# Why switch to DeviceKit-power?

**August 2008**

Richard Hughes, Software Engineer, Red Hat

# Executive Summary

- DeviceKit-power is a new system daemon that uses the DeviceKit framework and reports power related information to desktop clients and other system services. It allows the user to interact with power devices in the system, for instance suspending the system or disabling UPS outputs.
- DeviceKit-power is designed to compliment the other DeviceKit daemons such as DeviceKit-disks so that the power reporting functionality can be removed from HAL.
- **DeviceKit-power is nearly complete.**

# 5000ft Overview

- System activated system daemon
- Replaces HAL power management functionality
  - UPS devices
  - Laptop batteries and AC adapters
  - Wireless mice and keyboards
- Moves a lot of the profiling and statistics processing down from session clients such as gnome-power-manager.

# Why switch away from HAL

- HAL is a very successful, stable component that keeps having horrible code pushed into it to make hardware work
- Hal is unloved right now, as it's so complex
- Desktop clients need a thick wrapper to actually get sensible data from HAL
  - Multi-battery laptops?
  - Servers with more than one UPS

# Why switch away from HAL (II)

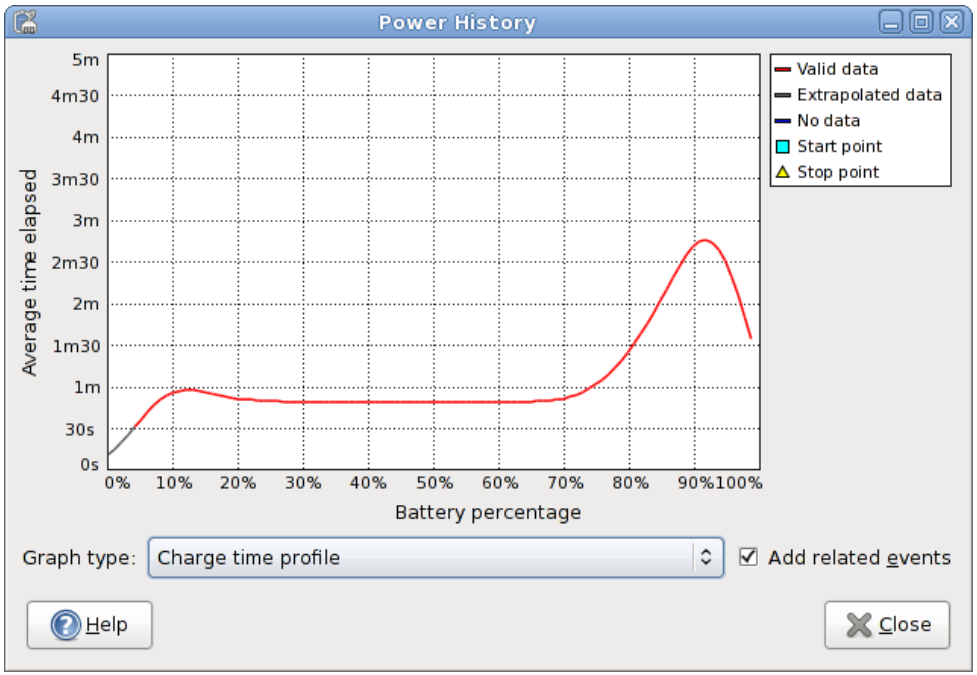
- HAL was written in the days before DBUS bindings
  - DBUS marshalling is clunky and huuuge.
- Having two device trees (udev -> HAL) means we have to essentially coldplug twice
  - Adding 3 seconds on every boot of my laptop
  - Adding minutes to servers with large arrays of disks

# Why switch away from HAL (III)

- Applications need to know the power state
  - e.g. does beagle start indexing, or are we on battery power?
  - HAL doesn't make this easy
    - Enumerate devices of {battery, ac_adapter} classes, get different properties on the devices, apply metric for broken hardware and then exclude non-supply devices.
  - Consequently most apps do this wrong, or just chicken out and read /proc/acpi/battery/*/state
  - **This needs to be much simpler**

# Move down from the session?

- gnome-power-manager does time profiling
  - Works great, but no data available when at GDM
  - Multiple clients collecting and storing the same data



# What can we do already?

```
[hughsie@hughsie-work DeviceKit-power]$ devkit-power --dump
Device: /devices/line_power_AC
  power supply:         no
  updated:              Tue Aug 19 13:32:52 2008 (3 seconds ago)
  line-power
    online:             yes
Device: /devices/battery_BAT0
  vendor:               SANYO
  model:                42T4504
  updated:              Tue Aug 19 13:32:52 2008 (3 seconds ago)
  battery
    rechargeable:       yes
    state:              fully-charged
    energy:             51.51 Wh
    energy-empty:       0 Wh
    energy-full:        51.55 Wh
    energy-full-design:  56.16 Wh
    energy-rate:        0 W
    percentage:         99.9224%
    capacity:           100%
    technology:         lithium-ion
Device: /devices/mouse_1_4x4
  native-path:          /sys/devices/pci0000:00/0000:00:1a.7/usb1/1-4/1-4.4
  vendor:               Logitech, Inc.
  model:                MX1000 Laser Mouse
  updated:              Tue Aug 19 13:32:53 2008 (2 seconds ago)
  mouse
    rechargeable:       yes
    state:              discharging
    percentage:         100%
```

# FDI merging made simple

- ATTR{idVendor}=="046d",
  ATTR{idProduct}=="c502",
  ENV{ID_PRODUCT}="Dual Receiver",
  ENV{DKP_BATTERY_TYPE}="mouse",
  ENV{DKP_CSR_DUAL}="1"

- **Using udev is simpler and quicker than XML hierarchal FDI files, even with a cache**

# Feature parity with HAL

- We can already do 80% of what HAL does
  - CSR, HID, power_supply
- Does not monitor obsolete kernel subsystems, for instance /dev/pmu or  /dev/apm

# Still left to do

- udev based sleep quirks
- Package up for Fedora
- Split the graphing stuff out from gnome-power-manager and push into DeviceKit-power-gnome
- Time profiling system wide (50% complete)
- Write migration document and start migrating applications
- Integration with *pm_qos* for low latency control?