Maemo Packaging Policy
v0.23

Authors:	The Nokia maemo team
Scope:	Packaging of open and proprietary components in IT OS 2008
Status:	Draft
Wiki:	TODO

May 8, 2008

Legal notice

Disclaimer

License

Contents

1 Introduction and processes

1.1 Target audience

The target audience for this document are developers wanting either to develop for or port packages to the maemo™ platform. The contents of these packages can be either open or proprietary, the rules are the same for both.

1.2 Purpose

This document describes the general rules agreed for creating maemo packages and some examples on how to conform to them. Following them makes package integration easier (non-compliance is a blocker or critical bug) and it will help maintaining the packages in the long run. Conforming packages should also install and behave better in the users' devices.

For related documents, see:

[Quick Start Guide] An overview of the maemo platform

[Debian Policy] Policy for Debian, the upstream distribution of maemo

[Maemo Application Packaging] A guide for making packages installable in the Application Manager

[Maemo Coding Guidelines] Guidelines for creating the software that is packaged

[Maemo Policy wiki] Wiki for information related to the policy process:

- Lists of current and rejected policy additions and update proposals
- Details of differences between maemo and Debian distributions
- Common packaging mistakes

1.3 New versions of this document

This manual is distributed as the `maemo-policy` package in the maemo repositories.

The `maemo-policy` package also includes the file `upgrading-checklist.txt` which indicates policy changes between versions of this document.

Note that policy follows the tools, not the other way round. The process is following:

1. Change tools (dpkg, apt etc.)
2. Update policy to match the changed tools

1.4 Process for updating the policy

The maemo packaging policy is maintained by the Nokia maemo team. Proposals for policy changes and additions shall be documented in the maemo packaging policy Wiki and discussed on the `maemo-developers@maemo.org` mailing list.

Proposals can be accepted into the policy after a certain consensus is established. Rejected proposals are listed separately in the Wiki.

E-mails concerning the policy should include the word "policy" in the subject. For errors in this document and any issues that need to be tracked, a bug can also be filed against the `maemo-policy` package in the maemo Bugzilla.

TODO: This is the intended update process. At the time of writing this version of the document, the policy wiki and bugzilla component do not exist yet. Also, a separate policy mailing list could be set up if necessary.

1.5 Relation to Debian Policy

This document is modeled after the Debian Policy manual version 3.7.2.2 (see [Debian Policy]). It lists only the differences between the Debian policy and the maemo packaging policy, stating reasons for the differences and including some other notes on subjects discussed in the Debian policy. The main sections here have the same numbering as the corresponding sections of the Debian Policy manual. Subsections do not necessarily have the same numbering but have (roughly) the same names as in the Debian Policy manual.

In general maemo packages should follow the Debian policy. However, there are some items where maemo:

- Is more strict because it's an embedded distribution, aimed at consumer products with more limited resources and also e.g. different legal requirements
- Is more relaxed because the target (Nokia) devices are used only through a specific UI and do not have multiple simultaneous users
- Differs from Debian because each distribution has its own objectives, maintainers and infrastructure.

The `lintian` command (from the `lintian` package) on Debian (and derived) Linux distributions can be used to check packages against the Debian Policy and good packaging practices[1]. Currently maemo doesn't provide a utility to check conformance to the additional policies stated in this document.

TODO: Check this policy against changes or additions in Debian Policy 3.7.3.0 which was released recently

[1] You can use the `-I` option for additional checks, but not all of them are strictly necessary to adhere to. You can also try Debian `linda` tool from the `linda` package.

1.6 Conformance and handling packaging differences

The policy does not require that maemo would have the same packages available (or the same versions of packages) as Debian.

However, when the same software or functionality is available in both maemo and Debian, it should be packaged similarly when possible, to improve compatibility and to reduce maintenance and porting overhead.

This section describes the processes and general guidelines for ensuring compatibility and policy conformance.

1.6.1 Policy violations

If a package doesn't conform to the maemo packaging policy, or to Debian Policy in issues not covered by the maemo packaging policy, one of the following actions MUST be taken for the package to be acceptable for maemo:

- The package is fixed
- The maemo packaging policy is updated to accept this exception
- The policy change is propagated to upstream (to Debian Policy)

Bugs filed for policy violations should include the keyword "policy".

1.6.2 Differences between corresponding maemo and Debian packages

When a package in maemo conforms to the policy, but the packaging differs from Debian in a way that could cause incompatibilities (and the difference is not because of something stated in this policy document), a bug with a "packaging" keyword SHOULD be filed.

Here incompatibilities means anything that would cause problems using the package together with other packages, including e.g. the location of files, package naming, whether the package provides what is expected from it, that it doesn't include conflicting files without declaring a conflict, etc.

If the bug is valid, the packaging SHOULD be changed to resolve the issue. (If the issue is minor or can't be fixed now, that should only affect the severity or schedule, and the bug should remain open.) Reasons for WONTFIX could be e.g.:

- the constraints of the device environment,
- that changing the current packaging or behavior would break some specific product feature, or
- that the packaging change has already been promoted upstream or Debian is otherwise considering a similar packaging change.

Currently maemo corresponds mostly to the Debian 4.0 ("Etch") release, so packaging SHOULD be modeled after that or a newer Debian version.

TODO: When the maemo policy wiki has been created, the process for documenting differences there should be defined

1.6.3 Maemo packages that are not yet in Debian

Open source maemo software SHOULD include packaging files (the `debian/` directory) in the project's public version control, even if the actual packages wouldn't yet be available in the public maemo repositories.

This helps to make sure that if Debian or some Debian derivative distribution decides to include that software, they have a model for how to package it, and unnecessary differences can be avoided. It also allows other distributions' maintainers to review and suggest improvements to the packaging.

When packaging new software, please note the correct use of the version string: If the package is native to maemo (does not have an upstream), a revision part should not be used; otherwise see the policy about packaging upstream sources in section 3.2.

1.7 Terminology

BTS Bug Tracking System. The public BTS for maemo is at `https://bugs.maemo.org/`

Upstream sources The original, unpackaged source code for some software

Upstream distribution The distribution where packaged sources are taken from. The upstream distribution for maemo is Debian. In some cases maemo also packages upstream sources directly instead of using or modifying Debian source packages.

Upstream package A package in the upstream distribution.

The uppercase words "MUST", "SHOULD" and "MAY" are used to distinguish the significance of the various guidelines in this policy document. Packages that do not conform to the guidelines denoted by "MUST" will generally not be considered acceptable for maemo. Non-conformance with guidelines denoted by "SHOULD" will generally be considered a bug, but will not necessarily render a package unsuitable for distribution.

2 The maemo archives

2.1 Categories

Debian's "main", "contrib" and "non-free" categories are not used as such in maemo packages. The directory structure of repositories may follow this model or something else, but that is not relevant to packaging and therefore not in the current scope of this document.

2.2 Sections

Packages are grouped into *sections* as in Debian, but SHOULD NOT specify a category in the *segment* part. (However it is not a bug if a package taken from Debian and made available in maemo retains its "contrib" or "non-free" segment.)

Instead maemo defines a `user` segment for controlling visibility in the Application Manager. Packages that are intended to be visible in the Application Manager MUST belong to the `user` segment, and packages that are not intended to be visible (such as libraries and other dependencies) MUST NOT belong to that segment.[2]

The section of packages in the `user` segment SHOULD be one of the following: *accessories, communication, games, multimedia, office, other, programming, support, themes, tools*.[3] The sections in this list will appear correctly localized in the Application Manager.

As an example, to put a package into the Office section and make it visible to the user, the value of the `Section` field in the `debian/control` file should be `user/office`.

Packages not in the `user` segment SHOULD use the sections listed in the Debian Policy (`http://www.debian.org/doc/debian-policy/ch-archive.html#s-subsections`).

2.3 Priorities

Maemo doesn't currently use package priorities for anything. For future compatibility, please conform to the Debian policy.

3 Binary packages

3.1 Package naming

Any use of the word "maemo" is subject to the trademark policies and guidelines set forth at maemo.org (see [Maemo Trademark Policy]).

In general, a package may use the word "maemo" as a part of its name *only* if the package is *both*:

- Created or approved in writing by Nokia Corporation, and
- Intended to be used *only* in the maemo environment (either within Scratchbox or devices)

Packages that were published with the word "maemo" in the package name before 11th May 2007 (when the trademark policy was published) may continue using that name; see the Trademark FAQ available at maemo.org.

3.2 The version of a package

If an upstream package is re-packaged or otherwise modified for maemo, a maemo revision MUST be appended to the upstream revision. The resulting revision string SHOULD have the form "*RidentifierX*" where:

[2] See also [Maemo Application Packaging]

[3] Note these sections differ from Debian, and therefore the `lintian` utility will complain about them.

- *R* is the upstream revision
- If the modification is intended (only) for the maemo platform, *identifier* SHOULD be "maemo".[4] In other cases some suitable other identifier should be used.
- *X* is a monotonically increasing number.

If upstream sources are packaged for maemo directly, when an upstream package for the corresponding version does not exist but is likely to become available later, a maemo revision prefixed with zero SHOULD be used (resulting in "0*identifierX*").

These rules ensure that when a new upstream package is released, its revision will be higher and it will be considered as an upgrade from the older maemo package (the future upstream package may or may not require modification to fit into maemo, but that should not be a consideration when choosing revision numbers). This is similar to how e.g. Ubuntu deals with Debian packages.[5]

To verify that the new version is higher than the previous maemo version, but lower than the next upstream version would be, compare the version numbers with:
`dpkg --compare-versions` *oldversion* `lt` *newversion* `&& echo true`

Otherwise the rules for package versions are the same as in Debian. (Please note the proper format of the version string: For native packages, i.e. software which doesn't have an upstream, use a version number without a revision part.)

3.2.1 Example of modifying an upstream package

- Version of upstream sources: 0.10
- Version of upstream package: 0.10-4
- Version of modified package in maemo: 0.10-4maemo1

3.2.2 Example of packaging sources not yet in the upstream distribution

- Version of upstream sources: 0.7
- Latest package in upstream distribution: 0.6-3
- Package version in maemo: 0.7-0maemo1

3.3 Package maintenance

TODO: Currently maemo does not have a process for orphaned packages.

3.4 Dependencies

See the discussion about dependencies to Essential packages in section 3.7 below.

[4] This is meant as a way of saying that the software is "for maemo", which is acceptable use of the trademark (see the trademark policy).

[5] The new '~' syntax would achieve the same result, see `http://lwn.net/Articles/194664/`

3.5 Virtual packages

Virtual package names intended for generic use SHOULD be commonly agreed and appear in the list included in the `maemo-policy` package.

Virtual package names MAY be created privately for use between a certain group of packages, provided that the chosen name clearly belongs to those packages (using e.g. *foo-theme* for package *foo*, or similar).

3.6 Base system

Maemo doesn't have a base system in the sense that Debian does, as there is no installer. In some ways everything that is included in the device (or SDK) images could be considered a "base system", but this does not affect packaging.

3.7 Essential packages

Packages marked Essential are always installed in Debian based systems, and according to the Debian Policy, other packages should not list dependencies to them unless a version dependency is needed.[6]

However, default maemo installations don't have all of the Debian essential packages, and therefore binary packages MUST list their *direct* dependencies also to packages that are essential in Debian *but missing in maemo*. (This is especially important to note when porting packages from Debian to maemo.) These packages are:

- diff
- e2fsprogs
- ncurses-bin
- ncurses-base
- perl-base[7]

Also, some of the essential packages are provided by BusyBox[8] in maemo, but it doesn't provide all commands that are included in the corresponding Debian packages. These commands include e.g. the real `bash` shell, `md5sum` (from coreutils), `locate` (from findutils), etc. Some of the BusyBox commands also lack options which their GNU counterparts on a normal Linux desktop provide (such as `-iname` in `find` or `-o` in `uname`).

As a result, package installation and uninstallation MUST be tested on a device which doesn't have extra software installed, before the package is uploaded to the repositories. Testing package installation in Scratchbox is **not** enough, as Scratchbox uses the real Debian binaries instead of BusyBox.[9]

[6] Note that emdebian doesn't have any essentials. There have been proposals that maemo shouldn't have them either; this policy section describes the current state.

[7] perl-base has been included in some releases but is not guaranteed to stay. Packages need to declare the dependency.

[8] See `http://www.busybox.net/`

[9] It might be possible to use QEMU system emulation (or a minimal root image with all scratchbox host tools disabled?) for testing in scratchbox, but no-one has created such an environment yet.

3.8 Prompting in maintainer scripts

As users do not see the command line, prompting the user MUST be done using GUI tools.

The following tools are provided in maemo for this:

maemo-confirm-text Present user with text and ask for confirmation[10]

maemo-select-menu-location Ask user where in the menu hierarchy an application's entry should be placed

A package using these MUST depend on `maemo-installer-utils` (for maemo-confirm-text) or `maemo-select-menu-location`.

For more information on the tools, see [Maemo Application Packaging].

All other installation time interaction requirements at SHOULD be first discussed on the `maemo-developers` mailing list.

TODO: Would be nice to have a Hildonized GUI frontend for `[c]debconf`[11].

3.9 Separating files into binary packages (new section)

Due to the limited space on available on devices, any files that are not needed at run time SHOULD either be removed from the binary packages or put in a separate binary package which is not installed by default. See the following subsections for details.

If this requires packaging the sources differently from the upstream distribution (Debian), the packaging changes SHOULD be propagated back to upstream (unless the changes are incompatible with the upstream policy or practices, as might be the case e.g. with documentation and localization packaging).

3.9.1 Library and program packages

Libraries SHOULD be packaged separately from programs using them. This also encourages better API maintenance.

3.9.2 Development packages

If there are development files (static libraries, headers, test programs etc), they MUST be in a separate development (`-dev`) package. Libtool `*.la` files MUST NOT be included in any packages, as they cause problems with indirect library dependencies.

[10] This can also be used in preinst, e.g. to display a license agreement before the package is unpacked.

[11] Currently the device doesn't have either of those available. At least `cdebconf` has dependencies that maemo doesn't have, so some patching would probably be needed. Also unclear how to handle localizations.

3.9.3 Localization packages

Localization files SHOULD be packaged separately from the programs using them, and according to language or locale. This both saves flash memory and reduces the amount of data users have to download when installing or upgrading a package.

TODO: This is still under discussion. An exception could be if the localization files are very small. Also, rules for localization package naming have not been decided. What do Debian and Ubuntu do (and is there any debhelper to simplify the task)?

3.9.4 Documentation packages

If documentation is included, it SHOULD be packaged separately. Also, if end-user documentation is localized, the version for each language or locale SHOULD be packaged as noted in 3.9.3 above.

Developer documentation SHOULD be packaged separately from end-user documentation, as developer documentation shouldn't need to be installed on end-user devices.

API documentation for libraries SHOULD be auto-generated from the sources[12].

3.9.5 Debug packages

All binary packages which contain executable code SHOULD have a corresponding debug (`-dbg`) package with debug symbol files for all ELF binaries and shared libraries in the corresponding binary package.[13]

Debug symbol files are required because on ARM (unlike e.g. on x86) optimized binaries or libraries *cannot*[14] be debugged without them. Debug symbols also help in getting more useful performance and memory profiling results.

The `debug-pkg-check` script from the `maemo-debug-scripts` package can be used to verify debug packages. See also section 4.3.2 (Compilation options and stripping) and [Creating Debug Packages].

TODO: This rule is most important for libraries and plugin based applications. Final policy on whether to require or just recommend it for other executables is undecided. As the current Debian build tools cannot build debug packages automatically (unlike e.g. Ubuntu), all relevant packages need to be modified.

3.9.6 Configuration packages

If the packaged software may have different configurations (e.g. for different regions, products, customizations, device and SDK environments, etc.), those configurations SHOULD be packaged separately. Configurations could be simple files in `/etc`, or something more complex such as themes or pluggable modules.

[12] The preferred tools for this are `gtk-doc` (for anything using or based on GLib) and `doxygen` (other sources)

[13] See the `--dbg-package` option in `dh_strip`

[14] Unless the -fno-omit-frame-pointer GCC option is used. This could be considered a GCC bug as according to the GCC manual -O[x] options shouldn't enable any optimizations which make debugging impossible.

Configuration packages SHOULD depend on the package that uses them (to avoid leaving the configuration installed and consuming flash space when the actual software is uninstalled).

The application package SHOULD only depend on the configuration if it is required for the software to function. If there can be any number of alternate configurations, a virtual package SHOULD be used for the dependency. If alternate configurations cannot coexist, they MUST declare a conflict (using the virtual package if provided).

This section mainly concerns software that is part of the maemo platform, but separating programs from configurations is often a good practice for other packages as well. Please consider even building the configuration from a different source package; this way the whole program doesn't have to be rebuilt (and possibly approved) for a simple configuration change, and the user has less packages to download when upgrading.

4 Source packages

4.1 Package relationships

Unlike the run-time environment on maemo devices (see section 3.7), the maemo *build* environment (= Scratchbox + Debian dev-kit) provides "host" versions of all the Debian essential and build-essential tools, so those SHOULD NOT be listed in the package build dependencies.

4.2 Changelog: `debian/changelog`

The main difference between maemo and Debian changelogs is that maemo uses a different Bugzilla bug tracking system (and there are actually two of them). While bug fixes in Debian are marked with a "Closes:" entry in the changelog, in maemo they are marked with "Fixes:"[15]. A Bug ID of the form of *NB#xxxx* refers to the Nokia internal BTS and *MB#xxxx* refers to the public maemo BTS. These are used for automated tracking of bug state changes.

The distribution and urgency fields in changelog are currently ignored.

TODO: Bugs in the Garage BTS could be marked with *GB#xxxx*, although it's not used in the current bug tracking tools.

4.3 Main building script: `debian/rules`

4.3.1 DEB_BUILD_OPTIONS (new section)

The DEB_BUILD_OPTIONS environment variable can be used to control certain features when building packages. If a package supports a feature discussed below, it MUST use the given option for controlling enabling of that feature.

Options currently used by the maemo build infrastructure:

thumb Whether to enable compilation with the ARM thumb instruction set[16], see ap-

[15] Other Debian derived distributions use other keywords, such as "LP:" in Ubuntu.

[16] For more info, see: `http://wiki.debian.org/ArmEabiPort`

pendix B.1 for an example. In most cases this can make the resulting executable code much smaller (usually ~20%) and some functions faster (usually up to 2x if the code now fits completely into the CPU instruction cache). Before using this option in a package, it should be checked that it doesn't have an adverse effect on performance critical code, as in some cases it can also make code slower.

vfp Whether to enable compilation with the ARM vfp (limited floating point) instruction set[17], see appendix B.2 for an example. Vfp can make functions doing a lot of floating point operations much faster (up to 10x). As it can also make code slower and larger (because currently it cannot be used together with thumb[18]), its systematic use is not recommended. Test which functionality in the package profits most from it and enable vfp only for those objects. This option is used only when building a package for a CPU architecture supporting vfp (770 doesn't, later devices do).

nolauncher Request application to be built without maemo-launcher[19] support. Note that if an application supports maemo-launcher, it should be enabled by default, as it makes the application start up significantly faster and saves memory[20].

parallel=N Asks the package to enable GNU Make's `-jN` parallel build option to speed up the package build, see appendix B.3 for an example. If this option is not set, the package should be built sequentially (e.g. to save memory).

Deprecated options:

maemo-launcher Application should be built with maemo-launcher support. This should now be the default behavior. See the *nolauncher* option above.

softfp Use soft floating point. After IT OS 2005, all of the IT OS 2006, 2007 and 2008 toolchains use soft floats by default.

TODO: Proposed/optional build options:

nodocs Asks the package to neither build nor install documentation. Building documentation (especially with gtk-doc) can take a long time and the documentation packages are not needed on the device, only in the development environment. It should be possible to generate the documentation only when needed.

nocheck Asks the package to neither build, run nor install tests. Running tests might take a long time or require a specific test environment, so it should be possible to do it only when needed. See the nocheck proposal in the Debian Bugzilla: `http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=416450`

[17] Notes about VFP opcodes and accuracy:
`http://maemo.org/pipermail/maemo-developers/2005-October/001602.html`

[18] THUMB-2 would allow using thumb mode with vfp instructions, but that's not supported by current HW/toolchains.

[19] See `https://stage.maemo.org/svn/maemo/projects/haf/trunk/maemo-launcher/README`

[20] Applications that don't use Hildon libraries or link to large C++ libraries can be exceptions to this. Please measure.

4.3.2 Compilation options and stripping (new section)

The Debian Policy "Binaries" section recommends optimizing and debug symbols: `http://www.debian.org/doc/debian-policy/ch-files.html#s10.1`

In maemo these rules are more strict:

- Code MUST be compiled with optimization to produce a faster or smaller executable (unless impossible because of compiler bugs). Generally `-O2` or `-Os` should be used, or `-O3` if that works better. If the compiler crashes when full optimization is enabled, `-O1` can be used instead.
- If sources are freely available, they MUST be compiled so that the resulting binaries contain debug symbols (`-g` compiler option(s)).
- The resulting binaries and libraries MUST be stripped and the debug symbols separated to debug symbol files (and put into `-dbg` debug packages, see section 3.9.5).
- If sources are not available for a library used by another component, or if the full debug information cannot be provided e.g. for contract reasons, that library SHOULD either be compiled with `-fno-omit-frame-pointer` or provide at least minimal debug symbols[21] so that problems in other components can be debugged through that library.

4.4 Original sources (new section)

When packaging upstream sources, the original sources SHOULD be provided separately from the `debian/` directory (and patches, etc.) in an `.orig.tar.gz` file. (This is not different from Debian but is currently not followed by all maemo packages. This policy may be changed to a "MUST" severity later, i.e. non-compliant packages will not be accepted.)

5 Control files and their fields

5.1 Package control files: `debian/control`

Distribution, priority and urgency fields are currently ignored. To be safe, use `optional` for priority unless there's a very good reason not to.

See the notes for other fields below.

5.1.1 `Maintainer`

If an upstream package is modified for maemo, the `Maintainer` field MUST be changed from the original (unless the original maintainer also maintains the maemo package).

The original maintainer MAY be marked with the field `XSBC-Original-Maintainer`.

[21] The `.debug_frame` section is enough for this, see [Creating Debug Packages] for more info

5.1.2 `Section`

See the discussion in 2.2 (Sections).

5.1.3 `Architecture`

Architectures currently supported by maemo are `i386` and `armel` (in addition to `any`, `all` and `source`).

Note: There's also a maemo specific `XS-scratchbox-architecture` field which can have the value `uarm` to indicate that the package is to be compiled with the uClibc toolchain instead of Glibc. This is used only for building the kernel and initfs, and may be phased out if they switch to using Glibc.

5.1.4 `Depends`

Packages MUST depend only on their *direct* dependencies, which MUST be generated automatically whenever possible (using `${misc:Depends}`, `${shlibs:Depends}` etc. — see "`man debhelper`")

5.1.5 `Pre-depends`

Pre-dependencies should be avoided when possible, and any needs SHOULD first be discussed on the *maemo-developers* mailing list.

5.1.6 `Standards-Version`

The `Standards-Version` field is for the Debian Policy version (latest[22] is v3.7.2.2).

TODO: A separate `XBS-Standards-Version` field could be defined for specifying the maemo packaging policy version. Currently this is not done yet, as the policy is still a draft, and we don't have tests for maemo specific policy compliance.

5.1.7 `Installed-Size`

Note that a package may take less space than indicated by this (automatically calculated) field if the JFFS-2 filesystem is able to compress the files.

[22] Latest when the work on this Policy draft was started. 3.7.3.0 has been released since then (and this document should be updated).

6 Package maintainer scripts and installation procedure

6.1 Differences between scratchbox and device environments (new section)

Maintainer scripts MUST take into account the differences of the device and scratchbox environments, and behave sensibly in both. The differences SHOULD NOT cause unnecessary warnings to be printed.

If a script needs to detect whether it's running in scratchbox, currently one way to do so is to test the presence of `/targets/links/scratchbox.config` (although this is not guaranteed to work in future versions of scratchbox).

`init.d` scripts MUST either be invoked using `invoke-rc.d`, or only run directly if the status returned by `policy-rc.d` allows it. (In practice the policy is such that services will not be started in scratchbox.)

GConf values need to be handled directly[23] in scratchbox, since gconfd (or DBus which is used for communicating with gconfd) is not running.

Please also note the shell commands available for scripts, as discussed in sections 3.7 (Essential packages) and 10.3 (Scripts).

TODO: Some general description of services available during package installation (in scratchbox or device) could be useful. Probably rather a topic for some other document, add link here if created.

6.1.1 Lifeguard reboots

Note that the lifeguard will reboot maemo devices if certain processes exit. Packages must take care that they do not cause such reboots during installation (either by restarting a service or by causing clients of a service to exit), as this could result in a broken package database. (This is mainly an issue for upgrades or replacements for system packages, as no others are watched by the lifeguard by default.) Package installation and upgrades MUST be tested in a live system (on an actual device).

6.2 Controlling terminal for maintainer scripts

Maintainer scripts do not have a controlling terminal when running on the device, and are only able to interact with the user through a GUI tool. See section 3.8 (Prompting in maintainer scripts).

7 Declaring relationships between packages

Like in Debian Policy. (Note, however, the differences in Essential packages, as discussed in section 3.7.)

[23] See the `--direct`, `--config-source` and `--makefile-install-rule` options in `gconftool-2`

8 Shared libraries

Like in Debian Policy.

9 The operating system

9.1 File system hierarchy

TODO: This section has not been written yet.

9.2 Users and groups

The "user" account has the UID and GID 29999 in maemo. The base `useradd` utility does not handle policy issues (the `adduser` wrapper is not available in maemo), and therefore dynamically added users and groups (including system users) currently go into the 30000–59999 range which is reserved in Debian. There is currently no decision to change this, but the numeric ID's should not be depended on in any way.

To maintain compatibility, packages MUST NOT allocate UID's or GID's in the global 0–99 and 60000–64999 ranges except as allowed in Debian.

TODO: Possibly notes about sudo or users in scratchbox

9.3 System run levels and `init.d` scripts

See the notes about `invoke-rc.d` and scratchbox in section 6.1.

9.4 Console messages from `init.d` scripts

Maemo has no strict requirements or policy for console messages, as the user doesn't normally see them. For consistency (and special circumstances such as debugging) it's still recommended that packages follow the guidelines of the Debian Policy.

9.5 Cron jobs

Currently cron is not available in maemo.

9.6 Menus

Applications in maemo are listed in the application menu if they provide a `.desktop` file in the `/usr/share/applications/hildon` directory. Files added to or removed from this directory are detected automatically.

By default the entry is shown in the Extras menu if no other location has been selected previously. A package can use the `maemo-select-menu-location` utility[24] to provide a different default and let the user select the location. This SHOULD only be done on the first installation (not upgrades).

Internally the menu structure is defined in `/etc/xdg/menus/applications.menu` and `${HOME}/.osso/menus/applications.menu`[25], but packages SHOULD NOT modify these files directly.

Note that the menu structure is completely different from Debian. Installing an unmodified Debian package with a proper `.desktop` file under `/usr/share/applications` will result in the application being displayed in the "Extras" category. Maemo also doesn't provide (and has currently no use for) the menu tools in the xdg-utils package.

9.7 Multimedia handlers

Maemo uses the freedesktop.org standards[26] for registering MIME types and handlers. Please see [Maemo Tutorial] for an example.

The MIME utilities from the xdg-utils package are not available in maemo.

TODO: Document here what packages need to do w/ MIME types and in postinst/postrm etc. For now please refer to the tutorial. (Also appendix D?)

9.8 Registering Documents using doc-base

The Debian doc-base package is not used.

Please see [Maemo Documentation Framework] for how to create help files with the maemo help framework.

10 Files

10.1 Binaries

As in Debian, but note the policy on build options, optimization and stripping in sections 4.3.1 and 4.3.2.

10.2 Libraries

Note the policy on `.la` files in section 3.9.2.

[24] See [Maemo Application Packaging]

[25] Following the freedesktop.org menu specification (`http://standards.freedesktop.org/menu-spec/latest/`), but `hildon-desktop` only implements a subset of it.

[26] `http://standards.freedesktop.org/shared-mime-info-spec/shared-mime-info-spec-latest.html`, `http://standards.freedesktop.org/desktop-entry-spec/latest/ar01s07.html`

10.3 Scripts

The `/bin/sh` on maemo devices is provided by BusyBox, and (unlike in Debian) the Bash shell is not available by default. Therefore, all shell scripts MUST work with a POSIX compatible shell.

Perl scripts SHOULD be replaced with POSIX compatible shell scripts if possible. (If a package has to use perl scripts, it MUST depend on perl-base, as noted in 3.7 (Essential packages).)[27]

10.4 Configuration files

Applications that store configuration options elsewhere in the user's home directory than in `$HOME/MyDocs` SHOULD add a configuration file telling the maemo backup system what data needs to be backed up and restored. If the application's configuration file format changes between releases, its restore script needs to be able to handle the older format appropriately. See [Maemo Tutorial] for more information.

Note that user configuration files under `$HOME`, and all GConf settings except defaults set in schemas, are removed if the user restores original device settings using the Control panel.

As also recommended in the Debian Policy, packages SHOULD NOT place files in `/etc/skel`: there is no good way to make them available to the existing user (and on maemo devices new users are normally not created).

10.5 Log files

Due to limited space, log files SHOULD be kept minimal if used at all. If a log file causes the device to run out of disk space, it can make the device unusable and force the user to reflash.

Log files should preferably be in a place where the user can remove them (through the UI), and they SHOULD NOT grow without bounds.

The logrotate package is not available (and simple rotation is not necessarily enough to limit the size of log files, depending on the logging behavior).

TODO: Should packages always remove their logs in postrm, define policy on this?

TODO: User should probably be able to decide whether to enable logging (at least for potentially large logs). This is more of a UI issue and not necessarily in the scope for policy.

11 Customized programs

11.1 Daemons

Maemo devices don't (by default) have any network services or an `inetd` daemon. Anything installed afterward SHOULD conform to Debian.

[27] When porting packages from Ubuntu, note that python isn't essential in maemo either, and packages using it need to declare the dependency.

TODO: Currently there is no formal process for requesting maemo-specific additions to `/etc/services`, `/etc/protocols` or `/etc/rpc`. Any needs for such should be discussed on the maemo-developers list.

11.2 Editors and pagers

By default the EDITOR or PAGER environment variables do not have a value (and the `/usr/bin` aliases specified in the Debian Policy do not exist).

12 Documentation

12.1 Manual pages and info documentation

Maemo devices don't have the utilities needed for viewing manual or info pages. As noted in 3.9.4, documentation should be packaged separately and installing it should be optional.

12.2 Preferred documentation formats

Documentation in well integrated maemo applications SHOULD be in the format used by the maemo help framework, see [Maemo Documentation Framework].

Note that although HTML documentation can be viewed in the browser, users would typically not be able to find it in `/usr/share/doc` (as used in Debian), since those directories are not visible in the File Manager.

12.3 Examples

Tests and examples are not needed for running the programs, so they SHOULD be in a separate package which the user does not have to install.

A Common problems

TODO: Common problems in maemo packages. Can also be in the wiki

B DEB_BUILD_OPTIONS examples

B.1 Thumb option

Add to `debian/rules` these lines:

```
# Use soft-float and thumb mode if it enabled.
ifneq (,$(findstring thumb,$(DEB_BUILD_OPTIONS)))
CFLAGS += -mthumb
endif
```

This change can be validated after build by checking the binaries with the `size` command (should be smaller), or by checking the addresses of function symbols which should be odd in THUMB mode — e.g.:
`readelf -s my-binary | egrep ': [0-9a-f]+[13579bdf] .*FUNC'`

B.2 VFP option

Add to `debian/rules` these lines:

```
# Use hardware floating point
ifneq (,$(findstring vfp,$(DEB_BUILD_OPTIONS)))
CFLAGS += -mfpu=vfp -mfloat-abi=softfp
endif
```

B.3 Parallel option

TODO: When this option has been added to the Debian policy, add link here to an appropriate section. For now, see the discussion in the following Debian policy bug:
`http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=209008`

B.4 Examples of nolauncher, nodocs and nocheck

TODO.

C Example of creating `-dbg` packages

TODO: Needed? (Already described in [Creating Debug Packages])

D Maintainer script utilities

TODO: List and briefly document the maemo-specific tools used in maintainer scripts.

E List of essential packages

TODO: A complete list of essential packages in maemo should be created and included in the maemo-policy package. A copy can be inserted here. Currently please refer to Debian and the differences described in section 3.7.

F List of virtual packages

TODO: The list of virtual packages in the maemo-policy package (when it has been created) can be inserted here.

G Lintian overrides

TODO: The differences between what the Debian `lintian` tool reports and what should be reported for maemo should be analyzed. Overrides and/or additional checks could be provided in the maemo-policy package, and possibly also listed here.

References

[Debian Policy]	Debian Policy Manual: http://www.debian.org/doc/debian-policy/
[Debian New Maintainers' Guide]	Debian New Maintainer's Guide: http://www.debian.org/doc/manuals/maint-guide/
[Debian Social Contract]	Debian Social Contract: http://www.debian.org/social_contract
[Maemo Application Packaging]	Making maemo Application Packages: http://maemo.org/development/documentation/how-tos/4-x/making_application_packages.html
[Quick Start Guide]	Maemo Quick Start Guide: http://maemo.org/development/documentation/maemo-quick-start-guide.pdf
[Maemo Coding Guidelines]	Maemo coding style and programming guidelines: http://maemo.org/development/documentation/how-tos/4-x/maemo_coding_style_and_programming_guidelines.html
[Maemo Tutorial]	Maemo (Developer's) Tutorial: http://maemo.org/development/documentation/tutorials/maemo_4-0_tutorial.html
[Maemo Trademark Policy]	Maemo trademark usage: http://maemo.org/intro/trademarks/trademark_usage_guidelines.html
[Maemo Documentation Framework]	Maemo Documentation Framework: http://maemo.org/development/documentation/how-tos/4-x/help_framework_howto.html
[FHS]	Filesystem Hierarchy Standard: http://www.pathname.com/fhs/
[Creating Debug Packages]	Creating debug packages: http://maemo.org/development/documentation/how-tos/4-x/how_to_make_a_debian_debug_package.html

[Maemo Policy wiki]	Maemo packaging policy Wiki, a part of the policy change and documentation process: **TODO**: Not created yet