

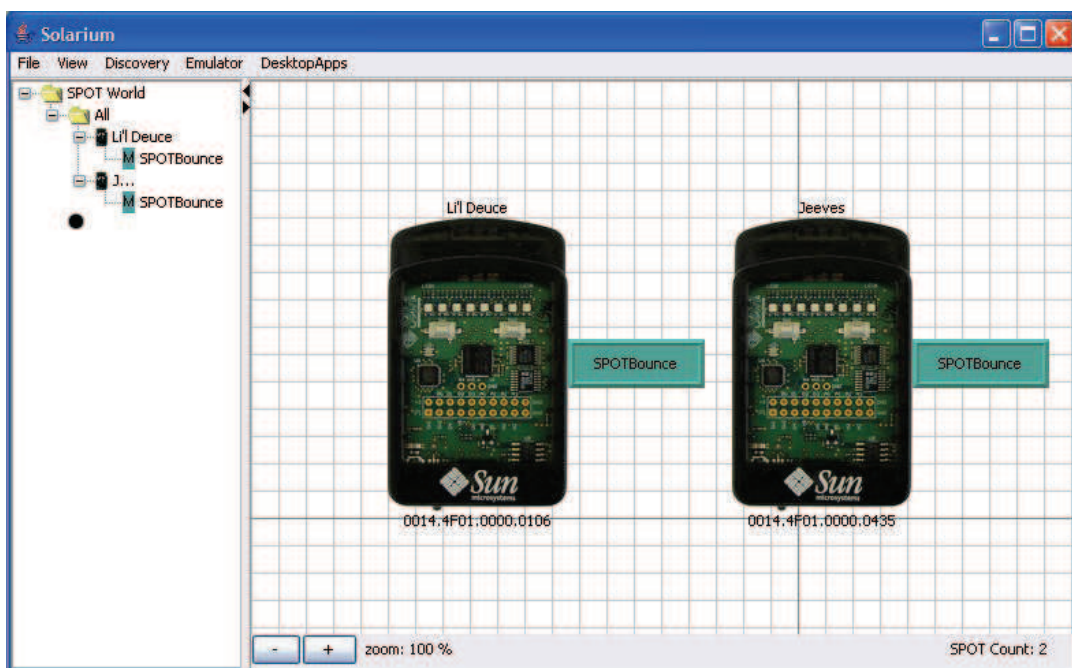
# Solarium Tool

Solarium is a host workstation application that manages application software on Sun SPOTs.

Solarium can be started from the Solarium tab in SPOTManager by clicking the Solarium button. You can also start Solarium by opening a command line window, navigating to any SPOT project directory (containing `build.xml` file) and executing the command `ant solarium`.

The Solarium application may take a few minutes to find all of the SPOTs that are within radio range and running the OTA command server. To learn how to start the OTA command server on a SPOT, see “Sun SPOTs Tab” on page 11.

After Solarium finds the SPOTs, the tool displays them and the applications they are running, as shown below.



You can display SPOTs in a tree-like Inspector view, shown here to the left, or a 2-dimensional SPOT view, shown here against the quadrille background to the right. The views are controlled through the View pull-down menu on the menu bar.

The first time a SPOT displays in Solarium, the IEEE numerical address is used for the SPOT’s name.

## Isolates and Jar Files

Before we discuss the things that can be done with SPOTs in Solarium, we need to introduce the topics of applications, MIDlets, isolates, and `jar` files.

In Java SE, an application consists of a static `main()` method defined in one of the loaded classes or it extends the `Applet` class if it is to run in a browser. In Java ME, an application is defined as a class that extends the `MIDlet` class. The Squawk VM used by Sun SPOTs implements Java ME, so all Sun SPOT applications extend the `MIDlet` class.

In standard Java ME, only one application can be run at a time in a Java VM, though that application may consist of many threads. Squawk allows for multiple applications to be run together in a single SPOT and uses a special *Isolate* class to prevent one application from interfering with the execution of another. Each `MIDlet`-based application is run in a separate isolate. While one isolate cannot directly access the instances in another, they all share the same underlying SPOT resources.

Some resources are unique, such as a radio connection on a specific port number. The first isolate to ask for that port is successfully given access to it, while any subsequent requests from other isolates fail. Other resources are truly shared such as the LEDs: one isolate might turn an LED on, and then another might turn it off or change its color.

The isolate controlling the Sun SPOT at the system level is the *master isolate*. There is only one master isolate per SPOT. The master isolate controls the radio stack for the SPOT and access to the other components of the SPOT. Normally a SPOT application consists of a single `MIDlet` that is run in the master isolate.

Solarium allows you to start and stop multiple `MIDlets` on a SPOT. These `MIDlets` are run as *child isolates*. Child isolates may suffer a slight performance penalty; for example, their radio transmissions have additional overhead because the child isolate must send requests to the master isolate to access the radio.

Java ME allows you to package up several `MIDlets` into a single *jar file*. The `MIDlets` must all be listed in a special file, `manifest.mf`. The `manifest.mf` file can be found in your project's `resources/META-INF` directory.

When you add a new `MIDlet` to your project you must add a new line for it in the manifest file. If you use the SPOT modules for NetBeans, you can use the pop-up menu on the package (select the *New > File/Folder* command and then select *MIDlet Class* from “Java Classes”) to create the new `MIDlet`. NetBeans automatically updates the manifest file. NetBeans also updates the manifest if you rename an existing `MIDlet`.

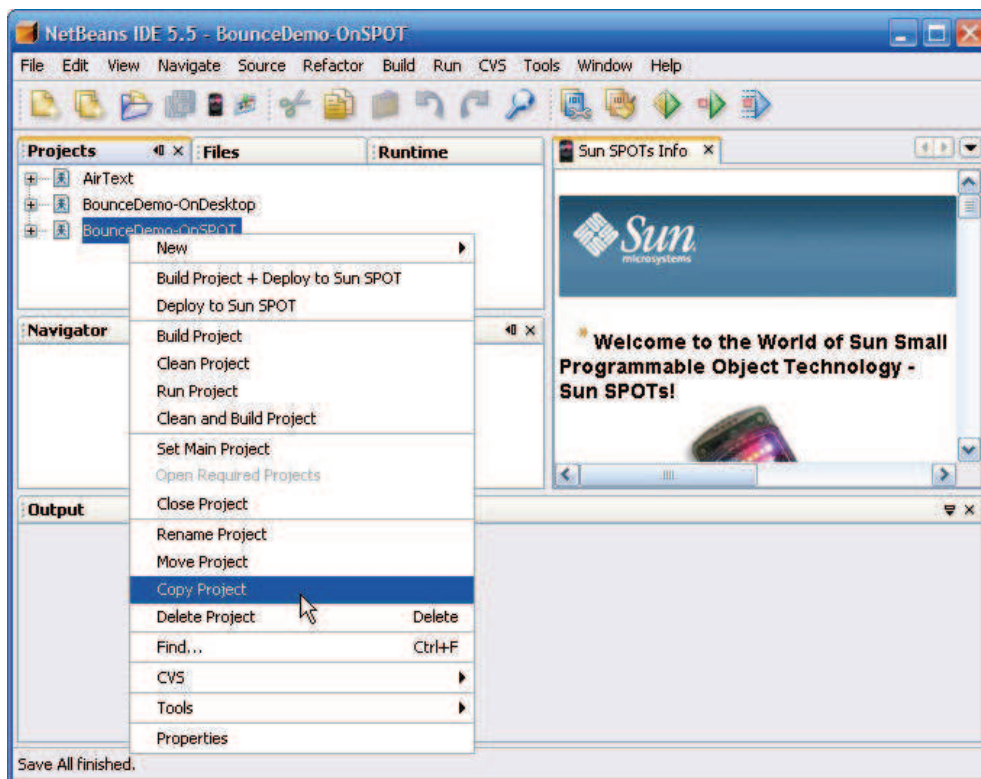
When the `jar` file is deployed to a Sun SPOT, all of the `MIDlets` are then available to be started from Solarium. One of them may be specified to be automatically run in the master isolate when the SPOT is rebooted. For now only one `jar` file may be deployed on a SPOT, but in the future it may be possible to deploy several.

### *Example: Creating a Jar File with Multiple MIDlets*

With this theoretical background in place, we will briefly go through an example of the creating of a `jar` file with two `MIDlets`. We will use two existing demonstration projects: the Bounce demo and the AirText demo. If you have worked your way through the Sun SPOT tutorial, as we strongly suggest, you will be familiar with these demos.

Briefly, we will copy the classes from the Bounce demo and the AirText demo into a new project, then edit the manifest file. After that, we will compile the new `jar` file and it will be available to deploy to a Sun SPOT.

1. Open NetBeans. Select the *Project* tab in the upper left, and select the “BounceDemo-OnSPOT” project. Right click the project and select *Copy Project*.



A dialog box displays, allowing you to specify the new project name. Here, we give it the name `ExampleJar`. If we were to compile this project now, we would get a `jar` file with one MIDlet in it, the Bounce demo. Now we need to add the AirText demo to the same project.

2. Select the `AirText` demo, click the “+” to its left, then open up the `src` selection with in it, then open the `org.sunspotworld.demo` selection within `src`. Select all four Java classes, right click, and select *Copy*.



3. Select the `ExampleJar` project, click the “+” to the left of it, then open up the `src` selection with in it, select the `org.sunspotworld.demo` node within `src`, right click `org.sunspotworld.demo`, and select *Paste*.

The `AirText` classes should now be copied into the `ExampleJar` project.

4. Select the Files tab in the upper left of NetBeans. Open the ExampleJar project, and the resources node within it. Open the resource META-INF and open the manifest.mf file with it.



The manifest file displays in a new window to the right:

```
MIDlet-Name: eSPOT Bounce Demo-OnSPOT
MIDlet-Version: 1.0.0
MIDlet-Vendor: Sun Microsystems Inc
MIDlet-1: ,, org.sunspotworld.demo.SPOTBounce
MicroEdition-Profile: IMP-1.0
MicroEdition-Configuration: CLDC-1.1
```

The format of the manifest file is

<property-name>: <space><property-value>

The individual MIDlets are specified with property names of “MIDlet-1,” “MIDlet-2,” and so on. The MIDlet property value is a string of three comma-separated, arguments. The first argument is a name for the application, the second is intended to define an icon, but is not used in Sun SPOTS, and the third specifies the application’s main class.

We will edit the manifest file to give it a new MIDlet name and tell it where to find the AirText demo. We will also give the MIDlet descriptions some user-friendly names.

5. Replace the MIDlet name with “Example Jar with two MIDlets.” Edit the MIDlet-1 line to include the phrase “Bounce Demo” between the colon and the first comma. Add a second line, below it, that reads:

```
MIDlet-2: Air Text Demo, ,org.sunspotworld.demo.AirTextDemo
```

The manifest file should now read:

```
MIDlet-Name: Example Jar with two MIDlets
MIDlet-Version: 1.0.0
MIDlet-Vendor: Sun Microsystems Inc
MIDlet-1: Bounce Demo,, org.sunspotworld.demo.SPOTBounce
MIDlet-2: Air Text Demo, ,org.sunspotworld.demo.AirTextDemo
```

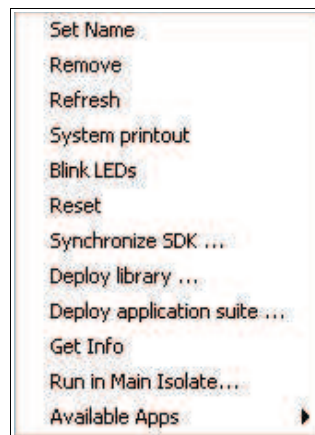
MicroEdition-Profile: IMP-1.0  
MicroEdition-Configuration: CLDC-1.1

## 6. Save the new version of the manifest file.

If you were to compile the project now, you would have a single jar with two MIDlets in it. We can use that jar file in Solarium.

## Manipulating Sun SPOTs in Solarium

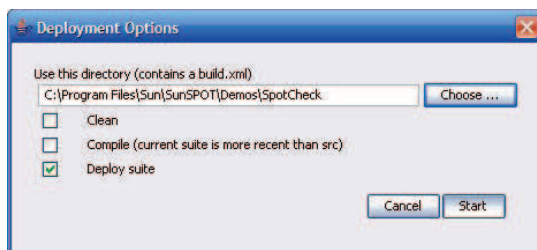
If you right-click a SPOT, the following menu displays:



The choices in this menu are:

- Set Name – Allows you to specify a name to replace the IEEE number in Solarium displays. This name is stored in persistent memory on the SPOT itself.
- Remove – Removes this SPOT from the Solarium display. Removed SPOTs can be added again through the Discovery menu on the main menu bar.
- Refresh – Inspects the SPOT for name and system information.
- System printout – Opens a window to display standard output from the main isolate operating on that SPOT.
- Blink LEDs – Blinks all of the LEDs on the SPOT 10 times, except the power LED. This is useful for finding a particular SPOT among several.
- Reset – Resets the SPOT. It has the same effect as pressing the *Control* button. It restarts the master isolate.
- Synchronize SDK – Downloads the system software from the host workstation's active SDK to the SPOT. This option is only accessible if the SPOT is connected to the host workstation by a USB cable.
- Deploy library – Downloads to the SPOT a fresh copy of the current library in the active SDK. This also removes any downloaded MIDlets and resets the SPOT.

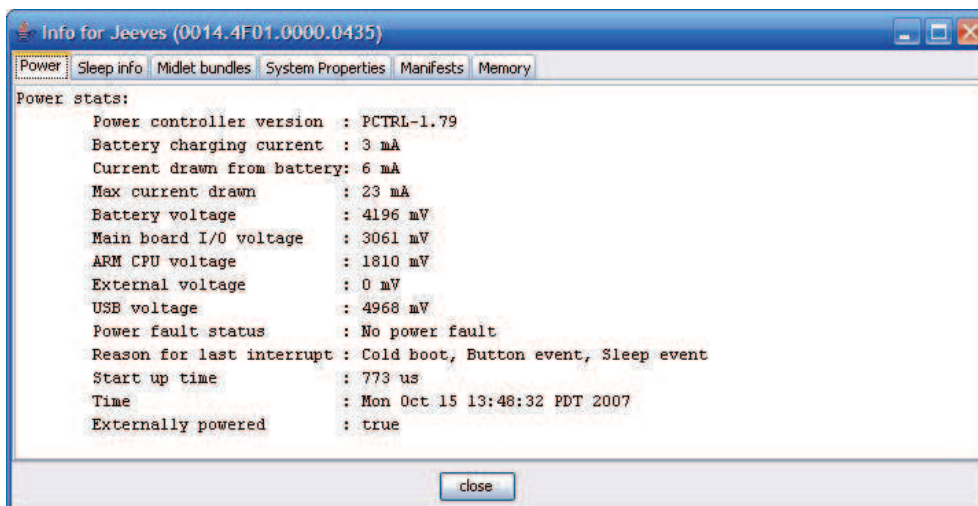
- Deploy application suite – Allows you to specify a project to be deployed to the selected Sun SPOT. The following dialog box displays:



The project directory will optionally be cleaned and compiled. The jar file is then deployed to the SPOT, after asking which MIDlet, if any, should be run in the main isolate.

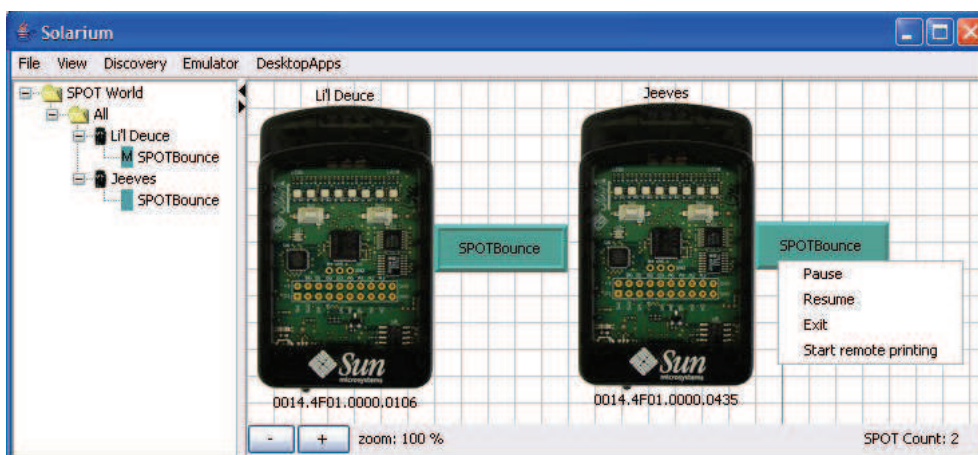


- Get Info – Opens a window containing six tabs. Each tab has information on a different aspect of the selected Sun SPOT.



- Run in Main Isolate – Allows you to specify a MIDlet to run in the main isolate. The system displays a menu of the MIDlets loaded on the SPOT and allows you to pick one. The selected application runs in the main isolate and begins execution the next time the SPOT is reset. Solarium displays the application name to the right of the SPOT in a sunken button display.
- Available Apps – Displays a menu of MIDlets that are available in the deployed jar file. Selecting one causes that MIDlet to start and it adds an application object to the Solarium display for that

SPOT. The application object can be moved around on the graphic display. Clicking the application object gives you a menu for that application.



In the illustration, the SPOTs are both running the Bounce demo. The SPOT on the left is running in the main isolate, as shown by M in the tree view. The SPOT on the right is running it as a child isolate.

The application menu choices are:

- Pause – Pauses execution of the selected application.
- Resume – Resumes execution of a paused application.
- Exit – Stops execution of the selected application.
- Start remote printing – Opens a dialog box that directs `System.out` for the selected application to a window. The window can be a new window or a pane with the Solarium window.

## Virtual Sun SPOTs

Solarium also has the ability to display and run applications in virtual Sun SPOTs.

To create a new virtual Sun SPOT in Solarium, pull down the Emulator menu from the main menu bar and select *New virtual SPOT*. Solarium creates a new virtual SPOT and places it in the upper left of the graphic display. It appears to have a blue rather than a smoke-colored plastic case. You can use the mouse to place the virtual SPOT any place in the display.

Virtual SPOTs can send and receive radio communications. Their radio communications with physical SPOTs are sent through a shared basestation.

A set of virtual SPOTs can be stored from Solarium. Select “Save virtual configuration” from the “Emulator” pull-down menu in the main menu bar. You are asked to specify a filename and Solarium stores the names of the virtual SPOTs, their IEEE number, their location in Solarium, the `jar` file that was loaded, if any, and the MIDlets that were running on the SPOTs. It also asks you to give a description of the configuration.

The virtual SPOTs in a stored configuration can be reloaded by selecting “Open virtual configuration” from the “Emulator” pull-down menu. Selecting “Display virtual configuration description” displays the description for the configuration most recently loaded.

For a tutorial introduction and more information about the emulator software, see the *Sun SPOT Emulator Tutorial*, and online HTML document.

---

**Note:** In previous releases there was a bug that caused the SPOT emulator to fail if the file path where the Sun SPOT SDK has been installed included a space character. Windows users encountered this problem because the standard installation location for Windows is C:\Program Files\. This bug was fixed in the 4.0 Blue release.

---

## Controlling SPOT Discovery

The Discovery pull-down menu in the main Solarium menu bar allows you to control the discovery process that Solarium uses to find the local SPOTs. The choices are:

- Discover Sun SPOTs – This choice sends a broadcast message, asking Sun SPOTs to identify themselves. If the broadcast message reaches a Sun SPOT running the OTA command server or if the Sun SPOT is connected to the host workstation via USB, the SPOT should be discovered.
- Discover one – This choice allows you to specify an IEEE number for a SPOT. Solarium attempts to find that particular SPOT.
- Set discovery time out – This choice specifies how long Solarium will wait for an answer to its broadcast message. It defaults to three seconds.
- Set discovery hop count – This choice specifies how many radio hops a discovery message may take before it is no longer forwarded from SPOT to SPOT. If using a shared basestation, it must be at least two, since the communication between the main process of the host workstation and the shared basestation counts as one hop. Therefore, at least two hops are required to reach any actual SPOTs. If you wanted Solarium to reach SPOTs that are not within radio range of the basestation, the hop count should be set to three or more.
- Set radio channel – Allows you to specify the radio channel used for discovery. Defaults to 26.  
**Note:** For a shared basestation, it is not possible to change the radio channel.
- Set radio PAN ID – Allows you to specify the PAN ID used by Solarium during discovery. Defaults to 3. Note: for a shared basestation, it is not possible to change the PAN ID.