 Version 1.7.1.

Submission Suggestions

As the MiscKit grows, it becomes increasingly important that things be done in certain ways so that submissions are all of at least a baseline level of quality. This document includes helpful hints that will add that extra touch to your submissions that makes them beyond the ordinary.

These suggestions will also help the administrator move your submissions into the MiscKit more rapidly by reducing the amount of work necessary.

Here is a list of the topics covered in this document:

;Submissions.rtfd;BasicFunctionality;¬ Basic Functionality
;Submissions.rtfd;SourceCodeFormatting;¬ Source Code Formatting
;Submissions.rtfd;QualityControl;¬ Quality Control
;Submissions.rtfd;NeXTMail;¬ Using NeXTMail to make submssions
;Submissions.rtfd;CopyrightNotices;¬ Copyright Notices
;Submissions.rtfd;HeaderImports;¬ Header Imports
;Submissions.rtfd;ObjectVersioning;¬ Object Versioning
;Submissions.rtfd;Thanks;¬ Thanks (Acknowledgements)

BasicFunctionality;¬Basic Functionality

One of the things that is considered important is that all MiscKit objects be archiveable and capable of being passed bycopy over Distributed Objects. As such, you should try to adopt the NXTransport and ±read:/±write: methods in all submissions of Objective-C classes. This may seem like a small thing, but most people have come to expect these capabilities of all MiscKit objects.

To aid in developing sub-libraries and palettes for the MiscKit, several palette

project templates may be found in the Examples area in the ªProjectTemplatesº folder. It is important to set certain flags in the Makefile preamble/postamble in order for things to be folded into the MiscKit easily.

You should also be sure to implement object versioning as described below;Submissions.rtfd;ObjectVersioning;¬, so that you may avoid future incompataility problems.

QualityControl;¬Quality Control

Although the MiscKit administrator will compile and run your code to make sure it at least superficially appears to work, quality control is the responsibility of the submitter/maintainer of a MiscKit resource. Be sure to thoroughly test your code before you submit it. If code is untested, *say so* in the documentation so that people don't waste too much time scratching their heads when things are broken. It is preferred that you simply submit code that works¼

If your code generates warnings when compiled (or compile time errors) it will

probably not be folded into the MiscKit until the warnings and/or errors are eliminated unless there is no reasonable way to do so and they are shown to be inconsequential. Remember that each warning requires an entry in the README.rtf explaining that it is an ªOKº warningÐand that file is too long already.

SourceCodeFormatting;¬Source Code Formatting

Everybody has their own way of formatting C source code. Rather than impose upon what is comfortable to a submitter, we simply ask that the code be formatted in such a way as to be readable to another programmer. This is deliberately vagueÐuse whatever style is most comfortable for you.

There are two requests, however. First, avoid using RTF source code. Although nifty, non-NeXT users will have trouble with it. Since some parts of the kit are still useful even without NEXTSTEP, authors should be respectful of non-NEXTSTEP users who might wish to benefit from the MiscKit. Second, most MiscKit source code looks best with Edit.app set to 4 spaces per tab stop. We recommend that everyone set Edit to four spaces per tab so that code formatting will be better

preserved when others are viewing it. (ie: Do both your formatting and viewing with Edit.app set this way).

NeXTMail;¬Sending in submissions with NeXTMail

It is highly encouraged that you use NeXTMail to send your submissions to Don as soon as they are ready. With programs such as **tnextmail** readily available, this is easy to do even from non-NeXT machines.

There is one major caveat, however. If you have links in your projectsÐor you attempt to mail a link to your source code, NeXTMail will send the link and not the actual files! This is important because typically one out of every three or four messages that are sent to Don contains a link instead of the actual files. Even people who know about this are continuously being bit by the problem. As Ernest Prabhakar put it:

ªThey can upload a file to the moon, but they can't make it follow a link...º

Unfortunately, NeXT does not consider this a bug, but rather a feature. The reason is because most large organizations cross-mount all the major filesystems. As such, NeXTMail within the organization will send linksÐnot filesÐthereby keeping inter±organization mail bandwidth to a minimum. That is a good thing, but not when we start sending things over the Internet. The best thing would be a preference in Mail.app that allows you to say to not follow links for the local domain but to follow them on external domains, perhaps allowing you to set which domains links are not followed on and assuming a complete copy for all others. NeXT does have this in their bug database, so don't bother reporting it to them. Eventually something might be done about it. In the meantime, please be sure to check your outgoing mail to be sure that you actually sent the attachments you meant to send and not just links to them.

CopyrightNotices;¬Copyright Notices

It is important that *every* file in your submission contain a copyright notice. Since you wrote the file, you hold the copyright. In all RTF files you should use the ª©º symbol (try Alternate-c to get it) in a message such as this:

Copyright © <year> <copyright holder>. All Rights Reserved. Version <version>.

If in a non-RTF file, you must completely spell out the word ªCopyrightº since typing ª(C)º is not sufficient. In a source code file, something like this should be at the top:

```
//
//  <file name> -- <one or two line summary>
//       Written by <author name> Copyright <year> by <copyright holder>.
//                Version <version>.  All rights reserved.
//
//       This notice may not be removed from this source code.
//
//  This object is included in the MiscKit by permission from the author
//  and its use is governed by the MiscKit license, found in the file
//  "License.rtf" in the MiscKit distribution.  Please refer to that file
//  for a list of all applicable permissions and restrictions.
//
```

All submissions need to have a copyright notice and a paragraph mentioning the fact that it has been licensed for use in the MiscKit according to the terms of the

License.rtf file. Without this, the submission cannot legally be added to the MiscKit. The easiest way to comply is cut and paste the above text into your submission.

Note that usually the copyright holder will be the same as the authorÐbut this does not have to be the case. The author can transfer the copyright to someone else.

You *can* use the actual © symbol itself in your source code, as well. If you have RTF source code, just type Alt-c or Alt-C. Since RTF source code is *strongly* discouraged, however, the only way to enter the © symbol is to type, in a Terminal.app window:

```
echo "c" | tr 'c' '\240'
```

It will give you a character you can cut and paste. Once you've grabbed the character, you may want to add it to the expansion dictionary in Edit.app (Command-E).

HeaderImports;¬Header Imports

When you are testing your submissions, you will probably include your headers using double quotes. Once in the MiscKit, your source file will need to find the header file elsewhereÐthe headers are kept in a different directory than the source. Before submitting your code, you should therefore change to the ª<>º style of imports. For example, say you are building a MiscFile object. An include like this

```
#import "MiscFile.h"
```

should be changed to this

```
#import <misckit/MiscFile.h>
```

before you make your submission. The administrator has to make this change if you don't, so please make his life a little easier...

If this causes compilation problemsÐand well it mayÐlook at the existing MiscKit palette and other projects to see how this has been solved using -I flags in the

Makefile.preamble and soft links withn the project folder.

ObjectVersioning;¬Object Versioning

It is important that your objects support versioningÐespecially those which are palettized. By doing this, changes in object's instance variables won't render .nib files with the old object unloadable in InterfaceBuiler. We strongly recommend that you use versioning in all your objects, however. If your object contains ±read: and ±write: methods, you should use versioning!

Here, contributed by Jason Beaver, is some source which clearly shows what you need to do to make sure that versioning of an object is handled correctly, with the important code in boldface:

MyClass.h ¬MyClass.m ¬

Note that the instance variables archived are an example set; your object will obviously use different variables. This simply demonstrates the general idea.

The basic idea is:

- Write out the latest version (this is not a hard and fast rule...you might want to write out an object which can be read back in by an older version of your program),
- Read the version of the object from the NXTypedStream.
- Read the data which was archived in that version of the NXTypedStream.
- Set any instance variables which have been added to the class since that version to reasonable values which make the object function as it did in that version.

Thanks;¬Thanks

Thanks to those who have contributed to this document:

Jason Beaver jason@dbkit.com (NeXTMail accepted)
Don Yacktman Don_Yacktman@byu.edu (NeXTMail accepted)