Sun™ Small Programmable Object Technology (Sun SPOT) Release Notes
Release 3.0

Sun Labs
October 2007

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95045 U.S.A.
650 960-1300

Part Number 820-2123-10
Document Revision 2.1
October 2007

Please
Recycle

Sun™ Small Programmable Object Technology (Sun SPOT) Release Notes

These release notes accompany release 3.0 of the Sun SPOT system software.

New features in Release 3.0

Release 3.0 of the Sun SPOT software contains both new features and bug fixes. The most significant new features are:

- Libraries can be deployed over-the-air
- Applications can create exit hooks
- Java device drivers can access dedicated DMA buffers
- New driver for SD memory cards
- Improved over-the-air (OTA) command architecture
- Improved support for isolates
- Improved MIDlet lifecycle support
- Support for a coming revision (6) of SPOT main boards
- New command to select and start the basestation (`ant startbasestation`)
- Much improved multi-hop radio communication
- SPOT-side of the Telemetry Demo rewritten to use new utility helper classes
- Rework of the sensor library to add listener functionality
- Addition of a SPOT emulator to SPOTWorld
- SPOTWorld no longer requires Kami, but accomplishes the same goals with isolates and the OTA command processor
- SPOTWorld can manage SPOT clients that are more than one radio hop away from the basestation

- SPOTManager is now a JNLP application so the newest version will automatically be loaded from `sunspotworld.com` server

Sensor Library Details

- Added Listeners to `Switch`, `InputPin`, `ScalarInput`, `LightSensor`, `TemperatureSensor` & `Accelerometer`.
- Eliminated some unused classes like `AbstractADT7411RangeInput`, `ADT7411InternalTemperatureInput` & `ADT7411RangeInput`.
- Changed `ITemperatureInput` from the `com.sun.spot.sensorboard.io.ITemperatureInput` package to `com.sun.spot.sensorboard.peripheral.ITemperatureInput` (this change requires modifying existing code).
- Removed and renamed the various `EDemoBoard.bindXXX` methods so that all access to sensor board devices is through routines named `getXXX`.
- Made it so that `PinDescriptors` are only needed internally. SPOT application developers no longer need to use them.

  Here's a code snippet showing the new way to access IO pins and switches:

```
EDemoBoard  demo = EDemoBoard.getInstance();
ISwitch     sw1 = demo.getSwitches()[EDemoBoard.SW1];
IIOPin      pinD0 = demo.getIOPins()[EDemoBoard.D0];
IOutputPin  pinH0 = demo.getOutputPins()[EDemoBoard.H0];
IScalarInput pinA0 = demo.getScalarInputs()[EDemoBoard.A0];
IToneGenerator tone  = new ToneGenerator(demo.getIOPins()
[EDemoBoard.D3]);
```

Radio Details

The networking and radio stack has been improved in the following ways.:

- `LowPan` now allows routing manager to be changed on the fly (`LowPan.setRoutingManager`)
- `SingleHopRoutingManager` is provided as additional example of implementing an `IRoutingManager` for `LowPan`
- `NetManagementServer` and `IRoutingManager` now implement the `IService` interface for stopping/starting
- `LowPan` and AODV routing manager may have their behavior altered using the `IRoutingPolicyManager` and `spot.mesh.routing.enable` property
- `TraceRouteServer` has been renamed to `NetManagementServer`. The associated property names have also been changed.
- Instrumented `LowPan` layer with statistics (`LowPanStats` class)
- Improved default routing (AODV) with neighbor announcements

- Improved compliance with LowPan specification
- Fixed multiple bugs related to packet fragmentation and header parsing

Special Note on Broadcast Mode

Please note that while the maximum broadcast packet size is 1260 bytes of payload, an individual 802.15.4 radio packet only carries about 100 bytes of data. Because the list of recipients is unknown, broadcast mode is inherently unreliable. Broadcasts datagrams of one or two fragments are fairly reliable. Datagrams broken into three fragments (over 200 bytes of payload) are likely to experience some loss. Broadcasts datagrams broken into more than three fragments will almost certainly see some loss.

Important data should generally be unicast via radiograms or radiostreams. The inherent ACK/retry mechanism of radiograms generally insures either delivery or notification of failure. Likewise, radiostreams provide automatic fragmentation and an additional level of assurance that fragments are reassembled in the proper order.

Alternatively, if broadcast is required, the application should attempt to limit packet size so that each broadcast results in less than 3 fragments. Single packet broadcasts result in more data space as a fragmentation header is not required. Additionally, inserting a 20ms pause between the sending of broadcast packet will assist in allowing the receiver to keep up with packet reception.

Telemetry Demo Details

The SPOT-side of the Telemetry Demo has been rewritten to use a number of utility helper classes that you may find useful for your applications:

- `LocateService` to locate a remote service (on a host)
- `PacketReceiver` to receive commands from the host application and dispatch them to whatever classes have registered to handle that command
- `PacketTransmitter` handles sending reply packets back to the host
- `PeriodicTask` provides for running a task, such as taking samples, at a regular interval using the timer/counter hardware.
- Spotlet start of a framework for robust SPOT applications. Provides simple lifecycle management of application code by handling any fatal errors the application causes.

General Bug Fixes

In addition to these new features, approximately 250 bugs have been fixed.