3.3 Release Notes: Preferences

This file contains release notes for the 3.3 and 3.0 releases of Preferences. Items specific to the 3.3 release are listed first, and the other notes follow.

Notes Specific to Release 3.3

New Features

The following new features have been added to Preferences since Release 3.1:

- Password changing UI has been improved.
- Preferences modules that aren't appropriate for the current system don't load.
- Preferences can now automatically launch a program for a module at startup, without the module having been loaded.
- It's now possible to drag the default menu location partially off of the screen.
- You can remove the menu from the screen altogether by command-clicking on the mini-menu.

- Preferences now has better control over sound preferences in a new sound module.

- You can now set your default paper size in the Preferences localization module.

- There's a new module to control loginwindow preferences. (this only appears for users logged in as **root**).

- There's a new module for controlling and viewing power management state for Intel machines that support the APM bios. This only loads if your machine has power management features. When the module loads, and extra icon clock is made available that shows the current battery life of your machine.

- Preferences now gives feedback when loading modules.

- Preferences now deals with duplicate preferences modules more intelligently. It will use the first one of the same name that it find in directories in this order: ~/Library/Preferences, /LocalLibrary/Preferences, /NextLibrary/Preferences and /NextApps/Preferences.app.

- Preferences now has API to allow you to define some attributes of a module at startup time. They can be defined via a file named "Info" in the module dir (or in the .lproj's of the module dir). Here's what the Power Management Info file looks like:

```
{
"Long Name" = "Power Management Preferences";
"Architectures" = ("i386");
"Test Program" = "apm_present";
}
```

  The "Long Name" key is the name of the module as it appears in the title bar. (You no longer need to put your own name in somewhere in the module nib).

  The "Architectures" is a way to quickly eliminate a module from being loaded. If the current

architecture isn't in the listed architectures, the module won't be loaded.

The "Test Program" is a program that should be run at startup to determine if this module is appropriate for this user on this machine. It should return 0 if the module should be loaded, and any other value if not. If the path to the program doesn't begin with a '/', it is assumed to exist in the module in question. Otherwise it is treated as an absolute path.

One final possibility is the "Bundle Daemon" field. This allows you to specify a program that is always run when Preferences is launched (if the module would passes the architecture and test program tests). Again, if it doesn't have a leading '/' it is assumed that the program is in the module's directory. NOTE: the daemon is forked off in the background. If Preferences is quit, the daemon will keep running (and doesn't get notified that Preferences has quit). This could lead to multiple instances of your daemon running simultaneously. It's up to you to make sure that this doesn't happen.

Notes Specific to Release 3.0

These notes were included with the Release 3.0 version of Preferences.

New Features

The following new features have been added to Preferences since Release 2.0.

· Display PreferencesÐThe color of the display's background can be set.

· Localization PreferencesÐPreferences contains a new layout for 3.0. The Localization layout provides control over all aspects of system localization. The keyboard mapping, system language, and measurement units can be specified.

- System PreferencesÐSeveral new font options have been added.

- Dynamic layout loadingÐSupport has been added to allow loading of layouts into the Preferences application. This feature is described briefly below and in more detail in **/NextLibrary/Documentation/NextDev/GeneralRef/15_Preferences.**

  Layouts are added by creating a directory for the new layout in **/NextLibrary/Preferences**, **/LocalLibrary/Preferences**, or **~/Library/Preferences**, or by opening a **.preferences** file package from Workspace. The name of the file package should correspond to the name of the layout. The file package must contain at least two files, a tiff image, and an object module both with the same name as the directory. Additionally an InterfaceBuilder file may be contained in a **.lproj** directory. For an example look at the directory named **Template.bproj** in **/NextApps/Preferences**. The directory **/NextApps/Preferences.app/Template** can be renamed to "Template.preferences" and opened from Workspace. The file package named "Template.preferences" contains the following files.

```
/NextApps/Preferences/Template.preferences/Template
/NextApps/Preferences/Template.preferences/Template.tiff
/NextApps/Preferences/Template.preferences/English.lproj/Template.nib
```

  The TIFF image file is the icon that will appear in the scrolling list of icons at the top of the Preferences window.

  The object module should contain a subclass of the object Layout and should be named after the layout. The header for this object is located in **apps/Preferences.h**. This class will be instantiated when the layout is loaded. If your layout contains more than one mach object module then you should create a single object file using ld. When creating the Mach-O you should use the "-r" option of **ld** to retain relocation symbols. For example:

```
ld -r -o Template.o AnotherClass.o funcs.o Template.o
```

This would create a single Mach-O called **Template.o** that contained three individual Mach-O's, **Template.o**, **AnotherClass.o**, and **funcs.o**. Note that the subclass of Layout must be specified as the last argument on the command line.