«Author» -- «Date»

Package Title (8 blank lines above this, 2 below)

Sometimes it is good to have a general description of the purpose of a set of calls that are documented in one file. This part is the general description, and I also like to have a list of all the calls grouped by logical grouping. The actual function documentation must list the functions alphabetically, so this list at the beginning can serve as a conceptual table of contents. If you want pristine NeXT style function docs, leave out everything from the "Package Title" line up to and including the "Function Descruiptions" line. I like this style better though.

Call	Description
AMSetDatabasePath	Sets the path where the API calls will expect to find the AlertManager database directory.
AMApplicationList	Returns an array of all the applications in the database.
AMApplicationFromName	Returns the application with the given name.
AMGroupList	Returns an array of all the groups in the database.
AMGroupFromName	Returns the group with the given name.
AMStaffList	Returns an array of all the staff members in the database.

AMStaffFromName	Returns the staff member with the given name.
AMAddStaffToGroup	Adds a given staff member to a given group.
AMRemoveStaffFromGroup	Removes a given staff member from a given group.

Data Structures (6 blank lines above this, 2 below)

Include a section like this if there are structures and constants and stuff that are important to the function package as a whole.

```
#define NAME_SIZE      30
#define LOGIN_NAME_SIZE     10
#define ADDRESS_SIZE        80

#define AM_DBSUCCESS        1
#define AM_DBFAIL      0

typedef struct _AMstaff_  {
   INT32          idNum;
   char           name[NAME_SIZE];
   char           login_name[LOGIN_NAME_SIZE];
   char           email[ADDRESS_SIZE];
   unsigned char  alert_types_mask;
} AMstaff;

typedef struct _AMgroup_  {
   INT32          idNum;
```

```
   char           name[NAME_SIZE];
   unsigned char  alert_types_mask;
} AMgroup;

typedef struct _AMapp_  {
   INT32          idNum;
   char           name[NAME_SIZE];
   INT32          num_actions;
} AMapp;
```

Function Descriptions (6 blank lines above this, 2 below)

NXAttachPopUpList(), NXCreatePopUpListButton()

SUMMARY

Set up a pop-up list

DECLARED IN

appkit/PopUpList.h

SYNOPSIS

void **NXAttachPopUpList(**id *button*, PopUpList **popUpList***)*
id **NXCreatePopUpListButton(**PopUpList **popUpList***)*

DESCRIPTION

These functions make it easy to use the PopUpList class. **NXCreatePopUpListButton()** returns a new Button object that will activate the pop-up list specified by *popUpList*. The new Button must then be added to the View hierarchy with View's **addSubview:** method.

NXAttachPopUpList() modifies *button* so that it activates *popUpList*. In addition, if *button* already has a target and an action, then they are used whenever a selection is made from the pop-up list. *button* must be either a Control that uses ButtonCell (or a subclass) as its Cell class, or an actual ButtonCell.

RETURN

NXCreatePopUpListButton() returns a new Button object. (6 blank lines before next function)

NXContainsRect() → See NXMouseInRect() (only 1 blank line between pointing references)

NXConvertCMYKAToColor() → See NXConvertRGBAToColor()

NXConvertCMYKToColor() → See NXConvertRGBAToColor()

NXConvertColorToCMYK() → See NXConvertColorToRGBA()

NXConvertColorToCMYKA() → See NXConvertColorToRGBA()

NXConvertColorToGray() → See NXConvertColorToRGBA()

NXConvertColorToGrayAlpha() → See NXConvertColorToRGBA()

NXConvertColorToHSB() → See NXConvertColorToRGBA()

NXConvertColorToHSBA() → See NXConvertColorToRGBA()

NXConvertColorToRGB() → See NXConvertColorToRGBA() (6 blanks lines before next function)

NXConvertColorToRGBA(), NXConvertColorToCMYKA(), NXConvertColorToHSBA(), NXConvertColorToGrayAlpha(), NXConvertColorToRGB(), NXConvertColorToCMYK(), NXConvertColorToHSB(), NXConvertColorToGray()

SUMMARY

Convert a color value to its standard components

DECLARED IN

appkit/color.h

SYNOPSIS

void **NXConvertColorToRGBA(**NXColor *color*, float **red*, float **green*, float **blue*, float **alpha***)**
void **NXConvertColorToCMYKA(**NXColor *color*, float **cyan*, float **magenta*, float **yellow*, float **black*, float **alpha***)**
void **NXConvertColorToHSBA(**NXColor *color*, float **hue*, float **saturation*, float **brightness*, float **alpha***)**
void **NXConvertColorToGrayAlpha(**NXColor *color*, float **gray*, float **alpha***)**
void **NXConvertColorToRGB(**NXColor *color*, float **red*, float **green*, float **blue***)**
void **NXConvertColorToCMYK(**NXColor *color*, float **cyan*, float **magenta*, float **yellow*, float **black***)**
void **NXConvertColorToHSB(**NXColor *color*, float **hue*, float **saturation*, float **brightness***)**
void **NXConvertColorToGray(**NXColor *color*, float **gray***)**

DESCRIPTION

These functions convert a color value, *color*, to its standard components. The first argument to each function is the NXColor data structure to be converted. Subsequent arguments point to **float** variables where the component values can be returned by reference.

The conversion can be to any set of components that might be used to specify a color value:

- Red, green, and blue (RGB) components
- Cyan, magenta, yellow, and black (CMYK) components
- Hue, saturation, and brightness (HSB) components
- A single component for gray scale images

A color initially specified by one set of components can be converted to another set. For example:

```
NXColor      color;
float        hue, saturation, brightness;

color = NXConvertRGBToColor(0.8, 0.3, 0.15);
NXConvertColorToHSB(color, &hue, &saturation, &brightness);
```

The first four functions in the list above report the coverage component, *alpha*, included in the color value, as well as the color components. The second four report only the color components; they're macros and are defined on the corresponding functions, but ignore the *alpha* argument.

The **float** values returned by reference will lie in the range 0.0 through 1.0. The value returned for the coverage component will be NX_NOALPHA if *color* doesn't include a coverage specification.

SEE ALSO

NXConvertRGBAToColor(), **NXSetColor()**, **NXEqualColor()**, **NXRedComponent()**, **NXChangeRedComponent()**, **NXReadColor()**