Base Library Priorities

- Introduction... GNUstep priorities
- MacOS-X (Cocoa) compatibility
- Stability and Reliability
- Cross platform portability issues

GNUstep priorities

- GNUstep is a cross platform development system
- Provides MacOS-X/Cocoa compatibility
- Write once, build/run anywhere.
- Uses the same standard toolset everywhere.
- Is the core on which a desktop is built.
- People wanting a cross platform development system should definitely be contributing to GNUstep
- People wanting a NeXTSTEP like system should contribute to the Backbone project and to GNUstep
- People wanting to try out new, non-standard technologies, should contribute to th Etoile project and to GNUstep

Cocoa compatibility

- All original Cocoa API present and working
- Except Apple scripting extensions
- Possibly faster and more reliable than Cocoa
- Complete some newer classes
- Add new methods to existing classes
- Need to add some newer classes
- Need option to build as a framework

Recent additions

- NSNetServices ...
- large contribution by Chris Vetter
- needs testing on windows
- NSXMLParser, used as fallback if no libxml2
- contribution by Nikolaus Schaller
- also NSPredicate and others.
- New URL stuff, incomplete.
- NSStream code, incomplete
- contribution by Derek Zhou
- NSAffineTransform (from gui)

Features to complete

- Key value observing
- SSL streams (unix and windows)
- SOCKS streams
- New URL system
- The latest methods for existing classes

Classes to add

- NSLocale
- Apple scripting classes
- NSCalendar/NSDateComponents
- NSXML tree representation classes
- NSSpellServer (from gui)

Other considerations

- Make base available as a framework
- CoreFoundation ... features not in Cocoa
- Do we want to support this API?
- If so, what parts do we prioritise?
- Can we build it on base?

Stability and Reliability

- Already very stable/reliable
- Keep it that way
- Backward compatibility
- Avoiding conflicts
- Release strategy

Already very stable/reliable

- Compares well with Foundation for bugs
- Compares well with Foundation for speed
- API complete wrt early Cocoa
- Feature removal deprecated first
- Stable releases typically at 8 month intervals

Keep it that way

- Everything else depends on base.
- Rigorous deprecation cycle
- Keep using and extending regression tests
- Add only MacOS-X classes/methods
- Any exceptions need very full debate

Backward compatibility

- Historically, maintained API compatibility
- Binary compatibility desirable for distributions
- Make feature deprecation/removal slow
- Make no changes to constants
- Make no changes to ivar layouts
- How can we check/ensure this?
- How soon can we be ready for this?

Avoiding conflicts

- Use of the NS prefix is a good start
- Use of _ prefix for private methods is good
- Use of _ prefix for private ivars is good
- Should use _GS for private classes/functions
- GS_ATTRIB_PRIVATE on functions
- tells compiler to hide function from outside

Release strategy

- New stable branch and (unstable) trunk
- Make frequent 'bugfix' releases from stable
- Make frequent 'unstable' releases from trunk
- ABI/API constant between bugfix releases
- ABI/API varies between unstable releases
- Old apps run with any bugfix release
- Occasionally (18 months?) make a 'full' release
- Trunk is copied to the stable branch and we ask distributions and apps to upgrade.

Cross platform portability

- Completeness on all platforms
- Configured implementation differences
- Dependencies
- Microsoft windows

Completeness on all platforms

- Write once, build/deploy anywhere is broken if something only works on some platforms
- Should encourage contributors to support all platforms (even mswindows)
- Should not release a feature (except in an unstable release) if it is not supported on all major platforms
- Must be flexible about this, but document any exceptions very well.

Configured implementation differences

- Some platform differences are inevitable
- Performance trade-offs (cpu/memory etc) vary by platform (eg. PDAs)
- May require different data structures and algorithms for the same functionality.
- Need a mechanism to formalise conditional compilation options.
- Perhaps an extra internal header file?

Dependencies

- Base has very very few dependencies, gcc, bash, make, gnustep-make, ffcall or ffi to build, ffcall/ffi libraries at runtime.
- Partial dependencies are openssl for HTTPS support, libxml2/libxslt for GSXML extensions and dns_sd for NSNetServices.
- Should depend on only standard tools/libraries where possible
- Make use of platform specific tools/libraries,but have a fallback mechanism where they are not present.

Microsoft windows

- Mswindows is so different to other platforms that we have to accommodate it specially.
- To maintain a consistent toolset we have to use mingw/msys and ports of various standard tools.
- We need an installer to easily provide the development environment.
- We need an installer for runtime dependencies too.
- There is room for a separate project to develop a native windows toolset.