System Integration and the GNOME desktop

GUADEC 2006, Vilanova, Spain

David Zeuthen - davidz@redhat.com

Topics

- HAL
- User Settings
- Enforcing Policy
- Privileges

- What's Next

What is HAL?

- Difficult to characterize
- Changed since the project was conceived in late 2003

What is HAL?

- “A library for enumerating the physical hardware in a system”

What is HAL?

- “A toolkit for enumerating the physical and logical devices in a system”

What is HAL?

- “A platform for discovering and configuring hardware”

What is HAL?

- Application developer POV:
  “A platform for discovering and configuring hardware”

- Kernel developer POV:
  “A user-space piece of code that crashes my driver and makes a lot of assumptions about it”

What is HAL?

- Application developer POV:
  “A platform for discovering and configuring hardware”

- Device driver developer POV:
  “A platform that provides applications with an OS and driver independent interface for device configuration”

- System integrator / OEM POV:
  “A platform for providing quirks and information about hardware”

HAL

- “Hardware Abstraction Layer”
- Service on the system message bus – org.freedesktop.Hal
- Device Objects for devices and capabilities of system
- Simple user API
- Extensible
- Merges information from several sources
- Designed with portability in mind

HAL

- D-BUS Interfaces
  - org.freedesktop.Hal.Manager - GetAllDevices(), FindDeviceByCapability()
  - org.freedesktop.Hal.Device - GetProperty(), Lock(), Unlock(), ...
- Device Specific D-BUS Interfaces
  - org.freedesktop.Hal.Device.Volume - Mount(), Format(), ...
  - org.freedesktop.Hal.Device.Volume.Crypto - Setup(string passphrase)
  - org.freedesktop.Hal.Device.SystemPowerManagement - Suspend(), Hibernate(), ...

Demo

- hal-device-manager
- LUKS stuff

HAL

- Device Objects
  - Properties (key/value pairs)
  - Capabilities
  - Signals / Conditions
  - Specified in the “HAL spec”
- Device Information Files (.fdi files)
  - Mechanism for associating information with devices
  - Used for: DAP (e.g. iPod); Card Readers; quirks for power management
  - Demo: .fdi files

HAL, design goals

- Designed with portability in mind
  - Linux 2.6: widely deployed
  - Solaris: In progress
  - FreeBSD: In progress
- Designed to be used by multiple desktops
  - GNOME: Early adopter
  - KDE: Some KDE 3.x code, Solid project for KDE 4.x
- Supposed to be easy to use
- Hides complexity of native kernel interfaces
- Mechanism, not policy

GNOME apps using HAL

- gnome-volume-manager
- gnome-power-manager
- gnome-vfs
- nautilus-cd-burner
- gnome-mount
- gstreamer hal sink

Some History

- Summer 2003: Havoc Pennington writes a paper called “Making Hardware Just Work”
- Nov 2003: First CVS commit
- Spring 2004: Robert Love and Joe Shaw starts writing gnome-volume-manager and Project Utopia is born
- April 2004: Kay Sievers starts work on the volume_id library
- Oct 2004: FC3 to ship with HAL 0.2.97
- Jan 2005: HAL gains functionality to report battery status (support for ACPI, PMU, APM and shortly after UPS'es). Richard Hughes starts work on gnome-power-manager
- July 2005: HAL gains functionality for defining device specific D-BUS interfaces. This is quickly used for LUKS integration and soon GNOME supports passworded media via both pmount and gnome-mount
- January 2006: fstab-sync, the tool used for updating /etc/fstab, is removed in favor of reading settings from the desktop session via e.g. gnome-mount. Sjoerd Simons contributes a patch so most of HAL runs unprivileged.
- March 2006: PolicyKit is split out as an independent project

User Settings

- *rm -rf /etc && rpm -e emacs vi gnome-terminal*

- Plug and Play should just work
  - Some times user intervention is required for configuration
- Philosophy
  - HAL is only a mechanism
  - All policy and settings need to come from the desktop session

User Settings

- Enforcing policy from the desktop session
  - gnome-volume-manager, gnome-power-manager
- Reads settings from gconf
  - Lock down, Defaults etc.
- Invoke methods on HAL to enforce policy

Policy daemons in the desktop session

- Runs in the desktop session
  - Attaches to the session bus to provide service to applications
  - Demo: gnome-power-manager
- Issues
  - Policy is enforced only when someone is logged in
  - Fast-user switching, Multi seat : important, but hard

Policy daemons in system context

- Runs in the system context
  - One per system
  - Not attached to any desktop session
  - Usually uses home-brewn IPC or system message bus
  - Usually reads settings from /etc
  - “Traditional” UNIX way

Problems with system daemons

- Desktop Integration is weak
  - UI config tool normally writes file in /etc
  - Most things are inherently per-session (power management daemon needs to communite with e.g. screen saver daemon)
  - Typically written by people with little or no clue about desktop and usability

    Example: grepping process list for mplayer

Problems with system daemons

- Desktop integration leads to writing a proxy for the desktop session so the system-wide part will, in effect, be reduced to a mere mechanism
- HAL is already a mechanism for interfacing with hardware and arbitrating access

- Just use HAL

When no one is logged in...

- Just run the desktop session policy daemons under gdm as user 'nobody'
  - g-p-m, nm-applet, g-v-m, g-s
  - Notification area in gdm?
  - Possibly headless operation for e.g. run level 3
  - Instances running as 'nobody' will read default setting from gconf
    - gnome-power-properties could have a “Make these settings system-wide” button or we could be even smarter about it
- However, we need infrastructure to actually do this...

Sessions & consoles

- There is still a lot of work left.. for example
  - Multi-seat and Sun Ray style desktops
  - Fast User Switching
- To do this right, I think we need some system service that
  - Enumerates the consoles in the system
  - Enumerates the sessions on each console
  - Tells what session is active on what console
- And we need integration with D-BUS
  - System bus needs to determine, in a secure way, what session the calling application stems from
    - Mechanisms such as HAL would refuse service to e.g. g-p-m unless it's from a session on an active console
- Said service would take care of starting / stopping instances of policy sessions daemons under gdm.
  - Perhaps the service would also be an initscript replacement

Privileges

- Desktop session is unprivileged
- HAL (essentially) runs privileged
- Unprivileged desktop apps can make HAL perform privileged operations

Privileges

- Up until now OS vendors have restricted most operations on HAL using either
  - at_console (Red Hat, SUSE, others); access restricted to console user
  - group membership (Debian, Ubuntu, others)
- Same mechanism used for access to files in /dev

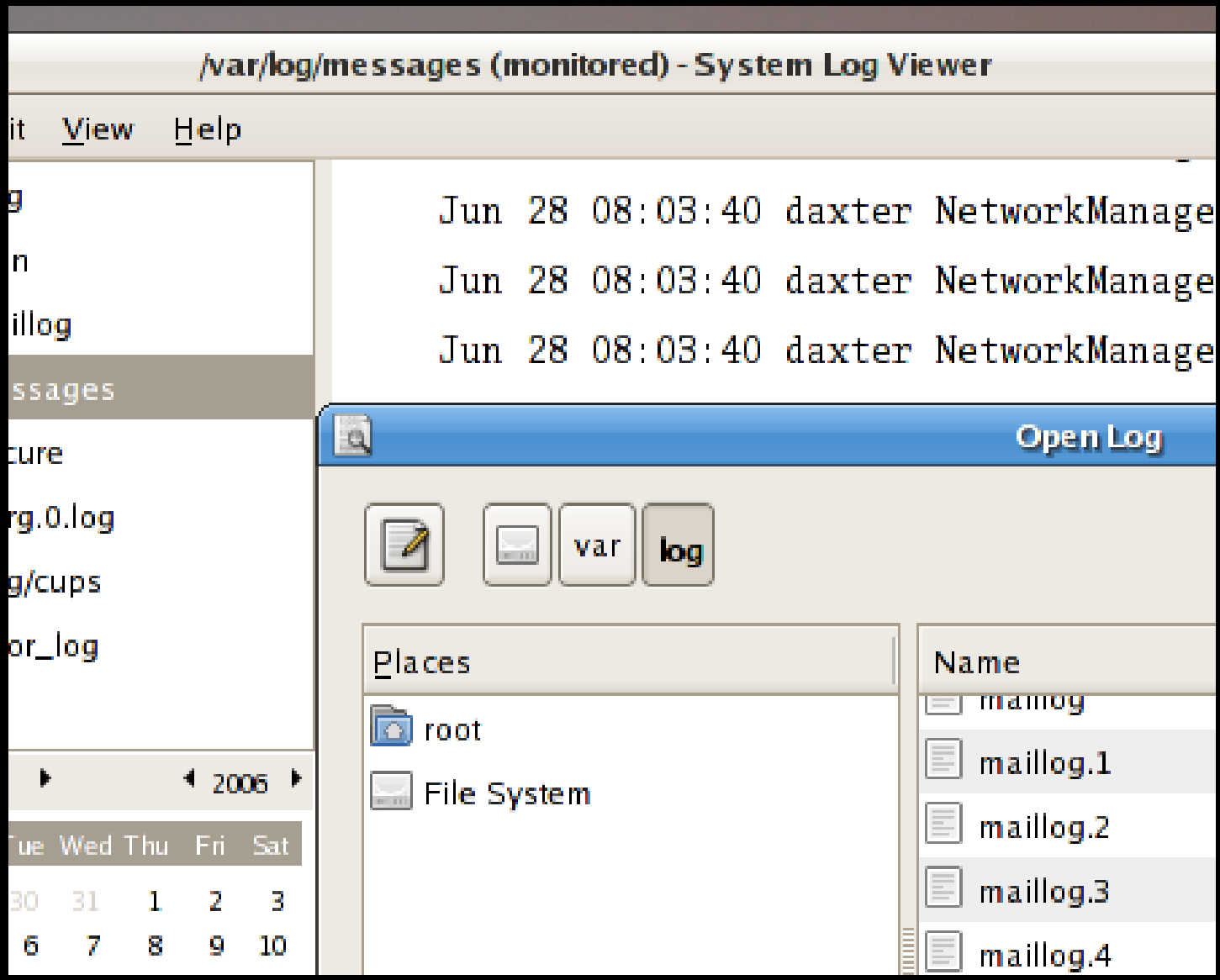
Privileges

- Both at_console and group membership are insufficient; we need a more fine grained mechanism
- Problem became more evident as HAL gained more features
  - Power Management
  - Mounting NTFS and HFS+ partitions on dual boot systems
  - Decided to fix this before adding Format() and PartitionDisk() to HAL as these really require a trusted path
- What mechanisms are used?

Privileges

- Most distributions use a “s u helper”
- X11 apps running as uid 0, yay!
- Python, GTK+ and more... Millions of LOC, including image loaders, run privileged
- Typically OS specific
- su helpers leads to laziness

What's wrong with this picture?



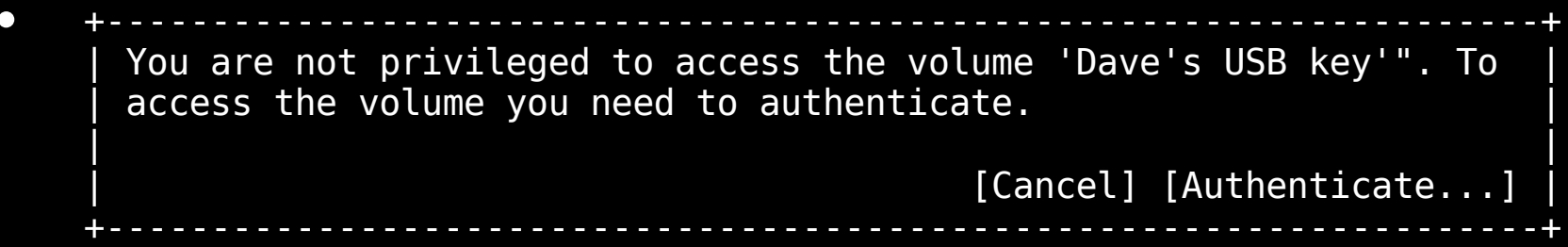
su helpers

- Basically broken by design, yet it seems to be in vogue
  - xdg-su from the Portland project
  - most system configuration tools
- If you need an su helper you most likely have design problems other places in the stack
- PolicyKit attempts to solve some of these problems

PolicyKit

- Example
  - Assume the user is not privileged to access removable storage devices

PolicyKit

- 
You are not privileged to access the volume 'Dave's USB key'". To access the volume you need to authenticate.
[Cancel] [Authenticate...]

PolicyKit

- 
User: [Dave_____]
Password: [_____]
Would you also like to automatically allow
(*) This user to mount 'Dave's USB key'
( ) Any user to mount 'Dave's USB key'
( ) This user to mount a removable storage device
( ) Any user to mount a removable storage device
[<- Drives and Media Preferences]
[Mount]

PolicyKit

- Why?
  - Some enterprises disable USB ports
  - Lock down, Kiosk
  - Malicious users
  - Sarbanes-Oxley
- May get in the way of the user getting work done
- “One time pain” dialogs
- OS independent; apps should be able to rely on it

- Need input from usability people and sys admins

PolicyKit

- Applications need to follow an architectural pattern
  - All privileged operations are carried out by a privileged helper on the system message bus, e.g. D-BUS service com.redhat.MyApp.Helper
  - Applications provide *privilege descriptors*, e.g. the file com-redhat-myapp-foo.privilege for the privilege Foo
  - When an unprivileged desktop application needs to do something privileged, it asks the helper, e.g. calls into com.redhat.MyApp.Helper

PolicyKit

- /etc/PolicyKit/privilege.d/com-redhat-myapp-foo.privilege

```
[Privilege]
RequiredPrivileges=desktop-console
SufficientPrivileges=
Allow=uid:500
Deny=
CanObtain=True
CanGrant=True
ObtainRequireRoot=True
```

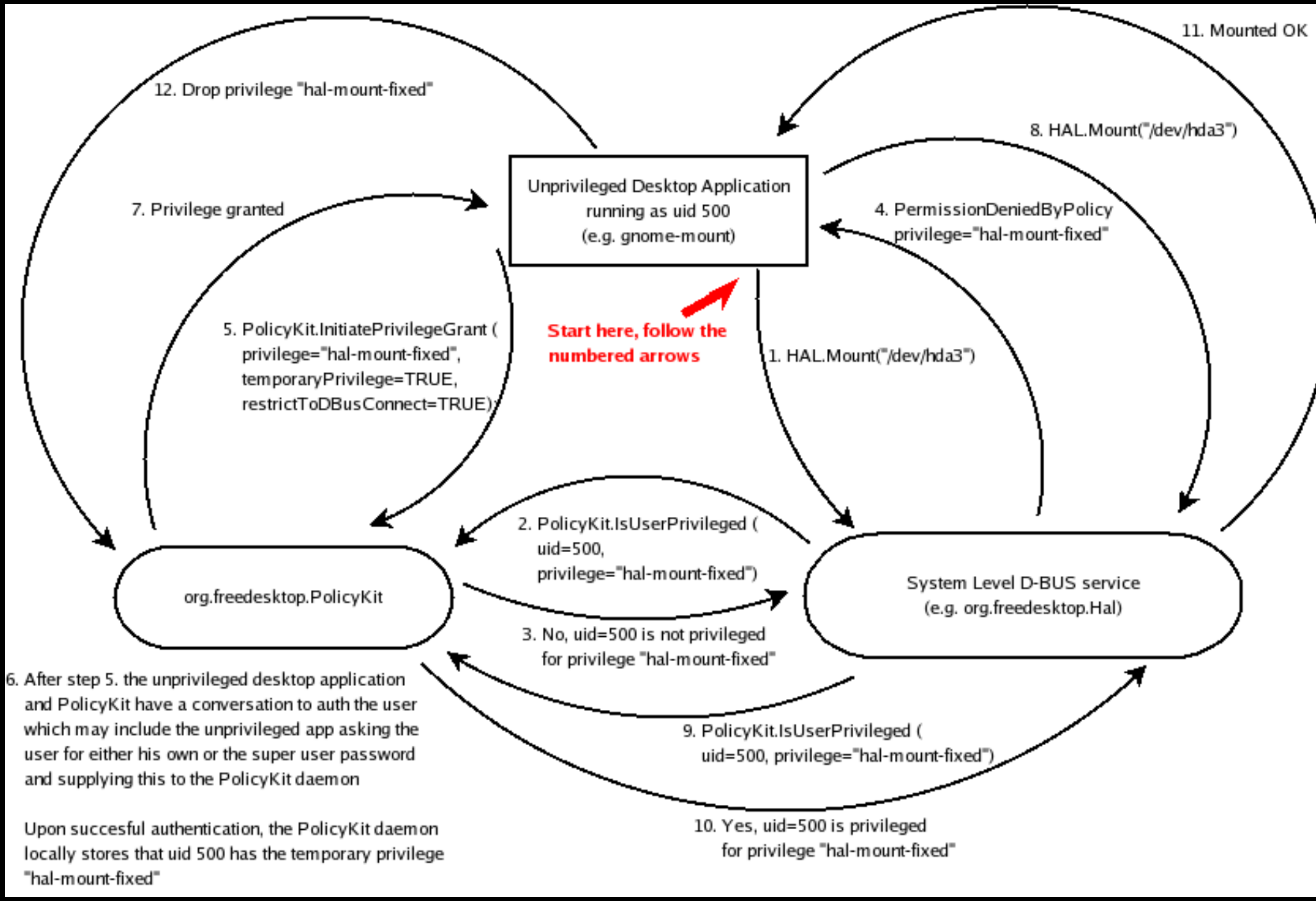
PolicyKit

- The helper asks the PolicyKit daemon if the calling user is privileged
- If the calling user is privileged the helper carries out the operation and we are done
- If the calling user is not privileged, the helper throws an exception, e.g. com.redhat.MyApp.Helper.NotPrivileged with the name of the privilege in the error detail, e.g. com-redhat-myapp-foo
- The application may now ask the PolicyKit daemon if the user can authenticate to gain the privilege com-redhat-myapp-foo

PolicyKit

- A privilege can specify whether a user can obtain it by auth and if so, whether he needs to auth as himself or the super user
- The temporary privilege can be restricted to the unique D-BUS system bus connection name of the caller. This ensures only the calling app may benefit from this privilege
- If the user succeeds in obtaining the privilege, the app may now call into the helper again

A picture says more...



PolicyKit

- Authentication is PAM wrapped in D-BUS
- Privileges can be granted on a per-resource basis
- Privilege descriptors are shipped with applications

PolicyKit

- User can optionally grant a privilege to himself and others if he successfully obtains it via authentication
- Automatic granting / revoking of desktop-console privilege when a user logs in / logs out
- System can be completely locked down by editing privilege files

PolicyKit

- Demo: command line utils

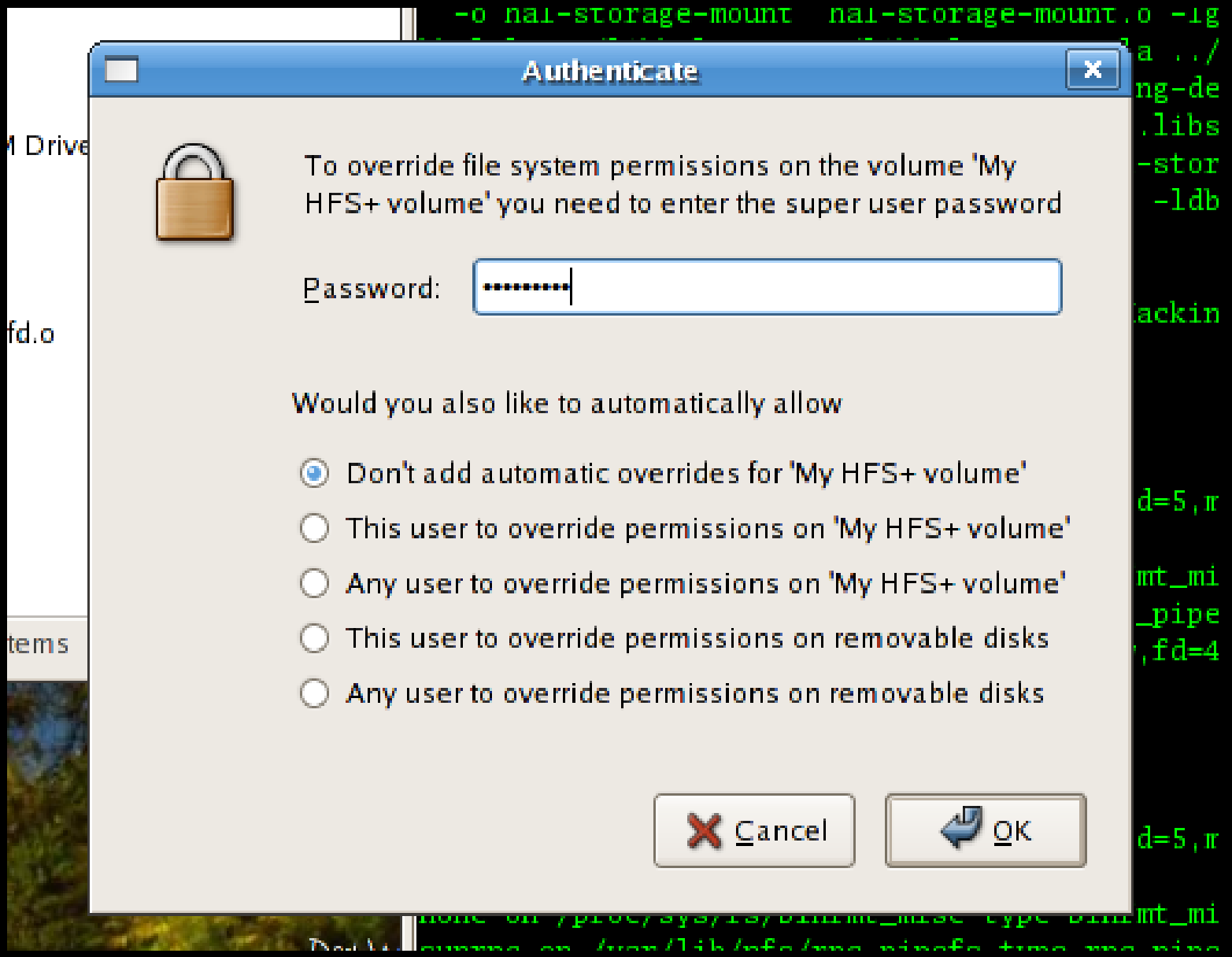
PolicyKit

- HAL 0.5.8 will depend on PolicyKit 0.2
  - All device specific operations on HAL will require the user to possess a privilege
  - gnome-mount will use PolicyKit for authentication if the user isn't privileged
- PolicyKit is mostly done

- Awaiting feedback from the community
- Make apps use PolicyKit

PolicyKit and gnome-mount

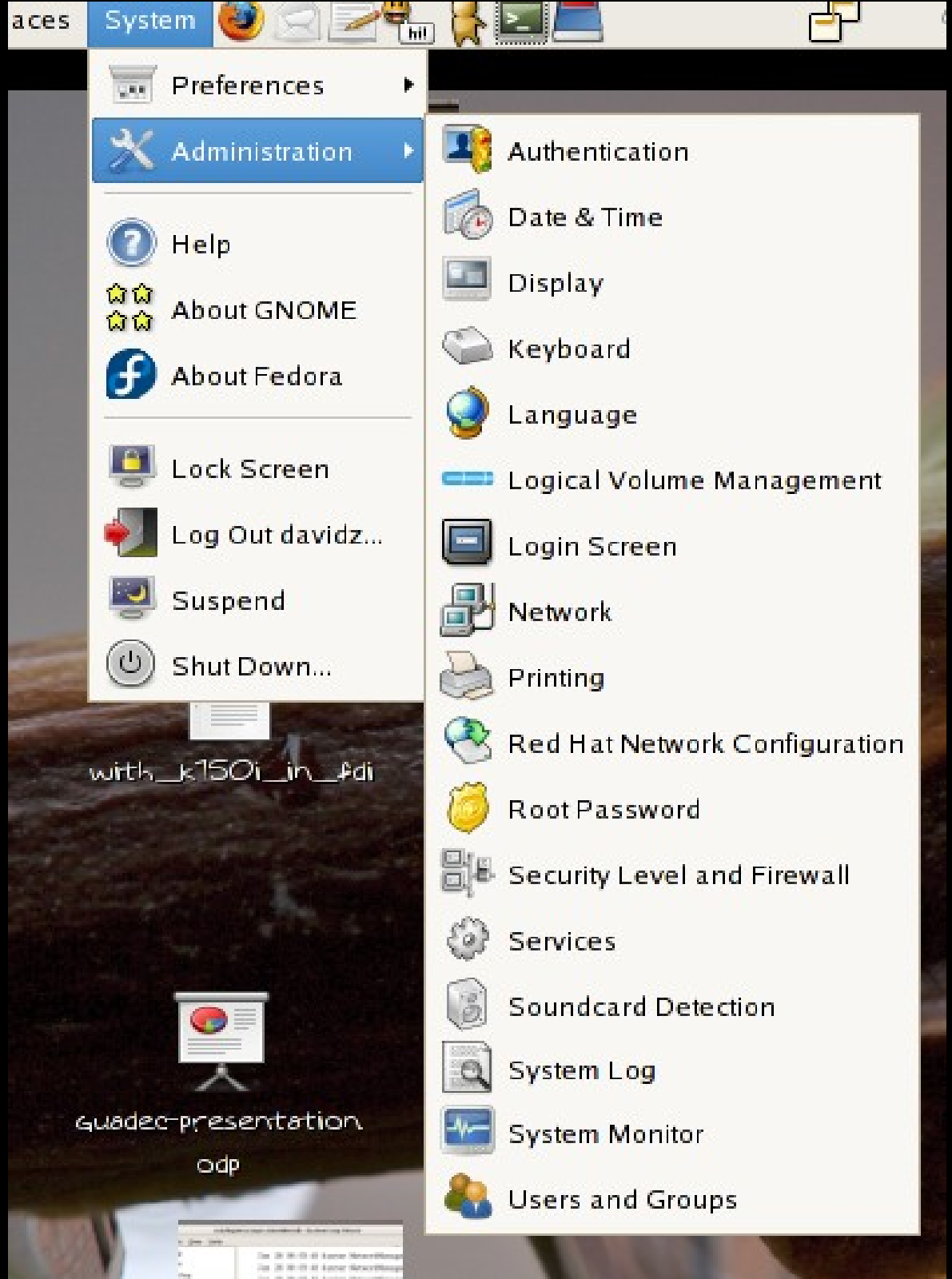
PolicyKit and gnome-mount



PolicyKit

- Where can PolicyKit be used?

- Today the root password is used for
  - Package Management
  - Add / Remove users
  - Setting date / time / timezone
  - Login screen
  - Much more...


aces
System
Preferences
Administration
Help
About GNOME
About Fedora
Lock Screen
Log Out davidz...
Suspend
Shut Down...
Authentication
Date & Time
Display
Keyboard
Language
Logical Volume Management
Login Screen
Network
Printing
Red Hat Network Configuration
Root Password
Security Level and Firewall
Services
Soundcard Detection
System Log
System Monitor
Users and Groups
with_k150i_in_fdi
guadec-presentation
odp

PolicyKit example

- gnome-user-share is the greatest thing since sliced bread...
- ... but it requires me to turn off the firewall to be useful
- In Fedora we could just allow any app to listen on a high port or a range of ports
- However, any random junk app could then turn your system into a zombie
- SELinux can actually help out here

PolicyKit

- SELinux integration
  - Label /usr/libexec/gnome-user-share as running in the *trusted-desktop-app* security context
  - Write helper app that claims the service org.freedesktop.PolicyKit.Utils.PunchHoleInFirewall (or gnome-system-tools?)
  - Introduce Allow=selinux:trusted-desktop-app for the punch-hole-in-firewall.privilege
  - Now g-u-s don’t need to prompt for auth to open a hole in the firewall for file sharing
  - Banshee, Rhythmbox, other apps could benefit

PolicyKit

- Ultimately make userhelper and su helpers go away – they're broken by design
- Minimize the amount of system tools shipped by distros by integrating GNOME and the system
- We all benefit from doing things upstream

- System Integration is not sexy, but it's still important

PolicyKit

- Makes it easy to integrate privileged operations into existing code
- Could read privileges from LDAP instead of local files
- GNOME could share privileged back-end code with e.g. KDE

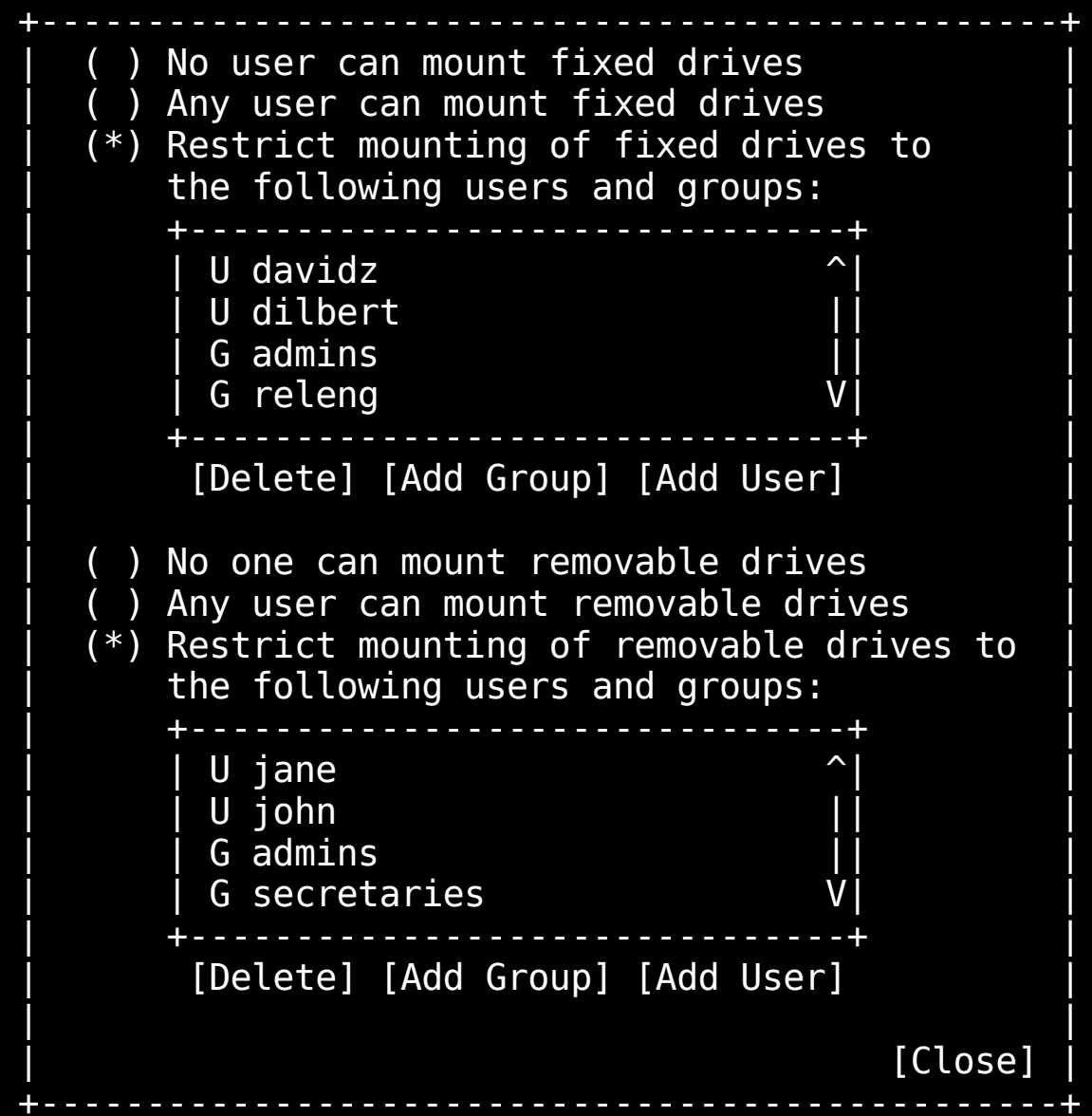
  “set time and date”, “set time zone”, “punch hole in firewall”, “add user”, ...

  PolicyKit-xdgutils freedesktop.org project?
- gnome-system-tools is looking at using PolicyKit

PolicyKit-gnome

- GNOME bits for using PolicyKit
  - Actually not written yet
- Authentication Dialogs
  - Code reuse
  - Should respect a gconf key so the system can be configured to never prompt
  - Could have a “File ticket with IS” button if user is not privileged?
- Privilege editor
  - sys admin tool akin to gconf-editor

PolicyKit-gnome

- 

What's next?

- Need to PolicyKit and HAL releases RSN
- Planned feature additions
  - Bluetooth support (mjg59)
  - Format() / PartitionDisk() / RenameVolume()
  - Better UPS support via nut integration
  - Power Management quirks
  - ALSA userspace driver integration
  - FreeBSD, Solaris support
- Try to make GNOME “just work” with hardware

- Think about HAL 1.0

HAL 1.0?

- No one really done this kind of thing before
  - Is the model right?
- What device support should be in HAL?
  - Can HAL be useful in dealing with X.org configuration?
- What features are needed?
  - GNOME
  - OS vendors
  - Hardware vendors

Hard Problems

- Linux is a moving target
  - Not to mention that Solaris and FreeBSD
- Driver models and device frameworks vary a lot
  - User space code is usually not, uhm, good at exposing data to other things such as HAL
    - X.org, Bluetooth, ALSA, ...
  - Linux 2.6's sysfs interface
- Cultural Issues
  - Fear of change
  - UNIX mentality
  - OS distributors