A

Data Formats

To make it easier for applications to share information, the NEXTSTEP pasteboard supports a number of standard data formats. Each format, or *pasteboard type*, is identified by a global variable:

Variable Name	**Type Description**
NXAsciiPboardType	Plain ASCII text
NXPostScriptPboardType	Encapsulated PostScript code (EPS)
N3DRIBPboardType	RenderMan Interface Bytestream code (RIB)
NXTIFFPboardType	Tag Image File Format (TIFF)
NXRTFPboardType	Rich Text Format (RTF)
NXSoundPboardType	Sound data
NXFilenamePboardType	ASCII text designating a file name
NXTabularTextPboardType	Tab-separated fields of ASCII text
NXFontPboardType	Font and character information
NXRulerPboardType	Paragraph formatting information

Data in other formats can also be placed in the pasteboard. However, the sending and receiving applications must both agree on the structure of the format, its name, and how to interpret it. Other formats may be adopted as standards in the future.

Each of the standard formats is discussed below. In most cases, the discussion is short and consists only of a reference to the primary source document for the format. In some cases, more information is given on modifications to or interpretations of the format in the NEXTSTEP environment.

NXAsciiPboardType

Text in this format consists only of characters from the ASCII character set as extended by NEXTSTEP encoding. None of the characters is given a special interpretation (in contrast to NXTabularTextPboardType and NXFilenamePboardType, for example). Standard ASCII is documented on-line in **/usr/pub/ascii** and the **ascii**(7) manual page. NEXTSTEP encoding is documented in Appendix C, ªKeyboard Event Information.º

NXPostScriptPboardType

This type is defined as PostScript code in the Encapsulated PostScript Files format (EPS). The PostScript language is documented by Adobe Systems Incorporated, principally in the *PostScript Language Reference Manual*. EPS conventions are documented in *Encapsulated PostScript Files*

Specification, also by Adobe.

N3DRIBPboardType

This type is for RenderMan Interface Bytestream (RIB) code. The format of RIB code is documented in *The RenderMan Interface*, by Pixar.

NXTIFFPboardType

This type is for image data in Tag Image File Format (TIFF). TIFF is documented in *Tag Image File Format Specification*, by Aldus Corporation and Microsoft Corporation.

TIFF support in the current NEXTSTEP release follows version 6.0 of the TIFF standard and is based on version 3.0 of Sam Leffler's freely distributed TIFF library. This library provides a good set of routines for dealing with TIFF files that conform to the 6.0 specification.

NEXTSTEP TIFF support is embodied in the Application Kit's NXBitmapImageRep class and the command-line program **tiffutil**. See the class specification for NXBitmapImageRep in Chapter 2, ªThe Application Kit,º and the **tiffutil**(1) manual page for more information.

Unsupported Fields

In the current release, some fieldsÐprincipally those having to do with response curvesÐ will be read correctly but ignored when imaging the data. Color palettes are not supported except when the palette entries are 8 bits and the stored colors are 24 bits. These files will be read correctly and converted to 24-bit images on the fly.

Multiple Images

Multiple forms of an image can now be stored in the same fileÐthat is, under the same TIFF header. ªMultiple formsº might mean the same image at different resolutions (for example, 72dpi and 400dpi) and at different bit depths or colors (for example, 2 bits per sample on a gray scale and 4 bits per sample RGB).

This feature is useful when you want to create color icons for an application and its documents. It's best to create both gray scale and color versions of the icons and store them in the same section of the __ICON segment. Both versions of the icon would be created at 72 dpi and would be 48 pixels wide by 48 pixels high. The gray-scale version would have two components (gray and alpha), with each component stored at 2 bits. The color version would have four components (red, green, blue, and alpha) and each component would be 4 bits deep. (It's recommended that application and document icons be stored at 4 bits per sample, not 8.)

Compression

NEXTSTEP software can both read and write compressed TIFF images. The Compression field in a TIFF file can have any of the following values:

Value	**Type**

1	No compression
3	CCITT Group 3 compression
4	CCITT Group 4 compression
5	LZW (Lempel-Ziv and Welch) compression
6	JPEG compression
32773	PackBits compression

JPEG compression can be used only for images that have a depth of at least 4 bits per sample; in all cases, the compressed images will be expanded to 8 bits per sample. CCITT Group 3 and Group 4 images can be applied only to monochrome images that have 1 bit per sample.

NXRTFPboardType

This is the pasteboard type for ªrich text,º text that follows the conventions of the Rich Text Format®, as described in *Rich Text Format Specification* by Microsoft Corporation.

To this specification, NeXT has added a control word to indicate how the user selected the text before copying it to the pasteboard. The control word is

\smartcopy<*num*>

where <*num*> can be 1 or 0. A value of 1 indicates that the user made the selection by double-clicking a word, or double-clicking and dragging over a group of words. The range of text in the pasteboard will be delimited by a word boundary on either side. The pasting application can use this information to correctly adjust the spacing around the word or words that are pasted.

NXSoundPboardType

This format is defined by the SNDSoundStruct structure in the header file **sound/soundstruct.h**. The structure is discussed in detail in Chapter 16, ªSound.º

NXFilenamePboardType

This format is a list of tab-separated file names (or pathnames), terminated by a null character (`\0').

NXTabularTextPboardType

This format is ASCII text where tabs (ASCII 0x09) and returns or newlines (ASCII 0x0D) are interpreted as separators between text fields. In a matrix, tabs separate columns and returns separate rows. The text is null-terminated.

NXFontPboardType

This format is used in the font pasteboard to record character properties that are copied and pasted using the Copy Font and Paste Font commands. It consists of RTF control words from the ªFont

Tableº and ªCharacter Formatting Propertiesº groups.

The following is an example of character data in this format:

```
{\rtf1\ansi{\fonttbl\f0\froman Times;}
\f0\b0\i\ul0\fs48}
```

The first two control words, **\rtf1** and **\ansi**, announce that the information enclosed within the outer braces is RTF version 1 in ANSI character encoding. These two control words, or their equivalent, are required by RTF conventions.

The group within the inner braces defines a font table, here with a single entry specifying font 0 to be Times-Roman. The font is then specified as Times-Roman (font 0), not bold, Oblique (italic), not underlined, and having a font size of 24 points (48 half points).

Among the fonts that can be specified in a font table are these:

```
\fmodern Courier;
\fswiss Helvetica;
\fmodern Ohlfs;
\ftech Symbol;
\froman Times;
```

Several synonyms are recognized for the Times-Roman font. Usually it's written as ªTimesº or ªTimes-Romanº.

If the font pasteboard contains RTF control words that don't belong to the ªFont Tableº or ªCharacter Formatting Propertiesº groups, they should be ignored. If control words specify more than one value for a font characteristic, the last value specified should be used when pasting.

NXRulerPboardType

This format is used in the ruler pasteboard to capture information about how a paragraph is formatted. It consists of RTF control words from the ªParagraph Formatting Propertiesº group.

The following is an example of this type:

```
{\rtf1\ansi
\pard\ql\tx1252\tx2716\tx4148\tx5592\tx7004\tx11520
\fi-540\li1260}
```

The first two control words are required by RTF conventions, as explained under ªNXFontPboardTypeº above. The next control word, **\pard**, resets the paragraph format to the default. The paragraph is then specified to be left-aligned and a series of six tabs are set. Next, the indentation of the first line is specified and, finally, the left indent. (The example is for a paragraph with a hanging indent.)

If the ruler pasteboard contains RTF control words that aren't in the ªParagraph Formatting Propertiesº group, they should be ignored. If it includes control words that first set then reset a paragraph property, the final specification should be the one that's used.