running state via a user directive from the MIDlet menu.
- Destroyed from a running state via abnormal MIDlet termination.
- Destroyed form a suspended state due to device power off.

NOTE: A running MIDlet CAN NOT be set to run in the background without focus and allow application multi-tasking.

The MIDlet lifecycle state machine on the Motorola OS devices is shown below.

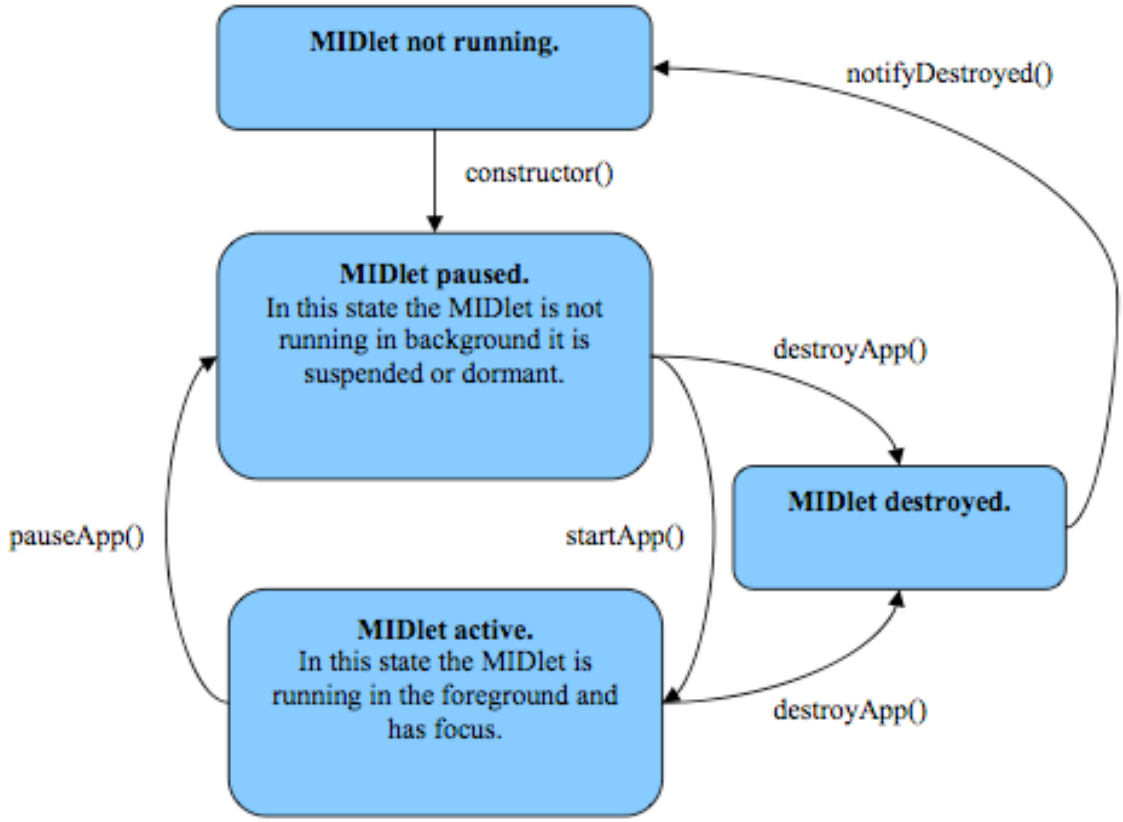

Figure 1: Motorola OS device MIDlet lifecycle state machine

New Motorola Linux OS MIDlet Lifecycle Behavior and Background Execution Support

Following the paradigm of Linux-Java to provide a multi-tasking environment, devices developed on the Linux-Java Platform will allow MIDlets to continue running in the background while the device is being used for other activities. This will permit an application such as an Email client, a music player, or an Instant Messaging application to continue running in the background, keep necessary network resources, and even play sounds or music while it is not in the foreground with focus.

On this family of devices, a MIDlet can have five states:

1. Not running
2. Active
3. Suspended (run state saved with MIDlet dormant) controlled programmatically in MIDlet using Lifecycle methods
4. Background execution (running without focus)
5. Destroyed

A running MIDlet can be started, suspended, run in background, and destroyed.

Started

- From the “Games” menu in the device UI.

Suspended

- From a running state via the user pressing the red end key followed by selecting the “Go to idle …” option from the AMS (Application Management Software) menu. However, the MIDlet must be built to specifically handle a suspend/pause MIDlet state by way of the `pauseApp()` and `startApp()` MIDlet lifecycle methods.
- From a running state via the user closing the handset flip (clamshell form factor devices only). However, the MIDlet must be built to specifically handle a suspend/pause MIDlet state by way of the `pauseApp()` and `startApp()` MIDlet lifecycle methods.
- From a running state by an interruption event such as an incoming phone call. However, the MIDlet must be built to specifically handle a suspend/pause MIDlet state by way of the pauseApp() and startApp() MIDlet lifecycle methods.

Run in Background

- Execution mode from a running state by the user pressing the red end key followed by selecting the “Go to idle …” option from the AMS (Application Management Software) menu. No special

action needs to be handled programmatically but be advised any code in the `pauseApp()` or `startApp()` methods will be executed.

- Execution mode from a running state by the user closing the handset flip (clamshell form factor devices only). No special action needs to be handled programmatically but be advised any code in the `pauseApp()` or `startApp()` methods will be executed.

Destroyed

- From a running state via the pressing the red end key followed by selecting “Close Application” from the AMS menu.
- From a running state via a user directive from the MIDlet menu.
- From a running state via abnormal MIDlet termination.
- From a suspended state due to device power off.

NOTE: A running MIDlet CAN be set to run in background without focus and allow application multi-tasking.

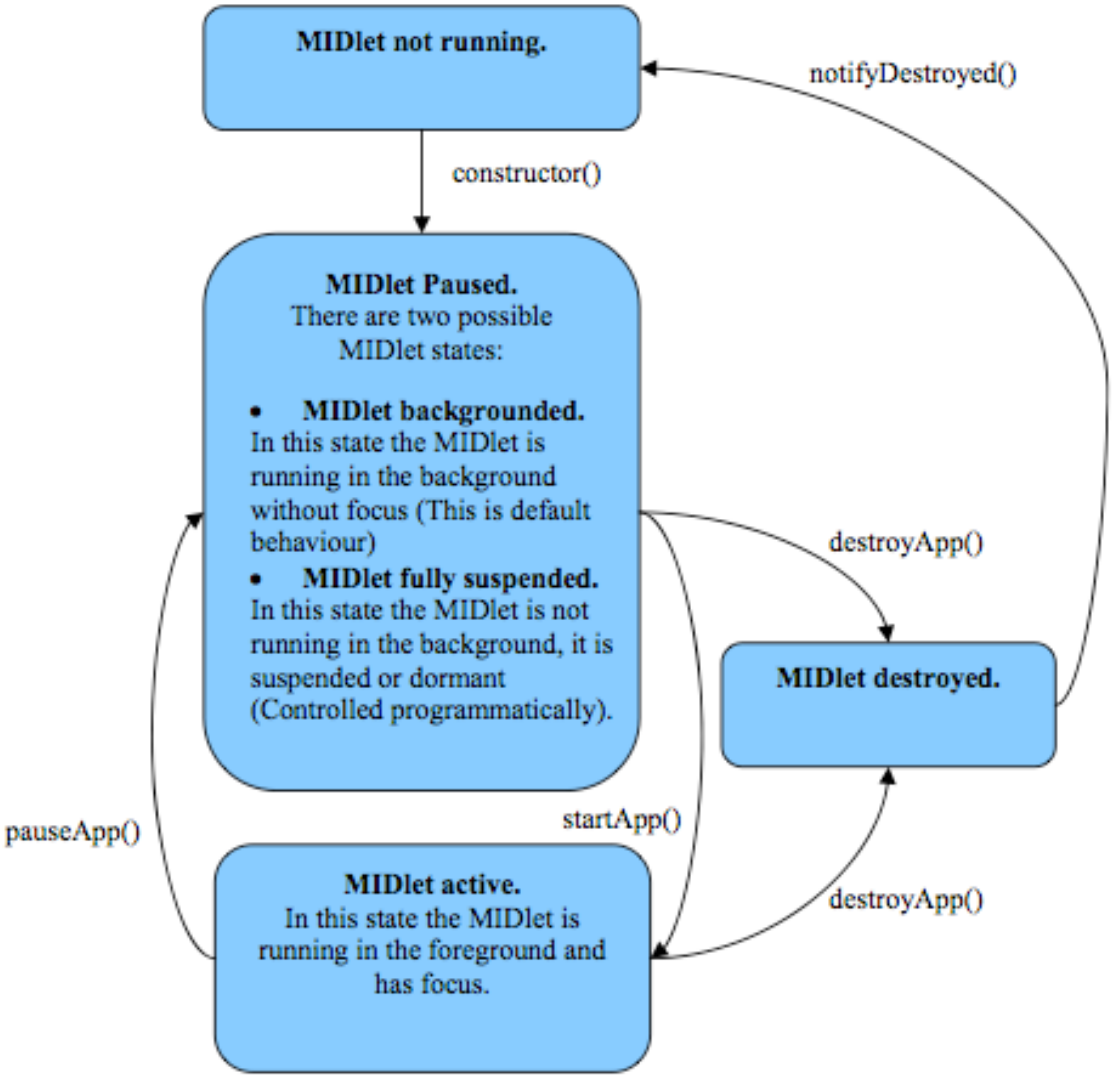

Figure 2: Motorola Linux OS device MIDlet lifecycle state machine

Entering and Exiting Background Mode

Beginning with the first device released on the Linux-Java platform (MOTOROKR Z6), MIDlets are able to run in the background. A MIDlet is informed that it is being placed in the background via the `pauseApp()` method. If the MIDlet does not pause itself or release any resources, the MIDlet will continue running in the background. If any resources (such as audio) are needed by a higher priority system function (such

as a voice call), the needed resources will be taken away from the MIDlet to service the system requirement.

It is advisable for the MIDlet developer to keep in mind and code a MIDlet to accommodate possible scenarios such as an incoming voice call which results in multi media player resources being taken from a running MIDlet and given to the system. Failing to do this correctly will result in poor user experience or incorrect MIDlet behavior.

By definition, once the `pauseApp()` method is called, the MIDlet is in the Paused state. Per the MIDP 2.0 specification, a MIDlet "SHOULD not be holding or using any shared resources". It is the responsibility of the MIDlet to release unneeded resources when it is "paused". However, since this is a recommended practice and not a mandatory requirement of the specification, a MIDlet may retain resources necessary to allow it to function.

Since the MIDP specification was created to support a minimum set of functionality that may or may not support background execution of MIDlets, special care must be taken when implementing a background executing MIDlet. The recommendations below must be followed closely when developing MIDlets for a Motorola Linux-Java device to ensure a good, reliable, user-friendly experience.

Flip Behaviors

On Motorola OS devices it is possible to instruct the AMS to ignore a flip closure/open event and allow a MIDlet to continue to run unaffected by the flip operation. To do this in a Motorola OS device the JAD attribute "FlipInsensitive" can be used and set to "True". The following line can be added to the JAD file:

```
FlipInsensitive: True
```

However, on the Motorola Linux OS devices this behavior is currently not available. It is planned and will be available in the very near future. Please reference the MOTODEV website at http://developer.motorola.com for news on when this feature is available in the Motorola Linux OS devices.

This means that on Motorola Linux OS devices when the flip is used the MIDlet will lose focus (it can still be running in the background) and the user will have to go to the Games menu to bring the focus back to the MIDlet when the flip is reopened. There will be visual indication to the user that the MIDlet is running in the background by way of a coffee cup in the status bar at the top of the device screen.

MIDlet Development Recommendations

Pausing a MIDlet

For developers of games or other applications that do not have a need or use for background mode execution, it is strongly recommended that the guidelines defined in the MIDP 2 specification be followed. Therefore, when the `pauseApp()` method is called by the system, the MIDlet MUST pause the game or application, release resources (for example, audio) that are not needed while the game is paused, and take any necessary steps to preserve system or user data. Unlike the Java environment provided by devices running Motorola's proprietary operating system, the Motorola Linux-Java platform will not automatically pause the execution of Java MIDlets based on calling `pauseApp()`. It is the responsibility

of the 3rd party Java developer to stop the execution of the application once the device informs the MIDlet to pause.

Resuming a MIDlet

For developers of games or other applications that do not have a need or use for background mode execution, following a suspend/pause and call to the `pauseApp()` method, the resources (for example, audio) that are not needed may have been released (recommended). On resuming the MIDlet it will therefore be necessary to re-instate these resources and system or user data to the exact same state at the point of suspension in order to give a good user experience. This is done by utilizing the `startApp()` MIDlet lifecycle method. The MIDlet can then continue its execution cycle.

Utilizing Background Mode

For developers that wish to make use of background mode on the Motorola Linux OS platform, the MIDlet must be designed with the understanding that a `pauseApp()` request will be passed to the MIDlet when the MIDlet is placed in the background. This call can be made due to a number of reasons such as incoming voice calls or flip/slider closure. The user can also manually put the application in the background by pressing the red END key and selecting the option from the AMS pop-up menu to return to IDLE with the MIDlet running. The `pauseApp()` method will be called once the background execution (Idle) option is selected from the AMS pop-up menu.

A MIDlet that wishes to run in the background must respond to the `pauseApp()` method call as required in the MIDP 2.0 specification, but can continue to utilize the resources it needs to perform necessary functions in the background. Since the MIDlet is in the Paused state once it responds to the `pauseApp()` method, it is highly recommended that any unnecessary processing, such as display updates, be stopped. This will also help provide a better overall user experience on the mobile device.

The MIDlet's `startApp()` method will be called to bring the MIDlet back to the Active state. When this method call is received, the MIDlet should request any necessary resources, resume updates to the display, and continue executing in the foreground of the device.

1 Audio Resource Handling

Executing in background mode does not guarantee audio resources will always be available. If a higher priority task occurs on the phone, such as a phone call is placed or received, the audio resources will be taken away. The system will inform the MIDlet that the audio is unavailable via a `DEVICE_UNAVAILABLE` event. The MIDlet will be informed when audio resources are made available again via a `DEVICE_AVAILABLE` event. The MIDlet should be able to process the event and take necessary actions to provide an optimal user experience. This could include pausing and restarting audio, stopping streaming audio, or replaying notifications once the audio resources are available again.

If the MIDlet is not concerned with the loss of audio resources while in background mode, it is highly recommended that the MIDlet re-request audio resources once the MIDlet is placed back into the Active state (`startApp()` is called). This should be done even if the audio resources were available at the time when `pauseApp()` was called.

If audio resources are not available when the MIDlet starts, it would be the MIDlet's responsibility to provide any error messages to the user and/or terminate itself.

2 Network Resource Handling

Executing in background mode does not guarantee network resources will always be available. If network resources are lost for any reason, while the MIDlet is executing in the background, an exception will be thrown when the MIDlet attempts to read or write over the lost connection.

3 Threads

Many MIDlet developers create multi-threaded MIDlets. These threads can be used to control a network connection, a media player, a canvas, etc. When entering the paused state any threads additional to the main MIDlet thread should be stopped, when leaving the paused state to a normal running state any stopped threads should be restarted again.

4 Record Management Store

MIDlets often store "persistent" data in the Record Management Store (RMS), this data is game information that the MIDlet must not lose between invocations such as a high score table or game state. In the event of an abnormal MIDlet termination when in background running mode, it is recommended that the RMS is closed and resources are released when a MIDlet enters a paused state. These resources should then be reallocated when they are needed by the MIDlet or when the MIDlet leaves the paused state and returns to a normal running state.

Potential Conflicts

Many MIDlet developers use methods that could cause potential conflicts.

- The `showNotify()` and `hideNotify()` methods to control pause/suspend behavior of MIDlets utilizing a `Canvas` or `GameCanvas`.
- The `Displayable.isShown()` method to determine if a `Form` or `TextBox` object is visible to the user.

These methods can be used by the developer to control the MIDlet with respect to lifecycle states but care must be taken to ensure that there are no resource conflicts introduced between the `pauseApp()` and `hideNotify()` methods and the `startApp()` and `showNotify()` methods.

- Often `pauseApp()` is used to deallocate resources and objects as is `hideNotify()`.
- Subsequently `startApp()` is used to reallocate resources and objects as is `showNotify()`

Conclusion

Motorola has implemented a slightly different lifecycle architecture in the latest Motorola Linux OS device. The behavior is subtly different from the Motorola OS device range KVM architecture in the following ways.

- There is no suspend function built into the AMS.

- A MIDlet must programmatically accommodate any necessary MIDlet pause state (especially needed for game MIDlets).
- A MIDlet can be sent to background execution mode by user instruction or flip action.
- Shared resources should be released before a MIDlet is placed into background mode wherever possible.
- Where audio resources are required to run in background execution mode the MIDlet should listen for media player events which prevent the MIDlet from having exclusive access to the Media Player.
- Care should be taken when releasing and reallocating resources and objects in the lifecycle methods (`pauseApp()` and `startApp()`) and the `showNotify()` and `hideNotify()` methods to avoid conflicts and unwanted object creations/initializations.
- Flip behavior cannot currently be ignored – the `pauseApp()` and `startApp()` methods will be called when the flip is closed/opened and the MIDlet will lose focus.

References

JSR 118: Mobile Information Device Profile 2.0
Version 2.1
http://www.jcp.org/en/jsr/detail?id=118