



PROGRAMMING INTERFACE SUMMARY



NEXTSTEPTM

Object - Oriented Software

NeXTSTEP™ PROGRAMMING INTERFACE SUMMARY

NeXTSTEP Developer's Library
NeXT Computer, Inc.

Release 3



Addison-Wesley Publishing Company

Reading, Massachusetts • Menlo Park, California • New York • Don Mills, Ontario
Wokingham, England • Amsterdam • Bonn • Sydney • Singapore • Tokyo • Madrid
San Juan • Paris • Seoul • Milan • Mexico City • Taipei

NeXT and the publishers have tried to make the information contained in this manual as accurate and reliable as possible, but assume no responsibility for errors or omissions. They disclaim any warranty of any kind, whether express or implied, as to any matter whatsoever relating to this manual, including without limitation the merchantability or fitness for any particular purpose. In no event shall NeXT or the publishers be liable for any indirect, special, incidental, or consequential damages arising out of purchase or use of this manual or the information contained herein. NeXT will from time to time revise the software described in this manual and reserves the right to make such changes without obligation to notify the purchaser.

NeXTSTEP Programming Interface Summary Copyright © 1990–1992 by NeXT Computer, Inc. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher or copyright owner. Printed in the United States of America. Published simultaneously in Canada.

NeXTSTEP 3.0 Copyright © 1988–1992 by NeXT Computer, Inc. All rights reserved. Certain portions of the software are copyrighted by third parties. U.S. Pat. No. 4,982,343. Other Patents Pending.

NeXT, the NeXT logo, NeXTSTEP, Application Kit, Database Kit, Indexing Kit, Interface Builder, Mach Kit, NetInfo, NetInfo Kit, Phone Kit, 3D Graphics Kit, and Workspace Manager are trademarks of NeXT Computer, Inc. PostScript and Display PostScript are registered trademarks of Adobe Systems, Incorporated. Novell and NetWare are registered trademarks of Novell, Inc. UNIX is a registered trademark of UNIX Systems Laboratories, Inc. All other trademarks mentioned belong to their respective owners.

Restricted Rights Legend: Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 [or, if applicable, similar clauses at FAR 52.227-19 or NASA FAR Supp. 52.227-86].

This manual describes NeXTSTEP Release 3.

Written by NeXT Publications.

This manual was designed, written, and produced on NeXT computers. Proofs were printed on a NeXT 400 dpi Laser Printer and NeXT Color Printer. Final pages were transferred directly from a NeXT optical disk to film using NeXT computers and an electronic imagesetter.

3 4 5 6 7 8 9 10–CRS–96959493
Third printing, November 1993

ISBN 0-201-63253-5

Contents

Introduction

1-1 Chapter 1: Root Class

1-1 Class

Object, p. 1-1.

1-5 Types and Constants

Defined Types, p. 1-5. Symbolic Constants, p. 1-6.

2-1 Chapter 2: Application Kit

2-1 Classes

ActionCell, p. 2-1. Application, p. 2-2. Box, p. 2-10. Button, p. 2-11. ButtonCell, p. 2-13. Cell, p. 2-17. ClipView, p. 2-21. Control, p. 2-23. Font, p. 2-26. FontManager, p. 2-27. FontPanel, p. 2-29. Form, p. 2-30. FormCell, p. 2-33. Listener, p. 2-34. Matrix, p. 2-36. Menu, p. 2-42. MenuCell, p. 2-44. NXBitmapImageRep, p. 2-45. NXBrowser, p. 2-47. NXBrowserCell, p. 2-52. NXCachedImageRep, p. 2-53. NXColorPanel, p. 2-54. NXColorPicker, p. 2-55. NXColorWell, p. 2-56. NXCursor, p. 2-57. NXCustomImageRep, p. 2-58. NXEPSImageRep, p. 2-58. NXHelpPanel, p. 2-60. NXImage, p. 2-61. NXImageRep, p. 2-64. NXJournaler, p. 2-66. NXPrinter, p. 2-67. NXSpellChecker, p. 2-68. NXSpellServer, p. 2-69. NXSplitView, p. 2-71. Object Additions, p. 2-72. OpenPanel, p. 2-72. PageLayout, p. 2-73. Panel, p. 2-74. Pasteboard, p. 2-75. PopUpList, p. 2-77. PrintInfo, p. 2-78. PrintPanel, p. 2-80. Responder, p. 2-81. SavePanel, p. 2-83. Scroller, p. 2-84. ScrollView, p. 2-86. SelectionCell, p. 2-88. Slider, p. 2-89. SliderCell, p. 2-90. Speaker, p. 2-92. Text, p. 2-95. TextField, p. 2-104. TextFieldCell, p. 2-106. View, p. 2-107. Window, p. 2-114.

- 2-123 Protocols
 - NXDraggingDestination, p. 2-123. NXDraggingInfo, p. 2-123.
 - NXDraggingSource, p. 2-124. NXNibNotification, p. 2-125.
 - NXPrintingUserInterface, p. 2-125. NXRTFErrorHandler, p. 2-125.
 - NXServicesRequests, p. 2-126.
- 2-127 Application Kit Functions
 - Rectangle Functions, p. 2-127. Color Functions, p. 2-128.
 - Text Functions, p. 2-130. Imaging Functions, p. 2-131.
 - Object Management Functions, p. 2-132. Error-Handling Functions, p. 2-133. Typed Stream Functions, p. 2-133.
 - Remote Messaging Functions, p. 2-133. Services Menu Functions, p. 2-134.
 - Event Function, p. 2-134. Memory Allocation Functions, p. 2-134.
 - Other Application Kit Functions, p. 2-135.
- 2-137 Types and Constants
 - Defined Types, p. 2-137. Symbolic Constants, p. 2-153.
 - Global Variables, p. 2-165.
- 3-1 Chapter 3: Common Classes and Functions**
- 3-1 Classes
 - HashTable, p. 3-1. List, p. 3-2. NXBundle, p. 3-4. NXStringTable, p. 3-5.
 - Storage, p. 3-5.
- 3-7 Functions
 - Character Classification Functions, p. 3-7. Defaults System Functions, p. 3-7.
 - Error-Handling Functions, p. 3-8. Stream Functions, p. 3-8.
 - Typed Stream Functions, p. 3-10. Memory Allocation Functions, p. 3-11.
 - Hash and String Table Functions, p. 3-12. String Functions, p. 3-13.
 - Miscellaneous Functions, p. 3-13.
- 3-14 Types and Constants
 - Defined Types, p. 3-14. Symbolic Constants, p. 3-15.
 - Global Variables, p. 3-16.

4-1 Chapter 4: Database Kit

4-1 Classes

DBAssociation, p. 4-1. DBBinder, p. 4-2. DBDatabase, p. 4-6.
DBEditableFormatter, p. 4-8. DBExpression, p. 4-10.
DBFetchGroup, p. 4-10. DBFormatter, p. 4-12. DBImageFormatter, p. 4-14.
DBImageView, p. 4-15. DBModule, p. 4-15. DBQualifier, p. 4-17.
DBRecordList, p. 4-18. DBRecordStream, p. 4-20. DBTableVector, p. 4-21.
DBTableView, p. 4-22. DBTextFormatter, p. 4-26. DBValue, p. 4-27.

4-28 Protocols

DBContainers, p. 4-28.DBCursorPositioning, p. 4-28.
DBCUSTOMAssociation, p. 4-29. DBEntities, p. 4-30.
DBExpressionValues, p. 4-30. DBFormatConversion, p. 4-31.
DBFormatInitialization, p. 4-31. DBFormatterValidation, p. 4-31.
DBFormatterViewEditing, p. 4-32. DBProperties, p. 4-33.
DBTableDataSources, p. 4-33. DBTableValues, p. 4-34.
DBTableVectors, p. 4-35. DBTransactions, p. 4-36. DBTypes, p. 4-36.

4-37 Types and Constants

Defined Types, p. 4-37. Symbolic Constants, p. 4-39.

5-1 Chapter 5: Display PostScript

5-1 PostScript Operators

5-24 Single-Operator Functions

5-40 Client Library Functions

Controlling a PostScript Execution Context, p. 5-40. Sending Data to the
Window Server, p. 5-41. User Objects and User Names, p. 5-41.
Event-Handling, p. 5-42. File and Port Monitoring, p. 5-43.
Text-Handling, p. 5-43. Debugging and Error-Handling, p. 5-43.
Functions Used by **pswrap**, p. 5-44.

5-45 Types and Constants

Defined Types, p. 5-45. Symbolic Constants, p. 5-49.

6-1 Chapter 6: Distributed Objects

6-1 Classes

NXConnection, p. 6-1. NXProxy, p. 6-4. Object Additions, p. 6-4.

6-5 Protocols

NXDecoding, p. 6-5. NXEncoding, p. 6-5. NXTransport, p. 6-6.

6-7 Types and Constants

Defined Types, p. 6-7. Symbolic Constants, p. 6-7.

7-1 Chapter 7: Indexing Kit

7-1 Classes

IXAttributeParser, p. 7-1. IXAttributeQuery, p. 7-2.

IXAttributeReader, p. 7-3. IXBTree, p. 7-4. IXBTreeCursor, p. 7-5.

IXFileFinder, p. 7-6. IXFileRecord, p. 7-6. IXLanguageReader, p. 7-7.

IXPostingCursor, p. 7-8. IXPostingList, p. 7-8. IXPostingSet, p. 7-10.

IXRecordManager, p. 7-11. IXStore, p. 7-13. IXStoreBlock, p. 7-14.

IXStoreDirectory, p. 7-15. IXStoreFile, p. 7-16.

IXWeightingDomain, p. 7-17.

7-18 Protocols

IXAttributeReading, p. 7-18. IXBlobWriting, p. 7-18.

IXBlockAndStoreAccess, p. 7-19. IXComparatorSetting, p. 7-19.

IXComparisonSetting, p. 7-20. IXCursorPositioning, p. 7-20.

IXFileFinderConfiguration, p. 7-21. IXFileFinderQueryAndUpdate, p. 7-22.

IXLexemeExtraction, p. 7-23. IXNameAndFileAccess, p. 7-24.

IXPostingExchange, p. 7-24. IXPostingOperations, p. 7-25.

IXRecordDiscarding, p. 7-26. IXRecordReading, p. 7-26.

IXRecordTranscription, p. 7-27. IXRecordWriting, p. 7-27.

IXTransientAccess, p. 7-28. IXTransientMessaging, p. 7-29.

7-30 Functions

7-32 Types and Constants

Defined Types, p. 7-32. Symbolic Constants, p. 7-33.

Global Variables, p. 7-33.

7-34 Query Language Symbols and Operators

7-37 Attribute Reader Format

8-1 Chapter 8: Interface Builder

8-1 Classes

IBInspector, p. 8-1. IBPalette, p. 8-2. Object Additions, p. 8-3.
View Additions, p. 8-3.

8-4 Protocols

IB, p. 8-4. IBConnectors, p. 8-5. IBDocuments, p. 8-5.
IBDocumentControllers, p. 8-7. IBEditors, p. 8-7. IBInspectors, p. 8-8.
IBObject, p. 8-9. IBSelectionOwners, p. 8-9.

8-10 Types and Constants

Symbolic Constants, p. 8-10. Global Variables, p. 8-10.

9-1 Chapter 9: Mach Kit

9-1 Classes

NXConditionLock, p. 9-1. NXData, p. 9-2. NXInvalidationNotifier, p. 9-3.
NXLock, p. 9-4. NXNetNameServer, p. 9-4. NXPort, p. 9-5.
NXProtocolChecker, p. 9-6. NXRecursiveLock, p. 9-6. NXSpinLock, p. 9-7.

9-8 Protocols

NXLock, p. 9-8. NXReference, p. 9-8. NXSenderIsInvalid, p. 9-9.

9-10 Types and Constants

Defined Types, p. 9-10.

10-1 Chapter 10: MIDI Driver API

10-1 Driver Functions

Clock Functions, p. 10-1. Data Sending Function, p. 10-2.
Driver Ownership Functions, p. 10-2. Ignore MIDI Codes Function, p. 10-2.
Queue Management Functions, p. 10-2. Reply Handling Functions, p. 10-3.
Request Functions, p. 10-3. Serial Port Ownership Functions, p. 10-3.

10-4 Types and Constants

Defined Types, p. 10-4. Symbolic Constants, p. 10-5.

11-1 Chapter 11: NetInfo Kit

11-1 Classes

NIDomain, p. 11-1. NIDomainPanel, p. 11-2. NILoginPanel, p. 11-4.
NIOpenPanel, p. 11-5. NISavePanel, p. 11-6.

11-7 Functions

11-8 Types and Constants

Symbolic Constants, p. 11-8. Structures, p. 11-8.

12-1 Chapter 12: Networks: Novell NetWare

13-1 Chapter 13: Phone Kit

13-1 Classes

NXPhone, p. 13-1. NXPhoneCall, p. 13-2. NXPhoneChannel, p. 13-4.

13-5 Functions

Error Function, p. 13-5.

13-6 Types and Constants

Defined Types, p. 13-6.

14-1 Chapter 14: Preferences

14-1 Classes

Application Additions, p. 14-1. Layout, p. 14-2.

14-3 Types and Constants

Symbolic Constants, p. 14-3.

15-1 Chapter 15: The Run-Time System

15-1 Classes

Protocol, p. 15-1.

15-2 Functions

Class Functions, p. 15-2. System Functions, p. 15-3.
Object Functions, p. 15-4. Method Functions and Macros, p. 15-4.
Selector Functions, p. 15-5.

15-6 Types and Constants

Defined Types, p. 15-6. Symbolic Constants, p. 15-7. Structures, p. 15-7.
Global Variables, p. 15-10

16-1 Chapter 16: Sound

16-1 Classes

NXPlayStream, p. 16-1. NXRecordStream, p. 16-2.
NXSoundDevice, p. 16-3. NXSoundIn, p. 16-5. NXSoundOut, p. 16-5.
NXSoundStream, p. 16-6. Sound, p. 16-8. SoundMeter, p. 16-11.
SoundView, p. 16-12.

16-16 Sound Functions

Accessing Sound Devices and Hardware, p. 16-16.
Recording and Playing, p. 16-16. Reading and Writing Soundfiles, p. 16-17.
Accessing Sound Data, p. 16-17. Accessing the DSP, p. 16-18.
Compressing Sound Data, p. 16-18. Converting Sound Data, p. 16-19.
Editing Sound Data, p. 16-20. Sound Errors, p. 16-20.

16-21 Driver Functions

DSP Functions, p. 16-21. Driver Setup and Access, p. 16-22.
Stream Setup and Access, p. 16-23.

16-24 Types and Constants

Defined Types, p. 16-24. Symbolic Constants, p. 16-27. Global
Variables, p. 16-30.

17-1 Chapter 17: 3D Graphics Kit

17-1 Classes

N3DCamera, p. 17-1. N3DContextManager, p. 17-6. N3DLight, p. 17-7.
N3DMovieCamera, p. 17-8. N3DRenderPanel, p. 17-9.
N3DRIBImageRep, p. 17-10. N3DRotator, p. 17-11. N3DShader, p. 17-12.
N3DShape, p. 17-14.

17-18 Functions

Data Component Functions, p. 17-18. Data Conversion Functions, p. 17-18.
Data Copying Functions, p. 17-18. Intersection Testing Function, p. 17-18.
Matrix Manipulation Functions, p. 17-19. Transformation Functions, p. 17-19.

17-20 Types and Constants

Defined Types, p. 17-20. Symbolic Constants, p. 17-21.
Global Variables, p. 17-22.

18-1 Chapter 18: Video

18-1 Classes

NXLiveVideoView, p. 18-1.

18-4 Types and Constants

Symbolic Constants, p. 18-4.

19-1 Chapter 19: Workspace Manager

19-1 Class

WMInspector, p. 19-1.

20-1 Chapter 20: Mach Functions

20-1 C-Thread Functions

Basic C-Thread Functions, p. 20-1. Mutex Functions, p. 20-2.

Condition Functions, p. 20-3.

20-4 Mach Kernel Functions

Task Functions, p. 20-4. Thread Functions, p. 20-5. Port Functions, p. 20-6.

Message Functions, p. 20-7. Virtual Memory Functions, p. 20-8. Host

Functions, p. 20-8. Processor Functions, p. 20-9. Exception

Functions, p. 20-10. Error String Functions, p. 20-10.

20-11 Network Name Server Functions

20-12 Bootstrap Server Functions

20-13 Kernel-Server Loader Functions

Introduction

NeXTSTEP™ Programming Interface Summary provides the syntax of all programming elements in the NeXTSTEP environment. It's intended to be used as a quick reference companion to the *NeXTSTEP General Reference*; everything that's described in the General Reference is summarized here.

The organization of this manual parallels that of *NeXTSTEP General Reference*. It starts with the root class: the Object class that is at the root of the inheritance tree of NeXTSTEP classes. Each of the specialized kits has a chapter to itself, including Application Kit™, Database Kit™, Indexing Kit™, Mach Kit™, NetInfo Kit™, Phone Kit™, and 3D Graphics Kit™. Three chapters deal with the programmatic interface to NeXT applications: Interface Builder™, Workspace Manager™, and Preferences. Three chapters treat topics that are not considered to be kits: sound, video, and MIDI. Five chapters are devoted to general facilities: the common classes and functions, Display PostScript®, the run-time system, distributed objects, and functions that invoke the Mach operating system.

This book is part of a collection of manuals called the NeXTSTEP Developer's Library.

Syntax Notation

In these summaries, bold and italic have the following significance:

- **Bold** denotes a word or character that is to be taken literally (typed as it appears).
- *Italic* denotes a placeholder for an element—typically an argument name—that you supply.



1 *Root Class*

Class

Object

Inherits From: Object is the root class

Initializing the Class

+ **initialize**

Implemented by subclasses to initialize the class

Creating, Copying, and Freeing Instances

+ **alloc**

Returns a new, uninitialized instance

+ **allocFromZone:(NXZone *)zone**

Returns a new, uninitialized instance allocated from *zone*

+ **new**

Returns a new, initialized instance

- **copy**

Returns an exact copy of the receiver

- **copyFromZone:(NXZone *)zone**

Returns an exact copy of the receiver allocated from *zone*

- **(NXZone *)zone**

Returns a pointer to the zone where the receiver resides

- **free**

Deallocates the memory occupied by the receiver

+ **free**

Returns **nil** (you can't free a class)

Initializing a New Instance

- **init** Initializes a new instance after it has been allocated

Identifying Classes

- + (const char *)**name** Returns the name of the class
- + **class** Returns the receiver, a class object
- **class** Returns the class object for the receiver's class
- + **superclass** Returns the class object for the receiver's superclass
- **superclass** Returns the class object for the receiver's superclass

Identifying and Comparing Instances

- (BOOL)**isEqual:***aObject* Returns whether the receiver and *aObject* are the same
- (unsigned int)**hash** Returns an unsigned integer unique to the receiver
- **self** Returns the receiver, an instance
- (const char *)**name** Implemented by subclasses to return the receiver's name
- (void)**printForDebugger:**(NXStream *)*stream* Writes information identifying the receiver to *stream*

Testing Inheritance Relationships

- (BOOL)**isKindOf:***aClassObject* Returns whether the receiver inherits from *aClassObject*
- (BOOL)**isKindOfClassNamed:**(const char *)*aClassName* Returns whether the receiver inherits from *aClassName*
- (BOOL)**isMemberOf:***aClassObject* Returns whether the receiver is an instance of *aClassObject*
- (BOOL)**isMemberOfClassNamed:**(const char *)*aClassName* Returns whether the receiver is an instance of *aClassName*

Testing Class Functionality

- (BOOL)**respondsTo:**(SEL)*aSelector* Returns whether the receiver can respond to *aSelector*
- + (BOOL)**instancesRespondTo:**(SEL)*aSelector* Returns whether instances can respond to *aSelector*

Testing for Protocol Conformance

- + (BOOL)**conformsTo:**(Protocol *)*aProtocol* Returns whether the receiver conforms to *aProtocol*
- (BOOL)**conformsTo:**(Protocol *)*aProtocol* Returns whether the receiver's class conforms to *aProtocol*

Sending Messages Determined at Run Time

- **perform:(SEL)aSelector** Sends an *aSelector* message to the receiver
- **perform:(SEL)aSelector with:anObject** Sends an *aSelector* message with one argument
- **perform:(SEL)aSelector with:anObject with:anotherObject** Sends an *aSelector* message with two arguments

Forwarding Messages

- **forward:(SEL)aSelector :(marg_list)argFrame** Implemented by subclasses to forward messages
- **performv:(SEL)aSelector :(marg_list)argFrame** Sends an *aSelector* message with *argFrame* arguments

Obtaining Method Information

- (IMP)**methodFor:(SEL)aSelector** Locates the receiver’s implementation of *aSelector*
- + (IMP)**instanceMethodFor:(SEL)aSelector** Locates the implementation of *aSelector*
- (struct objc_method_description *)**descriptionForMethod:(SEL)aSelector** Returns information about the *aSelector* method
- + (struct objc_method_description *)**descriptionForInstanceMethod:(SEL)aSelector** Returns information about the *aSelector* instance method

Posing

- + **poseAs:aClassObject** Substitutes the receiving class for *aClassObject*

Enforcing Intentions

- **notImplemented:(SEL)aSelector** Indicates that *aSelector* isn’t fully implemented
- **subclassResponsibility:(SEL)aSelector** Generates an error if *aSelector* isn’t implemented

Error Handling

- **doesNotRecognize:(SEL)aSelector** Generates an unrecognized-selector error message
- **error:(const char *)aString, ...** Generates a formatted error message using *aString*

Dynamic Loading

- + **finishLoading:(struct mach_header *)header** Implemented by a newly loaded class or category
- + **startUnloading** Implemented by a class or category about to be unloaded

Archiving

- **read:**(NXTypedStream *)*stream*
- **write:**(NXTypedStream *)*stream*
- **startArchiving:**(NXTypedStream *)*stream*
- **awake**
- **finishUnarchiving**
- + **setVersion:**(int)*aVersion*
- + (int)**version**

Implemented by subclasses to read receiver from *stream*

Implemented by subclasses to write the receiver to *stream*

Implemented by subclasses to prepare for archiving

Implemented by subclasses to reinitialize the receiver

Implemented by subclasses to replace the receiver

Sets the class version number to *aVersion*

Returns the version of the class definition

Types and Constants

Defined Types

BOOL

```
typedef char BOOL;
```

Class

```
typedef struct objc_class *Class;
```

id

```
typedef struct objc_object {  
    Class isa;  
} *id;
```

IMP

```
typedef id (*IMP) (id, SEL, ...);
```

SEL

```
typedef struct objc_selector *SEL;
```

STR

```
typedef char *STR;
```

Symbolic Constants

Boolean Constants

YES	(BOOL)1
NO	(BOOL)0

Empty Objects

nil	(id)0
Nil	(Class)0

2 *Application Kit*

Classes

ActionCell

Inherits From: Cell : Object

Configuring an ActionCell

- | | |
|--|---|
| – setEnabled: (BOOL) <i>flag</i> | Sets whether the ActionCell reacts to mouse events |
| – setBezeled: (BOOL) <i>flag</i> | Adds or removes the ActionCell's bezel |
| – setBordered: (BOOL) <i>flag</i> | Adds or removes the ActionCell's border |
| – setAlignment: (int) <i>mode</i> | Sets the ActionCell's text alignment to <i>mode</i> |
| – setFloatingPointFormat: (BOOL) <i>autoRange</i>
left: (unsigned int) <i>leftDigits</i>
right: (unsigned int) <i>rightDigits</i> | Sets the ActionCell's floating point format |
| – setFont: <i>fontObject</i> | Sets the ActionCell's Font to <i>fontObject</i> |
| – setIcon: (const char *) <i>iconName</i> | Sets the ActionCell's icon to the NXImage named <i>iconName</i> |

Manipulating ActionCell Values

- | | |
|-------------------------------|--|
| – (double) doubleValue | Returns the ActionCell's contents as a double |
| – (float) floatValue | Returns the ActionCell's contents as a float |

- (int)intValue
- setStringValue:(const char *)aString
- setStringValueNoCopy:(char *)aString
shouldFree:(BOOL)flag
- (const char *)stringValue

Returns the ActionCell's contents as an **int**
 Sets the ActionCell's contents to a copy of *aString*
 Sets the ActionCell's contents to a *aString*; will free the string when freed if *flag* is YES
 Returns the ActionCell's contents as a string

Displaying

- drawSelf:(const NXRect *)cellFrame
inView:controlView
- controlView

Draws the ActionCell in *controlView*
 Returns the View in which the ActionCell was most recently drawn (usually a Control)

Target and Action

- setAction:(SEL)aSelector
- (SEL)action
- setTarget:anObject
- target

Sets the ActionCell's action method to *aSelector*
 Returns the ActionCell's action method
 Sets the ActionCell's target object to *anObject*
 Returns the ActionCell's target object

Assigning a Tag

- setTag:(int)anInt
- (int>tag

Sets the ActionCell's tag to *anInt*
 Returns the ActionCell's tag

Archiving

- read:(NXTypedStream *)stream
- write:(NXTypedStream *)stream

Reads the ActionCell from *stream*
 Writes the ActionCell to *stream*

Application

Inherits From: Responder : Object

Initializing the Class Object

- + initialize Registers defaults for the application

Creating and Freeing Instances

- + **new** Returns a new Application object
- **free** Deallocates the Application object

Setting Up the Application

- + **workspace** Returns the **id** of WorkspaceManager
- **loadNibFile:(const char *)filename
owner:anObject** Loads objects built using Interface Builder
- **loadNibFile:(const char *)filename
owner:anObject
withNames:(BOOL)flag** Loads objects built using Interface Builder
- **loadNibFile:(const char *)filename
owner:anObject
withNames:(BOOL)flag
fromZone:(NXZone *)zone** Loads objects built using Interface Builder
- **loadNibSection:(const char *)name
owner:anObject** Loads objects built using Interface Builder
- **loadNibSection:(const char *)name
owner:anObject
withNames:(BOOL)flag** Loads objects built using Interface Builder
- **loadNibSection:(const char *)name
owner:anObject
withNames:(BOOL)flag
fromHeader:(const struct mach_header *)header** Loads objects built using Interface Builder
- **loadNibSection:(const char *)name
owner:anObject
withNames:(BOOL)flag
fromZone:(NXZone *)zone** Loads objects built using Interface Builder
- **loadNibSection:(const char *)name
owner:anObject
withNames:(BOOL)flag
fromHeader:(const struct mach_header *)header
fromZone:(NXZone *)zone** Loads objects built using Interface Builder
- **(const char *)appName** Returns the application's name
- **setMainMenu:aMenu** Makes *aMenu* the application's main menu
- **mainMenu** Returns the **id** of the application's main menu

Responding to Notification (Override in a Subclass)

- **applicationDidLaunch:***appName* Notice that *appName* launched; your arbitrary return
- **applicationDidTerminate:***appName* Notice that *appName* ended; your arbitrary return
- **applicationWillLaunch:***appName* Notice that *appName* will launch; your arbitrary return

Changing the Active Application

- (int)**activate:**(int)*contextNumber* Makes *contextNumber* the active application
- (int)**activateSelf:**(BOOL)*flag* Makes this the active application
- (int)**activeApp** Returns context number of the active application
- **becomeActiveApp** Responds to activating the application
- **deactivateSelf** Deactivates the application
- (BOOL)**isActive** Returns whether this is the active application
- **resignActiveApp** Responds to deactivating the application

Running the Event Loop

- **run** Starts the main event loop
- **stop:***sender* Stops the main event loop
- (int)**runModalFor:***theWindow* Starts a modal event loop for *theWindow*
- **stopModal** Stops the modal event loop
- **stopModal:**(int)*returnCode* Stops the event loop started by **runModalFor:**
- (void)**abortModal** Aborts the event loop started by **runModalFor:**
- (NXModalSession *)**beginModalSession:**(NXModalSession *)*session*
for:*theWindow* Sets up a modal session with *theWindow*
- (int)**runModalSession:**(NXModalSession *)*session* Runs a modal session
- **endModalSession:**(NXModalSession *)*session* Finishes a modal session
- **delayedFree:***theObject* Frees *theObject* after finishing the current event
- (BOOL)**isRunning** Returns whether the main event loop is running
- **sendEvent:**(NXEvent *)*theEvent* Dispatches events to other objects

Getting and Peeking at Events

- (NXEvent *)**currentEvent** Returns pointer to the current event
- (NXEvent *)**getNextEvent:**(int)*mask* Returns pointer to the next event matching *mask*
- (NXEvent *)**getNextEvent:**(int)*mask*
waitFor:(double)*timeout*
threshold:(int)*level* Returns pointer to the next event matching *mask*

- (NXEvent *)**peekAndGetNextEvent:(int)mask** Returns pointer to the next event matching *mask*
- (NXEvent *)**peekNextEvent:(int)mask**
into:(NXEvent *)eventPtr Returns pointer to the next event matching *mask*
- (NXEvent *)**peekNextEvent:(int)mask**
into:(NXEvent *)eventPtr
waitFor:(float)timeout
threshold:(int)level Returns pointer to the next event matching *mask*

Journaling

- (BOOL)**isJournalable** Returns whether the application can be journaled
- **setJournalable:(BOOL)flag** Sets whether the application can be journaled
- **masterJournaler** Returns the controlling NXJournaler object
- **slaveJournaler** Returns the controlled NXJournaler object

Handling User Actions and Events

- **applicationDefined:(NXEvent *)theEvent** Responds to an application-defined event
- **hide:sender** Hides all the application's windows
- (BOOL)**isHidden** YES if windows are hidden
- (int)**unhide** Responds to message to unhide windows
- **unhide:sender** Restores hidden windows to the screen
- **unhideWithoutActivation:sender** Restores hidden windows without activating their owner
- **powerOff:(NXEvent *)theEvent** Responds to a power-off subevent
- (int)**powerOffIn:(int)ms andSave:(int)aFlag** Responds to message from Workspace Manager
- **rightMouseDown:(NXEvent *)theEvent** Causes main menu to pop up under the mouse
- (int)**unmounting:(const char *)fullPath**
ok:(int *)flag Responds to message from Workspace Manager

Sending Action Messages

- (BOOL)**sendAction:(SEL)aSelector**
to:aTarget
from:sender Sends an action message to *aTarget* or up the responder chain
- (BOOL)**tryToPerform:(SEL)aSelector**
with:anObject Attempts to send a message to the application or the delegate
- **calcTargetForAction:(SEL)theAction** Looks up receiver for *theAction* message

Remote Messaging

- **setAppListener:***aListener* Makes *aListener* the application’s Listener
- **appListener** Returns the Listener for this application
- **setAppSpeaker:***aSpeaker* Makes *aSpeaker* the application’s Speaker
- **appSpeaker** Returns the Speaker for this application
- **(const char *)appListenerPortName** Returns name used to register Listener with name server
- **(port_t)replyPort** Returns port for synchronous return messages

Managing Windows

- **appIcon** Returns the Window with the application’s icon
- **findWindow:(int>windowNum** Returns Window object corresponding to *windowNum*
- **getWindowNumbers:(int **)list**
 count:(int *)winCount Gets window numbers for the application’s windows
- **keyWindow** Returns the the key window
- **mainWindow** Returns the main window
- **makeWindowsPerform:(SEL)aSelector**
 inOrder:(BOOL)flag Sends *aSelector* message to the Windows
- **setAutoupdate:(BOOL)flag** Sets whether to send windows **update** messages
- **updateWindows** Sends **update** message to all on-screen Windows
- **windowList** Returns a List of the application’s Windows
- **miniaturizeAll:** Miniaturizes all the receiver’s application windows
- **preventWindowOrdering** Suppresses usual window ordering entirely

Managing the Windows Menu

- **setWindowsMenu:***aMenu* Sets the Windows menu
- **windowsMenu** Returns the Windows menu
- **arrangeInFront:** Orders all registered Windows to the front
- **addWindowsItem:***aWindow*
 title:(const char *)aString
 filename:(BOOL)isFilename Adds a menu item for *aWindow*
- **removeWindowsItem:***aWindow* Removes the Window’s menu item
- **changeWindowsItem:***aWindow*
 title:(const char *)aString
 filename:(BOOL)isFilename Changes the Window’s menu item
- **updateWindowsItem:***aWindow* Updates the Window’s menu item

Managing Panels

- **showHelpPanel:***sender* Shows the application’s help panel or default
- **orderFrontDataLinkPanel:** Shows the shared instance; creates if need be

Managing the Services menu

- **setServicesMenu:***aMenu* Sets the Services menu
- **servicesMenu** Returns the Services menu
- **registerServicesMenuSendTypes:**(const char *const *)*sendTypes*
 andReturnTypes:(const char *const *)*returnTypes*
Registers pasteboard types the application can send/receive
- **validRequestorForSendType:**
 (NXAtom)*sendType*
 andReturnTypes:(NXAtom)*returnType*
Indicates whether the Application can send and receive the specified types

Managing Screens

- (const NXScreen *)**mainScreen** Returns the main screen
- (const NXScreen *)**colorScreen** Returns the best screen for color
- **getScreens:**(const NXScreen **)*list*
 count:(int *)*numScreens* Gets information about every connected screen
- **getScreenSize:**(NXSize *)*theSize* Provides the size of the screen in pixels

Querying the Application

- (DPSContext)**context** Returns the Application’s DPS context
- **focusView** Gets the currently lockFocus’ed View
- (const char *)**hostName** Returns machine running the Window Server

Reporting Current Languages

- (const char *const *)**systemLanguages** Gets a list of the user’s preferred languages

Opening Files

- (int)**openFile:**(const char *)*fullPath*
 ok:(int *)*flag* Asks the delegate to open the *fullPath* file
- (int)**openTempFile:**(const char *)*fullPath*
 ok:(int *)*flag* Asks the delegate to open a temporary file
- (int)**fileOperationCompleted:**(int)*operation* Notification of completion of NXWorkspaceRequest; arbitrary *operation* and return

Responding to Devices

- (int)mounted:(const char *)fullPath Notice that device at fullPath has been mounted; returns 0 (unless overridden in a subclass)
- unmounted:(const char *)fullPath Notice that device at fullPath has been unmounted; returns 0 (unless overridden in a subclass)

Printing

- setPrintInfo:info Makes info the application's PrintInfo object
- printInfo Returns the application's PrintInfo object
- runPageLayout:sender Runs the application's PageLayout panel

Color

- orderFrontColorPanel:sender Brings up the color panel
- doesImportAlpha YES if application responds to opacity in imported colors
- setImportAlpha: Enable/disable response to opacity in imported colors

Terminating the Application

- terminate:sender Frees the Application object and exits the application

Assigning a Delegate

- setDelegate:anObject Makes anObject the Application's delegate
- delegate Returns the Application's delegate

Implemented by the Delegate

- app:sender Notice that appName will launch†
applicationWillLaunch:(const char *)appName
- app:sender Notice that appName launched†
applicationDidLaunch:(const char *)appName
- appWillInit:sender Notifies delegate before initializing†
- appDidInit:sender Notifies delegate before getting first event†

– appDidBecomeActive: <i>sender</i>	Notifies delegate on activating the application
– appDidResignActive: <i>sender</i>	Notifies delegate on deactivating the application
– appDidHide: <i>sender</i>	Notifies delegate application has been hidden
– appDidUnhide: <i>sender</i>	Notifies delegate application has been unhidden
– appWillUpdate: <i>sender</i>	Notifies delegate application’s windows will be updated
– appDidUpdate: <i>sender</i>	Notifies delegate on updating the application’s windows
– (BOOL) appAcceptsAnotherFile: <i>sender</i>	YES if it’s okay to open another file†
– (int) app: <i>sender</i> openFile: (const char *) <i>filename</i> type: (const char *) <i>aType</i>	Opens <i>filename</i> †
– (int) app: <i>sender</i> openTempFile: (const char *) <i>filename</i> type: (const char *) <i>aType</i>	Opens temporary file <i>filename</i> †
– (NXDataLinkManager *) app: <i>sender</i> openFileWithoutUI: (const char *) <i>filename</i> type: (const char *) <i>aType</i>	Open application to run without user interface†
– app: <i>sender</i> fileOperationCompleted: (int) <i>operation</i>	Notice that <i>operation</i> completed; your arbitrary return†
– app: <i>sender</i> mounted: (const char *) <i>fullPath</i>	Notification that device at <i>fullPath</i> was mounted†
– (int) app: <i>sender</i> unmounting: (const char *) <i>fullPath</i>	Facilitates unmounting a device†
– app: <i>sender</i> unmounted: (const char *) <i>fullPath</i>	Notification that device at <i>fullPath</i> was unmounted†
– appWillTerminate: <i>sender</i>	Notification that application will terminate†
– app: <i>sender</i> willShowHelpPanel: <i>panel</i>	Notification that <i>sender</i> will show <i>panel</i>
– app: <i>sender</i> powerOffIn: (int) <i>ms</i> andSave: (int) <i>aFlag</i>	Responds to message from Workspace Manager†
– powerOff: (NXEvent *) <i>theEvent</i>	Responds to a power–off subevent
– app: <i>sender</i> applicationDidTerminate: (const char *) <i>appName</i>	Notification that application terminated†

† These methods may be defined in the delegate or in a subclass. If a method is implemented both in a subclass and in the delegate, the message is sent to the delegate.

Box

Inherits From: View : Responder : Object

Initializing and Freeing a Box

- **initWithFrame:**(const NXRect *)*frameRect*
- **free**

Initializes a new Box object with the given *frameRect*
Deallocates the Box

Modifying the Border and Title

- **setBorderType:**(int)*aType*
- (int)**borderType**
- **setTitlePosition:**(int)*aPosition*
- (int)**titlePosition**
- **setTitle:**(const char *)*aString*
- (const char *)**title**
- **cell**
- **setFont:***fontObj*
- **font**

Sets the Box's border to *aType*
Returns the Box's border type
Sets the position of the title to *aPosition*
Returns the position of the title
Sets the Box's title to *aString*
Returns the title of the Box
Returns the Cell used to draw the title
Sets the Font of the title to *fontObj*
Returns the Font used to draw the title

Setting and Placing the Content View

- **setContentView:***aView*
- **contentView**
- **setOffsets:**(NXCoord)*w* :(NXCoord)*h*
- **getOffsets:**(NXSize *)*aSize*

Replaces the **Box's content view** with *aView*
Returns the **content view**
Sets the distance between the border and the content view
Gets the distance between the border and the content view

Putting Views in the Box

- **addSubview:***aView*
- **replaceSubview:***aView* **with:***anotherView*

Adds *aView* as a subview of the content view
Replaces *aView* with *anotherView* within the content view

Resizing the Box

- **setFrameFromContentFrame:**(const NXRect *)*contentFrame*
- **sizeTo:**(NXCoord)*width* :(NXCoord)*height*
- **sizeToFit**

Resizes the Box to accommodate *contentFrame*
Resizes the Box to *width* and *height*
Resizes the Box to exactly enclose its subviews

Drawing the Box

- **drawSelf:**(const NXRect *)*rects* :(int)*rectCount* Draws the Box

Archiving

- **awake** Lays out the graphic elements of the Box
- **read:**(NXTypedStream *)*stream* Reads the Box object from the typed stream
- **write:**(NXTypedStream *)*stream* Writes the Box object to the typed stream

Button

Inherits From: Control : View : Responder : Object

Initializing the Button Factory

- + **setCellClass:***classId* Sets the subclass of ButtonCell used by Button

Initializing a Button

- **init** Initializes a new Button with title “Button”
- **initWithFrame:**(const NXRect *)*frameRect* Initializes a new Button within *frameRect*
- **initWithFrame:**(const NXRect *)*frameRect*
 icon:(const char *)*iconName*
 tag:(int)*anInt*
 target:*anObject*
 action:(SEL)*aSelector*
 key:(unsigned short)*charCode*
 enabled:(BOOL)*flag* Initializes a new Button within *frameRect*,
 with the NXImage named *iconName* as its icon,
 anInt as its tag,
 anObject as its target object,
 aSelector as its action message,
 a key equivalent of *charCode*,
 and enabled according to *flag*
- **initWithFrame:**(const NXRect *)*frameRect*
 title:(const char *)*aString*
 tag:(int)*anInt*
 target:*anObject*
 action:(SEL)*aSelector*
 key:(unsigned short)*charCode*
 enabled:(BOOL)*flag* Initializes a new Button within *frameRect*,
 with *aString* as its title,
 anInt as its tag,
 anObject as its target object,
 aSelector as its action message,
 a key equivalent of *charCode*,
 and enabled according to *flag*

Setting the Button Type

– **setType:(int)***aType*

Sets how the Button highlights and shows its state

Setting the State

– **setState:(int)***value*

Sets the Button’s state to *value* (0 or 1)

– (int)**state**

Returns the Button’s current state (0 or 1)

Setting the Repeat Interval

– **setPeriodicDelay:(float)***delay*
andInterval:(float)*interval*

Sets repeat parameters for continuous Buttons

– **getPeriodicDelay:(float *)***delay*
andInterval:(float *)*interval*

Gets repeat parameters for continuous Buttons

Setting the Titles

– **setTitle:(const char *)***aString*

Makes *aString* the Button’s title

– **setTitleNoCopy:(const char *)***aString*

Makes *aString* the Button’s title without copying it

– (const char *)**title**

Returns the Button’s title

– **setAltTitle:(const char *)***aString*

Makes *aString* the Button’s alternate title

– (const char *)**altTitle**

Returns the Button’s alternate title

Setting the Icons

– **setIcon:(const char *)***iconName*

Makes the `UIImage` named *iconName* the Button’s icon

– **setIcon:(const char *)***iconName*
position:(int)*aPosition*

Sets the icon by name, and its position

– (const char *)**icon**

Returns the name of the Button’s icon

– **setAltIcon:(const char *)***iconName*

Makes the `UIImage` named *iconName* the alternate icon

– (const char *)**altIcon**

Returns the name of the Button’s alternate icon

– **setImage:image**

Makes the `UIImage` *image* the Button’s icon

– **image**

Returns the Button’s image

– **setAltImage:altImage**

Makes the `UIImage` *image* the alternate icon

– **altImage**

Returns the Button’s alternate image

– **setIconPosition:(int)***aPosition*

Sets the position of the Button’s icon

– (int)**iconPosition**

Returns the position of the Button’s icon

Modifying Graphic Attributes

- **setTransparent:**(BOOL)*flag* Sets whether the Button is transparent
- (BOOL)**isTransparent** Returns whether the Button is transparent
- **setBordered:**(BOOL)*flag* Sets whether the Button has a bezeled border
- (BOOL)**isBordered** Returns whether the Button has a bezeled border

Displaying

- **display** Displays the Button
- **highlight:**(BOOL)*flag* Highlights (or unhighlights) the Button according to *flag*

Setting the Key Equivalent

- **setKeyEquivalent:**(unsigned short)*charCode* Makes *charCode* the Button’s key equivalent
- (unsigned short)**keyEquivalent** Returns the Button’s key equivalent

Handling Events and Action Messages

- (BOOL)**acceptsFirstMouse** Ensures that the Button accepts first mouse-down
- **performClick:***sender* Simulates the user clicking the Button
- (BOOL)**performKeyEquivalent:**(NXEvent *)*theEvent* Simulates a mouse click, if the key is right

Setting the Sound

- **setSound:***soundObject* Sets the Sound played when the Button is pressed
- **sound** Returns the Sound played when the Button is pressed

ButtonCell

Inherits From: ActionCell : Cell : Object

Initializing, Copying, and Freeing a ButtonCell

- **init** Initializes a new ButtonCell with title “Button”
- **initWithTextCell:**(const char *)*aString* Initializes a new ButtonCell with title *aString*

- **initWithCell:**(const char *)*iconName* Initializes a new ButtonCell with an NXImage named *iconName* as its icon
- **copyFromZone:**(NXZone *)*zone* Returns a copy of the ButtonCell allocated from *zone*
- **free** Deallocates the ButtonCell

Determining Component Sizes

- **calcCellSize:**(NXSize *)*theSize*
 inRect:(const NXRect *)*aRect* Calculates and returns the size of the ButtonCell
- **getDrawRect:**(NXRect *)*theRect* Returns the rectangle the ButtonCell draws in
- **getTitleRect:**(NXRect *)*theRect* Returns the rectangle the title is drawn in
- **getIconRect:**(NXRect *)*theRect* Returns the rectangle the icon is drawn in

Setting the Titles

- **setTitle:**(const char *)*aString* Makes a copy of *aString* the ButtonCell’s title
- **setTitleNoCopy:**(const char *)*aString* Makes *aString* the ButtonCell’s title without copying it
- (const char *)**title** Returns the ButtonCell’s title
- **setAltTitle:**(const char *)*aString* Makes a copy of *aString* the ButtonCell’s alternate title
- (const char *)**altTitle** Returns the ButtonCell’s alternate title
- **setFont:***fontObject* Sets the Font used to draw the title

Setting the Icons

- **setIcon:**(const char *)*iconName* Makes the NXImage named *iconName* the ButtonCell’s icon
- (const char *)**icon** Returns the name of the ButtonCell’s icon
- **setAltIcon:**(const char *)*iconName* Makes the NXImage named *iconName* the ButtonCell’s alternate icon
- (const char *)**altIcon** Returns the name of the ButtonCell’s alternate icon
- **setImage:***image* Makes the NXImage object *image* the ButtonCell’s icon
- **image** Returns the ButtonCell’s icon
- **setAltImage:***altImage* Makes the NXImage object *image* the alternate icon
- **altImage** Returns the ButtonCell’s alternate icon
- **setIconPosition:**(int)*aPosition* Sets the position of the ButtonCell’s icon to *aPosition*
- (int)**iconPosition** Returns the position of the ButtonCell’s icon

Setting the Sound

- **setSound:***aSound*
- **sound**

Sets the Sound played by the ButtonCell on a mouse-down
Returns the Sound played by the ButtonCell

Setting the State

- **setDoubleValue:**(double)*aDouble*
- (double)**doubleValue**
- **setFloatValue:**(float)*aFloat*
- (float)**floatValue**
- **setIntValue:**(int)*anInt*
- (int)**intValue**
- **setStringValue:**(const char *)*aString*
- **setStringValueNoCopy:**(const char *)*aString*
- (const char *)**stringValue**

Sets the ButtonCell's state (value) to *aDouble*
Returns the ButtonCell's state as a **double**
Sets the ButtonCell's state (value) to *aFloat*
Returns the ButtonCell's state as a **float**
Sets the ButtonCell's state (value) to *anInt*
Returns the ButtonCell's state as an **int**
Sets the ButtonCell's state (value) to a copy of *aString*
Sets the ButtonCell's state (value) to *aString*
Returns the ButtonCell's state as a string

Setting the Repeat Intervae

- **setPeriodicDelay:**(float)*delay*
andInterval:(float)*interval*
- **getPeriodicDelay:**(float *)*delay*
andInterval:(float *)*interval*

Sets repeat parameters for continuous ButtonCells

Gets repeat parameters for continuous ButtonCells

Tracking the Mouse

- (BOOL)**trackMouse:**(NXEvent *)*theEvent*
inRect:(const NXRect *)*cellFrame*
ofView:*controlView*

Plays the Sound, then tracks the mouse

Setting the Key Equivalent

- **setKeyEquivalent:**(unsigned short)*charCode*
- **setKeyEquivalentFont:***fontObj*
- **setKeyEquivalentFont:**(const char *)*fontName*
size:(float)*fontSize*
- (unsigned short)**keyEquivalent**

Sets the ButtonCell's key equivalent
Sets the Font used to draw the key equivalent
Sets the Font and size used to draw the key equivalent
Returns the ButtonCell's key equivalent

Setting Parameters

- **setParameter:(int)aParameter to:(int)value** Sets various flag values
- **(int)getParameter:(int)aParameter** Returns various flag values

Modifying Graphic Attributes

- **setBordered:(BOOL)flag** Sets whether the ButtonCell has a bezeled border
- **(BOOL)isBordered** Returns whether the ButtonCell has a bezeled border
- **setTransparent:(BOOL)flag** Sets whether the ButtonCell is transparent
- **(BOOL)isTransparent** Returns whether the ButtonCell is transparent
- **(BOOL)isOpaque** Returns whether receiver is opaque

Modifying Graphic Attributes

- **setType:(int)aType** Sets the ButtonCell's display behavior
- **setHighlightsBy:(int)aType** Sets how the ButtonCell highlights
- **(int)highlightsBy** Returns how the ButtonCell highlights
- **setShowsStateBy:(int)aType** Sets how the ButtonCell shows its alternate state
- **(int)showsStateBy** Returns how ButtonCell shows its alternate state

Simulating a Click

- **performClick:sender** Simulates clicking the ButtonCell

Displaying

- **drawInside:(const NXRect *)aRect**
inView:controlView Draws the inside of the ButtonCell
- **drawSelf:(const NXRect *)cellFrame**
inView:controlView Draws the ButtonCell
- **highlight:(const NXRect *)cellFrame**
inView:controlView
lit:(BOOL)flag Highlights the ButtonCell

Archiving

- **read:(NXTypedStream *)stream** Reads the ButtonCell from *stream*
- **write:(NXTypedStream *)stream** Writes the ButtonCell to *stream*

Cell

Inherits From: Object

Initializing, Copying, and Freeing a Cell

- **init** Initializes a new Cell
- **initWithCell:(const char *)*iconName*** Initializes a new Cell with the NXImage named *iconName*
- **initWithTextCell:(const char *)*aString*** Initializes a new Cell with title *aString*
- **copyFromZone:(NXZone *)*zone*** Returns a copy of the receiving Cell from *zone*
- **free** Deallocates the Cell

Determining Component Sizes

- **calcCellSize:(NXSize *)*theSize*** Returns the minimum size needed to display the Cell
- **calcCellSize:(NXSize *)*theSize*
inRect:(const NXRect *)*aRect*** Returns the minimum size needed to display the Cell
- **calcDrawInfo:(const NXRect *)*aRect*** Implemented by subclasses to recalculate drawing sizes
- **getDrawRect:(NXRect *)*theRect*** Returns the rectangle the Cell draws in
- **getIconRect:(NXRect *)*theRect*** Returns the rectangle that an icon is drawn in
- **getTitleRect:(NXRect *)*theRect*** Returns the rectangle that a title is drawn in

Setting the Cell's Type

- **setType:(int)*aType*** Sets the Cell's type to *aType*
- **(int)*type*** Returns the Cell's type

Setting the Cell's State

- **setState:(int)*value*** Sets the state of the Cell to *value* (0 or 1)
- **incrementState** Increments the state of the Cell
- **(int)*state*** Returns the state of the Cell (0 or 1)

Enabling and Disabling the Cell

- **setEnabled:(BOOL)*flag*** Sets whether the Cell reacts to mouse events
- **(BOOL)*isEnabled*** Returns whether the Cell reacts to mouse events

Setting the Icon

- **setIcon:**(const char *)*iconName*
- (const char *)**icon**

Sets the Cell's icon to the NXImage named *iconName*
Returns the name of the Cell's icon

Setting the Cell's Value

- **setDoubleValue:**(double)*aDouble*
- (double)**doubleValue**
- **setFloatValue:**(float)*aFloat*
- (float)**floatValue**
- **setIntValue:**(int)*anInt*
- (int)**intValue**
- **setStringValue:**(const char *)*aString*
- **setStringValueNoCopy:**(const char *)*aString*
- **setStringValueNoCopy:**(char *)*aString*
shouldFree:(BOOL)*flag*
- (const char *)**stringValue**

Sets the Cell's value to *aDouble*
Returns the Cell's value as a **double**
Sets the Cell's value to *aFloat*
Returns the Cell's value as a **float**
Sets the Cell's value to *anInt*
Returns the Cell's value as an **int**
Sets the Cell's value to a copy of *aString*
Sets the Cell's value to *aString*
Sets the Cell's value to *aString*; will free the string when freed if *flag* is YES
Returns the Cell's value as a string

Interacting with Other Cells

- **takeDoubleValueFrom:***sender*
- **takeFloatValueFrom:***sender*
- **takeIntValueFrom:***sender*
- **takeStringValueFrom:***sender*

Sets the Cell's value to *sender's doubleValue*
Sets the Cell's value to *sender's floatValue*
Sets the Cell's value to *sender's intValue*
Sets the Cell's value to *sender's stringValue*

Modifying Text Attributes

- **setAlignment:**(int)*mode*
- (int)**alignment**
- **setFont:***fontObject*
- **font**
- **setEditable:**(BOOL)*flag*
- (BOOL)**isEditable**
- **setSelectable:**(BOOL)*flag*
- (BOOL)**isSelectable**
- **setScrollable:**(BOOL)*flag*
- (BOOL)**isScrollable**
- **setTextAttributes:***textObject*
- **setWrap:**(BOOL)*flag*

Sets the alignment of text in the Cell to *mode*
Returns the alignment of text in the Cell
Sets the Font used to display text in the Cell to *fontObject*
Returns the Font used to display text in the Cell
Sets whether the Cell's text is editable
Returns whether the Cell's text is editable
Sets whether the Cell's text is selectable
Returns whether the Cell's text is selectable
Sets whether the Cell scrolls to follow typing
Returns whether the Cell scrolls to follow typing
Sets Text parameters for drawing or editing
Sets whether the Cell's text is word-wrapped

Editing Text

- **edit:**(const NXRect *)*aRect*
inView:*aView*
editor:*textObject*
delegate:*anObject*
event:(NXEvent *)*theEvent* Allows text editing in response to a mouse-down event
- **endEditing:***textObject* Ends any text editing occurring in the Cell
- **select:**(const NXRect *)*aRect*
inView:*aView*
editor:*textObject*
delegate:*anObject*
start:(int)*selStart*
length:(int)*selLength* Allows text selection in response to a mouse-down event

Validating Input

- **setEntryType:**(int)*aType* Sets the type of data the user can type into the Cell
- (int)**entryType** Returns the type of data the user can type into the Cell
- (BOOL)**isEntryAcceptable:**(const char *)*aString* Returns whether *aString* is acceptable for the entry type

Formatting Data

- **setFloatingPointFormat:**(BOOL)*autoRange*
left:(unsigned)*leftDigits*
right:(unsigned)*rightDigits* Sets the display format for floating point values

Modifying Graphic Attributes

- **setBezeled:**(BOOL)*flag* Sets whether the Cell has a bezeled border
- (BOOL)**isBezeled** Returns whether the Cell has a bezeled border
- **setBordered:**(BOOL)*flag* Sets whether the Cell has a plain border
- (BOOL)**isBordered** Returns whether Cell has a plain border
- (BOOL)**isOpaque** Returns whether the Cell is opaque

Setting Parameters

- **setParameter:**(int)*aParameter* **to:**(int)*value* Sets various Cell flags
- (int)**getParameter:**(int)*aParameter* Returns various Cell flag values

Displaying

- **controlView** Implemented by subclasses to return the View last drawn in
- **drawInside:(const NXRect *)cellFrame**
inView:aView Draws the area within the Cell's border in *aView*
- **drawSelf:(const NXRect *)cellFrame**
inView:aView Draws the Cell in *aView*
- **highlight:(const NXRect *)cellFrame**
inView:aView
lit:(BOOL)flag Highlights the Cell according to *flag* in *aView*
- **(BOOL)isHighlighted** Returns whether the Cell is highlighted

Target and Action

- **setAction:(SEL)aSelector** Implemented by subclasses to set the action method
- **(SEL)action** Implemented by subclasses to return the action method
- **setTarget:anObject** Implemented by subclasses to set the target object
- **target** Implemented by subclasses to return the target object
- **setContinuous:(BOOL)flag** Sets whether the Cell continuously sends action
- **(BOOL)isContinuous** Returns whether the Cell continuously sends action
- **(int)sendActionOn:(int)mask** Determines when the action is sent while tracking

Assigning a Tag

- **setTag:(int)anInt** Implemented by subclasses to set an identifier tag
- **(int)tag** Implemented by subclasses to return the identifier tag

Handling Keyboard Alternatives

- **(unsigned short)keyEquivalent** Implemented by subclasses to return a key equivalent

Tracking the Mouse

- + **(BOOL)prefersTrackingUntilMouseUp** Returns NO, so tracking stop when the mouse leaves the Cell; subclasses may override
- **(int)mouseDownFlags** Returns the event flags set at the start of mouse tracking
- **getPeriodicDelay:(float*)delay**
andInterval:(float*)interval Returns repeat values for continuous sending of the action
- **(BOOL)trackMouse:(NXEvent *)theEvent**
inRect:(const NXRect *)cellFrame
ofView:aView Controls tracking behavior of the Cell

- (BOOL)**startTrackingAt:**(const NXPoint *)*startPoint*
inView:*aView* Determines whether tracking should begin base on *startPoint* within *aView*
- (BOOL)**continueTracking:**(const NXPoint *)*lastPoint*
at:(const NXPoint *)*currentPoint* Returns whether tracking should continue based on *lastPoint* and *currentPoint* within *aView*
inView:*aView*
- **stopTracking:**(const NXPoint *)*lastPoint* Allows the Cell to update itself to end tracking, based on *lastPoint*, *stopPoint*, within *aView*; *flag* is YES if the
at:(const NXPoint *)*stopPoint* this method was invoked because mouse went up
inView:*aView*
mouseIsUp:(BOOL)*flag*

Managing the Cursor

- **resetCursorRect:**(const NXRect *)*cellFrame* Sets text Cells to show I-beam cursor
inView:*aView*

Archiving

- **read:**(NXTypedStream *)*stream* Reads the Cell from *stream*
- **write:**(NXTypedStream *)*stream* Writes the Cell to *stream*
- **awake** Reinitializes the Cell after being read

ClipView

Inherits From: View : Responder : Object

Initializing the Class

- + **initialize** Initializes the ClipView class

Initializing and Freeing a ClipView

- **initWithFrame:**(const NXRect *)*frameRect* Initializes a new ClipView instance
- **free** Releases the ClipView’s storage

Modifying the Frame Rectangle

- **moveTo:**(NXCoord)*x* :(NXCoord)*y* Moves the origin of the frame rectangle
- **rotateTo:**(NXCoord)*angle* Overridden to disable rotation
- **sizeTo:**(NXCoord)*width* :(NXCoord)*height* Resizes the ClipView’s frame

Modifying the Coordinate System

- **rotate:**(NXCoord)*angle* Overridden to disable rotation
- **scale:**(NXCoord)*x* :(NXCoord)*y* Rescales the coordinate system
- **setDrawOrigin:**(NXCoord)*x* :(NXCoord)*y* Sets the origin of the coordinate system
- **setDrawRotation:**(NXCoord)*angle* Disables rotation of the coordinate system
- **setDrawSize:**(NXCoord)*width* :(NXCoord)*height* Scales the coordinate system
- **translate:**(NXCoord)*x* :(NXCoord)*y* Shifts the coordinate system

Managing Component Views

- **docView** Returns the ClipView’s document view
- **setDocView:***aView* Makes *aView* the ClipView’s document view
- **getDocRect:**(NXRect *)*aRect* Returns the document rectangle
- **getDocVisibleRect:**(NXRect *)*aRect* Gets the visible portion of the document view
- **resetCursorRects** Resets the cursor rectangle for the document view
- **setDocCursor:***anObj* Sets the cursor for the document view

Modifying Graphic Attributes and Displaying

- (float)**backgroundGray** Returns the ClipView’s background gray
- **setBackgroundGray:**(float)*value* Sets the ClipView’s background gray
- (NXColor)**backgroundColor** Returns the ClipView’s background color
- **setBackground-color:**(NXColor)*color* Sets the ClipView’s background color
- **drawSelf:**(const NXRect *)*rects* :(int) *rectCount* Fills the background gray where needed

Scrolling

- **autoscroll:**(NXEvent *)*theEvent* Scrolls in response to mouse-dragged events
- **constrainScroll:**(NXPoint *)*newOrigin* Prevents scrolling to an undesirable position
- **rawScroll:**(const NXPoint *)*newOrigin* Lowest-level unconstrained scrolling routine
- **setCopyOnScroll:**(BOOL)*flag* Sets how the visible areas are redrawn
- **setDisplayOnScroll:**(BOOL)*flag* Sets how the document view is displayed during scrolling

Coordinating with Other Views

- **descendantFlipped:***sender*
- **descendantFrameChanged:***sender*

Notification that the document's orientation has changed

Notification that the document's frame has changed

Archiving

- **awake**
- **read:**(NXTypedStream *)*stream*
- **write:**(NXTypedStream *)*stream*

Initializes the ClipView after unarchiving

Reads the ClipView from the typed stream

Writes the ClipView to the typed stream

Implemented by ClipView's Superview

- **reflectScroll:***aClipView*
- **scrollClip:***aClipView* **to:**(const NXPoint *)*aPoint*

Notifies the superview to update indicators

Notifies the superview of a scroll

Control

Inherits From: View : Responder : Object

Initializing and Freeing a Control

- **initWithFrame:**(const NXRect *)*frameRect*
- **free**

Initializes a new Control

Deallocates the Control

Setting the Control's Cell

- + **setCellClass:***classId*
- **setCell:***aCell*
- **cell**

Implemented by subclasses to set the Cell class used

Sets the Control's Cell to *aCell*

Returns the Control's Cell

Enabling and Disabling the Control

- **setEnabled:**(BOOL)*flag*
- (BOOL)**isEnabled**

Sets whether the Control reacts to mouse events

Returns whether the Control reacts to mouse events

Identifying the Selected Cell

- `selectedCell`
- `(int)selectedTag`

Returns the Control's selected Cell

Returns the tag of the Control's selected Cell

Setting the Control's Value

- `setDoubleValue:(double)aDouble`
- `(double)doubleValue`
- `setFloatValue:(float)aFloat`
- `(float)floatValue`
- `setIntValue:(int)anInt`
- `(int)intValue`
- `setStringValue:(const char *)aString`
- `setStringValueNoCopy:(const char *)aString`
- `setStringValueNoCopy:(char *)aString`
 `shouldFree:(BOOL)flag`
- `(const char *)stringValue`

Sets the Control's value to *aDouble*

Returns the Control's value as a **double**

Sets the Control's value to *aFloat*

Returns the Control's value as a **float**

Sets the Control's value to *anInt*

Returns the Control's value as an **int**

Sets the Control's value to *aString*

Sets the Control's value to *aString*

Sets the Control's value to *aString*

Returns the Control's value as a string

Interacting with Other Controls

- `takeDoubleValueFrom:sender`
- `takeFloatValueFrom:sender`
- `takeIntValueFrom:sender`
- `takeStringValueFrom:sender`

Sets the Control's value to *sender's doubleValue*

Sets the Control's value to *sender's floatValue*

Sets the Control's value to *sender's intValue*

Sets the Control's value to *sender's stringValue*

Formatting Text

- `setAlignment:(int)mode`
- `(int)alignment`
- `setFont:fontObject`
- `font`
- `setFloatingPointFormat:(BOOL)autoRange`
 `left:(unsigned)leftDigits`
 `right:(unsigned)rightDigits`

Sets the alignment of text in the Control to *mode*

Returns the alignment of text in the Control

Sets the Font used to draw text in the Control to *fontObject*

Returns the Font used to draw text in the Control

Sets the display format for floating point values

Managing the Field Editor

- `abortEditing`
- `currentEditor`
- `validateEditing`

Aborts editing of text displayed by the Control

Returns the object used to edit text in the Control

Validates the user's changes to editable text

Managing the Cursor

- **resetCursorRects**

Sets text-bearing Controls to show an I-beam cursor

Resizing the Control

- **calcSize**
- **sizeTo:(NXCoord)width :(NXCoord)height**
- **sizeToFit**

Recalculates internal size information

Resizes the Control to *width* and *height*

Resizes the Control to fit its Cell

Displaying the Control and Cell

- **drawCell:aCell**
- **drawCellInside:aCell**
- **drawSelf:(const NXRect *)rects :(int)rectCount**
- **selectCell:aCell**
- **update**
- **updateCell:aCell**
- **updateCellInside:aCell**

Redraws *aCell* if it's the Control's Cell

Redraws *aCell*'s inside if it's the Control's Cell

Draws the Control

Selects *aCell* if it's the Control's cell

Redisplays the Control or marks it for later redisplay

Redisplays *aCell* or marks it for redisplay

Redisplays the inside of *aCell* or marks it for redisplay

Target and Action

- **setAction:(SEL)aSelector**
- **(SEL)action**
- **setTarget:anObject**
- **target**
- **setContinuous:(BOOL)flag**
- **(BOOL)isContinuous**
- **sendAction:(SEL)theAction to:theTarget**
- **(int)sendActionOn:(int)mask**

Sets the Control's action method to *aSelector*

Returns the Control's action method

Sets the Control's target object to *anObject*

Returns the Control's target object

Sets whether the Control continuously sends its action

Returns whether the Control continuously sends its action

Has the Application object send *theAction* to *theTarget*

Determines when the action is sent while tracking

Assigning a Tag

- **setTag:(int)anInt**
- **(int)tag**

Sets the Control's tag to *anInt*

Returns the Control's tag

Tracking the Mouse

- **ignoreMultiClick:(BOOL)flag**
- **mouseDown:(NXEvent *)theEvent**
- **(int)mouseDownFlags**

Sets whether multiple clicks are ignored

Handles a mouse-down event in the Control

Returns flags in effect at beginning of tracking

Archiving

- **read:**(NXTypedStream *)*stream* Reads the Control from *stream*
- **write:**(NXTypedStream *)*stream* Writes the Control to *stream*

Font

Inherits From: Object

Initializing the Class Object

- + **initialize** Performed automatically at start-up
- + **useFont:**(const char *)*fontName* Registers that *fontName* is used in the document

Creating and Freeing a Font Object

- + **newFont:**(const char *)*fontName*
size:(float)*fontSize* Returns the specified Font object
- + **newFont:**(const char *)*fontName*
size:(float)*fontSize*
matrix:(const float *)*fontMatrix* Returns the specified Font object
- + **newFont:**(const char *)*fontName*
size:(float)*fontSize*
style:(int)*fontStyle*
matrix:(const float *)*fontMatrix* Returns the specified Font object
- + **boldSystemFontOfSize:**(float)*fontSize*
matrix:(const float *)*fontMatrix* Returns the Font object representing the bold system font of size *fontSize* and matrix *fontMatrix*
- + **userFixedPitchFontOfSize:**(float)*fontSize*
matrix:(const float *)*fontMatrix* Returns the Font object representing the application's fixed-pitch font of size *fontSize* and matrix *fontMatrix*
- + **userFontOfSize:**(float)*fontSize*
matrix:(const float *)*fontMatrix* Returns the Font object representing the application's standard font of size *fontSize* and matrix *fontMatrix*
- + **systemFontOfSize:**(float)*fontSize*
matrix:(const float *)*fontMatrix* Returns the Font object representing the system font of size *fontSize* and matrix *fontMatrix*
- **free** Has no effect

Querying the Font Object

- (const float *) displayName	Returns the full name of the font
- (const float *) familyName	Returns the name of the font's family
- (const char *) name	Returns the name of the font
- (int) fontNum	Returns the Window Server's font number
- (float) getWidthOf:(const char *)string	Returns the width of <i>string</i> in this font
- (BOOL) hasMatrix	Returns whether font differs from identity matrix
- (const float *) matrix	Returns a pointer to the font matrix
- (NXFontMetrics *) metrics	Returns pointer to a record of font information
- (float) pointSize	Returns the size of the font in points
- (NXFontMetrics *) readMetrics:(int)flags	Reads <i>flags</i> information into the font record
- screenFont	Returns the screen font for this font
- (int) style	Returns the font style

Setting the Font

- set	Makes this font the graphic state's current font
- setStyle:(int)aStyle	Sets the Font's style
+ setUserFixedPitchFont:(Font *)aFont	Sets the fixed-pitch font used by default in the application
+ setUserFont:(Font *)aFont	Sets the standard font used by default in the application

Archiving

- awake	Reinitializes the Font object
- finishUnarchiving	Checks whether the Font object already exists
- read:(NXTypedStream *)stream	Reads the Font object from the typed stream
- write:(NXTypedStream *)stream	Writes the Font object to the typed stream

FontManager

Inherits From: Object

Creating a FontManager

+ new	Returns the application-wide FontManager
--------------	--

Converting Fonts

- **convertFont:***fontObj*
- **convertWeight:**(*BOOL*)*upFlag of:fontObj*
- **convert:***fontObj toFace:*(*const char **)*typeface*
- **convert:***fontObj toFamily:*(*const char **)*family*
- **convert:***fontObj toSize:*(*float*)*size*
- **convert:***fontObj toHaveTrait:*(*NXFontTraitMask*)*trait*
- **convert:***fontObj toNotHaveTrait:*(*NXFontTraitMask*)*trait*
- **findFont:**(*const char **)*family traits:*(*NXFontTraitMask*)*traits weight:*(*int*)*weight size:*(*float*)*size*
- **getFamily:**(*const char ***)*family traits:*(*NXFontTraitMask **)*traits weight:*(*int **)*weight size:*(*float**)*size ofFont:**fontObj*

Converts the font in response to **changeFont:**

Raises or lowers the weight of the font

Converts the font to the specified typeface

Converts the font to the specified family

Converts the font to the specified point size

Converts the font to have the specified trait

Converts the font to remove the specified trait

Tries to find a font that matches the specified characteristics

Provides the characteristics of the given *fontObj*

Setting Parameters

- **setAction:**(*SEL*)*aSelector*
- + **setFontPanelFactory:***classId*
- + **setFontManagerFactory:***classId*
- **setSelFont:***fontObj isMultiple:*(*BOOL*)*flag*
- **setEnabled:**(*BOOL*)*flag*

Sets the action sent by the FontManager

Sets the class used to create the Font panel

Sets the class used to create the font manager

Notifies FontManager of selection's current font

Enables and disables the Font panel and menu

Querying Parameters

- (*SEL*)**action**
- (*char ***)**availableFonts**
- **getFontMenu:**(*BOOL*)*create*
- **getFontPanel:**(*BOOL*)*create*
- (*BOOL*)**isMultiple**
- **selFont**
- (*BOOL*)**isEnabled**

Gets the action sent by the FontManager

Provides a list of all available fonts

Returns the Font menu

Returns the Font panel

Returns whether selection contains multiple fonts

Returns the first font in the current selection

Returns whether the Font panel and menu are enabled

Target and Action Methods

- **modifyFont:***sender* Converts current selection's font
- **addFontTrait:***sender* Causes trait to be added to font in current selection
- **removeFontTrait:***sender* Causes trait to be removed from font in current selection
- **modifyFontViaPanel:***sender* Converts font according to Font panel settings
- **orderFrontFontPanel:***sender* Orders the FontPanel front
- **sendAction** Dispatches *action* message up responder chain

Assigning a Delegate

- **setDelegate:***anObject* Sets the FontManager's delegate
- **delegate:** Returns the FontManager's delegate

Archiving the FontManager

- **finishUnarchiving** Finishes unarchiving by creating the FontManager

FontPanel

Inherits From: Panel : Window : Responder : Object

Creating a FontPanel

- + **new** Returns a FontPanel object
- + **newContent:***(const NXRect *)contentRect* Returns a FontPanel object
 - style:***(int)aStyle*
 - backing:***(int)bufferingType*
 - buttonMask:***(int)mask*
 - defer:***(BOOL)flag*

Setting the Font

- **setPanelFont:***fontObj isMultiple:(BOOL)flag* Sets the Font panel's current font
- **panelConvertFont:***fontObj* Converts *fontObj* to the user's choice from the panel

Configuring the FontPanel

- **accessoryView** Returns the application-customized view
- **setAccessoryView:***aView* Adds application-customized View to the FontPanel
- **setEnabled:**(BOOL)*flag* Enables and disables the FontPanel’s Set button
- (BOOL)**isEnabled** Returns whether the FontPanel’s Set button is enabled
- (BOOL)**worksWhenModal** Returns whether FontPanel works when another window is modal

Editing the FontPanel’s Fields

- **textDidEnd:***textObject* endChar:(unsigned short)*endChar* Detects completion of size field editing
- **textDidGetKeys:***textObject* isEmpty:(BOOL)*flag* Detects empty size field

Displaying the FontPanel

- **orderWindow:**(int)*place* relativeTo:(int)*otherWin* Repositions panel and updates it if necessary

Resizing the FontPanel

- **windowWillResize:***sender* toSize:(NXSize *)*frameSize* Constrains FontPanel resizing

Form

Inherits From: Matrix : Control : View : Responder : Object

Setting Form’s Cell Class

- + **setCellClass:***classId* Sets the subclass of Cell used by Form

Initializing a Form

- **initWithFrame:**(const NXRect *)*frameRect* Initializes a new Form in *frameRect*

Laying Out the Form

- **addEntry:**(const char *)*title* Adds a new entry with *title* as its title at the end of the Form
- **addEntry:**(const char *)*title*
tag:(int)*anInt* Adds a new entry with *title* as its title at the end of the Form
target:*anObject* and sets its tag, target, and action
action:(SEL)*aSelector*
- **insertEntry:**(const char *)*title at:*(int)*index* Inserts a new entry with *title* as its title at *index*
- **insertEntry:**(const char *)*title*
at:(int)*index* Inserts a new entry with *title* as its title at *index* and sets
tag:(int)*anInt* its tag, target, and action
target:*anObject*
action:(SEL)*aSelector*
- **removeEntryAt:**(int)*index* Removes the entry at *index*
- **setInterline:**(NXCoord)*spacing* Sets the spacing between entries

Assigning a Tag

- **setTag:**(int)*anInt at:*(int)*index* Sets the tag of the entry at *index* to *anInt*

Finding Indices

- (int)**findIndexWithTag:**(int)*aTag* Returns the index for the entry with tag *aTag*
- (int)**selectedIndex** Returns the index of the currently selected entry

Modifying Graphic Attributes

- **setBezeled:**(BOOL)*flag* Sets whether entries have a bezeled border
- **setBordered:**(BOOL)*flag* Sets whether the entries have a plain border
- **setFont:***fontObject* Sets the Font used to draw both titles and text
- **setTitleFont:***fontObject* Sets the Font used to draw entry titles
- **setTextFont:***fontObject* Sets the Font used to draw entry text
- **setTitleAlignment:**(int)*mode* Sets how titles are aligned
- **setTextAlignment:**(int)*mode* Sets how text is aligned within the entries

Setting Item Titles

- **setTitle:**(const char *)*aString at:*(int)*index* Sets the title of the entry at *index* to *aString*
- (const char *)**titleAt:**(int)*index* Returns the title of the entry at *index*

Setting Item Values

- **setDoubleValue:(double)aDouble at:(int)index** Sets the value of the entry at *index* to *aDouble*
- **(double)doubleValueAt:(int)index** Returns the value of the entry at *index* as a **double**
- **setFloatValue:(float)aFloat at:(int)index** Sets the value of the entry at *index* to *aFloat*
- **(float)floatValueAt:(int)index** Returns the value of the entry at *index* as a **float**
- **setIntValue:(int)anInt at:(int)index** Sets the value of the entry at *index* to a *anInt*
- **(int)intValueAt:(int)index** Returns the value of the entry at *index* as an **int**
- **setStringValue:(const char *)aString at:(int)index** Sets the value of the entry at *index* to a *aString*
- **(const char *)stringValueAt:(int)index** Returns the value of the entry at *index* as a string

Editing Text

- **selectTextAt:(int)index** Selects the text in the entry at *index*

Resizing the Form

- **calcSize** Recalculates title positions in the Form
- **setEntryWidth:(NXCoord)width** Sets the width of all the entries
- **sizeTo:(NXCoord)width :(NXCoord)height** Resizes the Form and updates the widths of its entries
- **sizeToFit** Modifies the Form's frame to fit its entries

Displaying

- **drawCellAt:(int)index** Displays the Cell at the specified *index*

Target and Action

- **setAction:(SEL)aSelector at:(int)index** Sets the action method of the entry at *index* to *aSelector*
- **setTarget:anObject at:(int)index** Sets the target object of the entry at *index* to *anObject*

FormCell

Inherits From: ActionCell : Cell : Object

Initializing, Copying, and Freeing a FormCell

- **init** Initializes a new FormCell with “Field” as its title
- **initWithTextCell:(const char *)aString** Initializes a new FormCell with *aString* as its title
- **copyFromZone:(NXZone *)zone** Returns a copy of the FormCell allocated from *zone*
- **free** Deallocates the FormCell

Determining a FormCell’s Size

- **calcCellSize:(NXSize *)theSize
inRect:(const NXRect *)aRect** Calculates the FormCell’s size within *aRect*

Enabling the FormCell

- **setEnabled:(BOOL)flag** Sets whether the FormCell reacts to events

Modifying the Title

- **setTitle:(const char *)aString** Sets the FormCell’s title to *aString*
- **(const char *)title** Returns the FormCell’s title
- **setTitleFont:fontObject** Sets the Font used to draw the title
- **titleFont** Returns the Font used to draw the title
- **setTitleAlignment:(int)mode** Sets the alignment of the title
- **(int)titleAlignment** Returns the alignment of the title
- **setTitleWidth:(NXCoord)width** Sets the width of the FormCell’s title field to *width*
- **(NXCoord)titleWidth:(const NXSize *)aSize** Returns the width of the title, constrained to *aSize*
- **(NXCoord)titleWidth** Returns the width of the title

Modifying Graphic Attributes

- **(BOOL)isOpaque** Returns whether the FormCell is opaque

Displaying

- **drawInside:**(const NXRect *)*cellFrame*
inView:*controlView* Draws the editable text portion of the cell
- **drawSelf:**(const NXRect *)*cellFrame*
inView:*controlView* Draws the entire FormCell

Managing Cursor Rectangles

- **resetCursorRect:**(const NXRect *)*cellFrame*
inView:*controlView* Resets the cursor rectangle so that the cursor becomes an I-beam when over the editable portion of the FormCell

Tracking the Mouse

- (BOOL)**trackMouse:**(NXEvent*)*event*
inRect:(const NXRect*)*aRect*
ofView:*controlView* Overrides Cell’s method to allow editing

Archiving

- **read:**(NXTypedStream *)*stream* Reads the FormCell from *stream*
- **write:**(NXTypedStream *)*stream* Writes the FormCell to *stream*

Listener

Inherits From: Object

Initializing the Class

- + **initialize** Sets up a table of understood messages

Initializing a New Listener Instance

- **init** Initializes the Listener after it’s allocated

Freeing a Listener

- **free** Deallocates the Listener and its ports

Setting Up a Listener

– addPort	Sets procedure to receive messages at port
– removePort	Removes procedure that receives messages
– (int) checkInAs :(const char *) <i>name</i>	Allocates a port and registers it as <i>name</i>
– (int) usePrivatePort	Allocates a port but doesn't register it
– (int) checkOut	Unregisters the port, making it private
– (port_t) listenPort	Returns the Listener's port
– (port_t) signaturePort	Returns the port used to validate the Listener
– (const char *) portName	Returns registered name of the Listener's port
– setPriority :(int) <i>level</i>	Sets the priority for receiving messages to <i>level</i>
– (int) priority	Returns priority level for receiving messages
– setTimeout :(int) <i>ms</i>	Sets how long to wait on sending reply
– (int) timeout	Returns how long to wait on sending reply
+ run	Enables Listener in absence of an Application object

Providing for Program Control

– (int) msgCalc :(int *) <i>flag</i>	Receives message to update the current window
– (int) msgCopyAsType :(const char *) <i>aType</i> ok :(int *) <i>flag</i>	Receives message to copy the selection
– (int) msgCutAsType :(const char *) <i>aType</i> ok :(int *) <i>flag</i>	Receives message to cut selection as <i>aType</i> data
– (int) msgDirectory :(char *const *) <i>fullPath</i> ok :(int *) <i>flag</i>	Receives message asking for the current directory
– (int) msgFile :(char *const *) <i>fullPath</i> ok :(int *) <i>flag</i>	Receives message asking for the current document
– (int) msgPaste :(int *) <i>flag</i>	Receives message to paste data from pasteboard
– (int) msgPosition :(char *const *) <i>aString</i> posType :(int *) <i>anInt</i> ok :(int *) <i>flag</i>	Receives message requesting selection information
– (int) msgPrint :(const char *) <i>fullPath</i> ok :(int *) <i>flag</i>	Receives message to print the <i>fullPath</i> file
– (int) msgQuit :(int *) <i>flag</i>	Receives a remote message to quit
– (int) msgSelection :(char *const *) <i>bytes</i> length :(int *) <i>numBytes</i> asType :(const char *) <i>aType</i> ok :(int *) <i>flag</i>	Receives message requesting the current selection

- (int)msgSetPosition:(const char *)aString
posType:(int)anInt
andSelect:(int)selectFlag
ok:(int *)flag Receives message to scroll so *aString* is visible
- (int)msgVersion:(char *const *)aString
ok:(int *)flag Receives message requesting version information

Receiving Remote Messages

- messageReceived:(NXMessage *)msg Receives messages at the Listener's port
- (int)performRemoteMethod:(NXRemoteMethod *)method
paramList:(NXParamValue *)params Performs Listener's remote *method*
- (NXRemoteMethod *)remoteMethodFor:(SEL)aSelector
 Looks up remote method for *aSelector*

Assigning a Delegate

- setDelegate:anObject Makes *anObject* the Listener's delegate
- delegate Returns the Listener's delegate
- setServicesDelegate:anObject Makes *anObject* the receiver of service requests
- servicesDelegate Returns the object that receives service requests

Archiving

- read:(NXTypedStream *)stream Reads the Listener from *stream*
- write:(NXTypedStream *)stream Writes the Listener to *stream*

Matrix

Inherits From: Control : View : Responder : Object

Initializing the Matrix Class

- + initialize Initializes the Matrix class
- + setCellClass:className Sets the default class used to make Cells

Initializing and Freeing a Matrix

- **initWithFrame:**(const NXRect *)*frameRect*
Initializes a new Matrix object in *frameRect*
- **initWithFrame:**(const NXRect *)*frameRect*
mode:(int)*aMode*
cellClass:*classId*
numRows:(int)*numRows*
numCols:(int)*numCols*
Initializes a new Matrix object in *frameRect*,
with *aMode* as the selection mode,
classId as the class used to make new Cells,
and having *numRows* rows
and *numCols* columns
- **initWithFrame:**(const NXRect *)*frameRect*
mode:(int)*aMode*
prototype:*aCell*
numRows:(int)*numRows*
numCols:(int)*numCols*
Initializes a new Matrix object with the given values
with *aMode* as the selection mode,
aCell as the prototype copied to make new Cells,
and having *numRows* rows
and *numCols* columns
- **free**
Deallocates the Matrix and all its Cells

Setting the Selection Mode

- **setMode:**(int)*aMode*
Sets the selection mode of the Matrix
- (int)**mode**
Returns the selection mode of the Matrix

Configuring the Matrix

- **setEnabled:**(BOOL)*flag*
Sets whether the Matrix reacts to events
- **setEmptySelectionEnabled:**(BOOL)*flag*
Sets whether there may be no Cells selected
- (BOOL)**isEmptySelectionEnabled**
Returns whether there may be no Cells selected
- **setSelectionByRect:**(BOOL)*flag*
Sets whether a user can drag a rectangular selection
(the default is YES)
- (BOOL)**isSelectionByRect**
Returns whether a user can drag a rectangular selection

Setting the Cell class

- **setCellClass:***classId*
Sets the subclass of Cell used to make new Cells
- **setPrototype:***aCell*
Sets the prototype Cell copied to make new Cells
- **prototype**
Returns the prototype Cell copied to make new Cells

Laying Out the Matrix

- **addCol**
Adds a new column of Cells to the bottom of the Matrix
- **addRow**
Adds a new row of Cells to the right of the Matrix
- **insertColAt:**(int)*col*
Inserts a new column of Cells at *col*, creating as many as
needed to make the Matrix *col* columns wide
- **insertRowAt:**(int)*row*
Inserts a new row of Cells at *row*, creating as many as
needed to make the Matrix *row* rows wide

- **removeColAt:**(int)*col* and**Free:**(BOOL)*flag* Removes the column at *col*, freeing the Cells if *flag* is YES
- **removeRowAt:**(int)*row* and**Free:**(BOOL)*flag* Removes the row at *row*, freeing the Cells if *flag* is YES
- **makeCellAt:**(int)*row* :(int)*col* Creates a new Cell at *row*, *col* in the Matrix and returns it
- **putCell:***newCell* at:(int)*row* :(int)*col* Replaces Cell at *row* and *col* with *newCell*; returns old Cell
- **renewRows:**(int)*newRows* cols:(int)*newCols* Changes the number of rows and columns in Matrix without freeing any Cells
- **setCellSize:**(const NXSize *)*aSize* Sets the width and height of all Cells in the Matrix
- **getCellSize:**(NXSize *)*theSize* Gets the width and height of Cells in the Matrix
- **getCellFrame:**(NXRect *)*theRect*
at:(int)*row*
:(int)*col* Returns the frame of the Cell at *row* and *col*
- **setIntercell:**(const NXSize *)*aSize* Sets the vertical and horizontal spacing between Cells
- **getIntercell:**(NXSize *)*theSize* Gets the vertical and horizontal spacing between Cells
- (int)*cellCount* Returns the number of Cells in the Matrix
- **getNumRows:**(int *)*rowCount*
numCols:(int *)*colCount* Gets the number of rows and columns in the Matrix

Finding Matrix Coordinates

- **getRow:**(int *)*row*
and**Col:**(int *)*col*
ofCell:*aCell* Gets the *row* and *col* position of *aCell*
- **getRow:**(int *)*row*
and**Col:**(int *)*col*
forPoint:(const NXPoint *)*aPoint* Gets the *row* and *col* position corresponding to *aPoint*, and returns the Cell at that point

Modifying Individual Cells

- **setIcon:**(const char *)*iconName*
at:(int)*row*
:(int)*col* Sets the icon for the Cell at *row* and *col* to the NXImage named *iconName*
- **setState:**(int)*value* at:(int)*row* :(int)*col* Sets the state of the Cell at *row* and *col* to *value*
- **setTitle:**(const char *)*aString* at:(int)*row* :(int)*col* Assigns Cell at *row* and *col* the title *aString*
- **setTag:**(int)*anInt* at:(int)*row* :(int)*col* Assigns the Cell at *row* and *col* the tag *anInt*
- **setTag:**(int)*anInt*
target:*anObject*
action:(SEL)*aSelector*
at:(int)*row*
:(int)*col* Assigns a tag, target, and action to the specified Cell

Selecting Cells

- **selectCell:***aCell*
Selects the Cell *aCell* if it is in the Matrix
- **selectCellAt:**(int)*row* :(int)*col*
Selects the Cell at *row* and *col*
- **selectCellWithTag:**(int)*anInt*
Selects the Cell with the tag *anInt*
- **setSelectionFrom:**(int)*startPos*
to:(int)*endPos*
anchor:(int)*anchorPos*
lit:(BOOL)*flag*
Selects the Cells in the Matrix from *startPos* to *endPos*, counting in row order from the upper left, as though *anchorPos* were the number of the last Cell selected, and highlighting the Cells according to *flag*
- **selectAll:***sender*
Selects all the Cells in the Matrix
- **selectedCell**
Returns the last (lowest and rightmost) selected Cell
- **getSelectedCells:**(List *)*aList*
Puts the selected Cells into *aList*
- (int)**selectedCol**
Returns the column of the selected Cell
- (int)**selectedRow**
Returns the row of the selected Cell
- **clearSelectedCell**
Deselects the selected Cell

Finding Cells

- **findCellWithTag:**(int)*anInt*
Returns the Cell with *anInt* as its tag
- **cellAt:**(int)*row* :(int)*col*
Returns the Cell at row *row* and column *col*
- **cellList**
Returns the Matrix's List of Cells

Modifying Graphic Attributes

- **setBackgroundColor:**(NXColor)*aColor*
Sets the color of the background between Cells to *aColor*
- (NXColor)**backgroundColor**
Returns the color of the background between Cells
- **setBackgroundGray:**(float)*value*
Sets the gray of the background between Cells to *value*
- (float)**backgroundGray**
Returns the gray of the background between Cells
- **setCellBackgroundColor:**(NXColor)*aColor*
Sets the color of the background within Cells to *aColor*
- (NXColor)**cellBackgroundColor**
Returns the color of the background within Cells
- **setCellBackgroundGray:**(float)*value*
Sets the gray of the background within Cells to *value*
- (float)**cellBackgroundGray**
Returns the gray of the background within Cells
- **setBackgroundTransparent:**(BOOL)*flag*
Sets whether the background between Cells is transparent
- (BOOL)**isBackgroundTransparent**
Returns whether the background between Cells is transparent
- **setCellBackgroundTransparent:**(BOOL)*flag*
Sets whether the background within Cells is transparent
- (BOOL)**isCellBackgroundTransparent**
Returns whether the background within Cells is transparent
- **setFont:***fontObject*
Sets the Font used to display text in the Cells
- **font**
Returns the Font used to display text in the Cells

Editing Text in Cells

- `selectText:sender`
- `selectTextAt:(int)row :(int)col`

Selects the text in the first or last editable Cell
Selects the text of the Cell at *row*, *col* in the Matrix

Setting Tab Key Behavior

- `setNextText:anObject`
- `setPreviousText:anObject`

Sets the object selected when the user hits Tab while editing the last text Cell
Sets the object selected when user hits Shift-Tab while editing the first text Cell

Assigning a Text Delegate

- `setTextDelegate:anObject`
- `textDelegate`

Sets the delegate for messages from the field editor
Returns the delegate for messages from the field editor

Text Object Delegate Methods

- (BOOL)`textWillChange:textObject`
- `textDidChange:textObject`
- `textDidGetKeys:textObject`
 `isEmpty:(BOOL)flag`
- (BOOL)`textWillEnd:textObject`
- `textDidEnd:textObject`
 `endChar:(unsigned short)whyEnd`

Responds to a message from the field editor (see Text)
Responds to a message from the field editor (see Text)
Responds to a message from the field editor (see Text)
Responds to a message from the field editor (see Text)
Responds to a message from the field editor (see Text)

Resizing the Matrix and Cells

- `setAutosizeCells:(BOOL)flag`
- (BOOL)`doesAutosizeCells`
- `calcSize`
- `sizeTo:(float)width :(float)height`
- `sizeToCells`
- `sizeToFit`
- `validateSize:(BOOL)flag`

Sets whether the Matrix resizes its Cells automatically
Returns whether the Matrix resizes its Cells automatically
Calculates Cell sizes
Resizes the Matrix to *width* and *height*
Resizes the Matrix to fit its Cells exactly
Resizes the Cells and Matrix to fit the Cell contents
Sets whether the Cell size needs to be recalculated

Scrolling

- **setAutoScroll:**(BOOL)*flag* Sets whether the Matrix automatically scrolls when dragged in
- **setScrollable:**(BOOL)*flag* Makes all the Cells scrollable
- **scrollCellToVisible:**(int)*row* :(int)*col* Scrolls Matrix so the Cell at *row* and *col* is visible

Displaying

- **display** Draws the Matrix and its Cells
- **drawSelf:**(const NXRect *)*rects* :(int)*rectCount* Draws the Matrix and its Cells
- **drawCell:***aCell* Draws *aCell* if it's in the Matrix
- **drawCellAt:**(int)*row* :(int)*col* Displays the Cell at *row* and *col*
- **drawCellInside:***aCell* Draws the inside of *aCell* if it's in the Matrix
- **highlightCellAt:**(int)*row* :(int)*col* **lit:**(BOOL)*flag* Highlights (or unhighlights) the Cell at *row*, *col*

Target and Action

- **setTarget:***anObject* Sets the target of the Matrix to *anObject*
- **target** Returns the target of the Matrix
- **setAction:**(SEL)*aSelector* Sets the action of the Matrix to *aSelector*
- (SEL)**action** Returns the action of the Matrix
- **setDoubleAction:**(SEL)*aSelector* Sets the action method used on double-clicks to *aSelector*
- (SEL)**doubleAction** Returns the action method for double clicks
- **setErrorAction:**(SEL)*aSelector* Sets the action method for editing errors to *aSelector*
- (SEL)**errorAction** Returns the action method for editing errors
- **setTarget:***anObject* **at:**(int)*row* :(int)*col* Assigns *anObject* as the target of the Cell at *row*, *col*
- **setAction:**(SEL)*aSelector* **at:**(int)*row* :(int)*col* Assigns *aSelector* as the action method of the Cell at *row*, *col*
- **sendAction** Sends the selected Cell's action, or the Matrix's action if the Cell doesn't have one
- **sendAction:**(SEL)*theAction* **to:***theTarget* Has the Application object send *theAction* to *anObject*
- **sendAction:**(SEL)*aSelector* **to:***anObject* **forAllCells:**(BOOL)*flag* Sends *aSelector* to *anObject*, for all Cells if *flag* is YES
- **sendDoubleAction** Sends the action corresponding to a double-click
- **setReaction:**(BOOL)*flag* Sets whether sending an action clears the selection

Handling Event and Action Messages

- (BOOL)**acceptsFirstMouse** Returns NO only if mode is NX_LISTMODE
- **mouseDown:(NXEvent *)theEvent** Responds to a mouse-down event
- (int)**mouseDownFlags** Returns the event flags in effect at start of tracking
- (BOOL)**performKeyEquivalent:(NXEvent *)theEvent** Simulates mouse click in the appropriate Cell

Managing the Cursor

- **resetCursorRects** Resets cursor rectangles so that the cursor becomes an I-beam over text Cells

Archiving

- **read:(NXTypedStream *)stream** Reads the Matrix from *stream*
- **write:(NXTypedStream *)stream** Writes the Matrix to *stream*

Menu

Inherits From: Panel : Window : Responder : Object

Creating a Menu Zone

- + **setMenuZone:(NXZone *)zone** Sets the zone from which Menus should be allocated
- + (NXZone *)**menuZone** Returns the zone from which Menus should be allocated, creating one if necessary

Initializing a New Menu

- **init** Initializes a new Menu with the title "Menu"
- **initWithTitle:(const char *)aTitle** Initializes a new Menu with *aTitle* as its title

Setting Up the Menu Commands

- **addItem:**(const char *)*aString*
 action:(SEL)*aSelector*
 keyEquivalent:(unsigned short)*charCode*
 Adds a new item to the end of the Menu
- **setItemList:***aMatrix*
 Replaces the current Matrix of items with *aMatrix*
- **itemList**
 Returns the Menu's Matrix of MenuCell items

Finding Menu Items

- **findCellWithTag:**(int)*aTag*
 Returns the MenuCell that has *aTag* as its tag

Building Submenus

- **setSubmenu:***aMenu* **forItem:***aCell*
 Makes *aMenu* a submenu controlled by *aCell*
- **submenuAction:***sender*
 Activates a submenu attached to *sender*'s Menu

Managing Menu Windows

- **moveTopLeftTo:**(NXCoord)*x* :(NXCoord)*y*
 Moves the Menu's top left corner to *x*, *y*
- **windowMoved:**(NXEvent *)*theEvent*
 Handles a submenu being torn off its supermenu
- **getLocation:**(NXPoint *)*theLocation*
 forSubmenu:*aSubmenu*
 Determines where to display an attached submenu when it's brought up
- **sizeToFit**
 Resizes the Menu to exactly fit the command items
- **close**
 Removes the Menu (and any submenus) from the screen

Displaying the Menu

- **display**
 Displays the Menu, resizing if needed
- **setAutoupdate:**(BOOL)*flag*
 Sets whether Menu reacts to **update** messages
- **update**
 Updates each MenuCell item

Handling Events

- **mouseDown:**(NXEvent *)*theEvent*
 Tracks the cursor in the Menu and submenus
- **rightMouseDown:**(NXEvent *)*theEvent*
 Pops the main menu up under the cursor

Archiving

- **read:**(NXTypedStream *)*stream*
 Reads the Menu from *stream*
- **write:**(NXTypedStream *)*stream*
 Writes the Menu to *stream*
- **awake**
 Reinitializes a Menu as it's unarchived

MenuCell

Inherits From: ButtonCell : ActionCell : Cell : Object

Initializing a New MenuCell

- **init** Initializes a new MenuCell with “Menu Item” as its title
- **initWithCell:(const char *)aString** Initializes a new MenuCell with *aString* as its title

Setting the Update Action

- **setUpdateAction:(SEL)aSelector
forMenu:aMenu** Sets the update action for the MenuCell to *aSelector*,
and sets *aMenu* to auto-update
- **(SEL)updateAction** Returns the update action for the MenuCell

Checking for a Submenu

- **(BOOL)hasSubmenu** Returns whether the MenuCell has a submenu

Tracking the Mouse

- **(BOOL)trackMouse:(NXEvent *)theEvent
inRect:(const NXRect *)cellFrame
ofView:controlView** Refers mouse tracking to the MenuCell’s Menu

Setting User Key Equivalents

- + **useUserKeyEquivalents:(BOOL)flag** Sets the class to apply user-assigned key equivalents
- **(unsigned short)userKeyEquivalent** Returns the user-assigned key equivalent for the MenuCell

Archiving

- **read:(NXTypedStream *)stream** Reads the MenuCell from *stream*
- **write:(NXTypedStream *)stream** Writes the MenuCell to *stream*

NXBitmapImageRep

Inherits From: NXImageRep : Object

Initializing a New NXBitmapImageRep object

- **initWithSection:**(const char *)*name* Initializes the new object from TIFF data in the section
- **initWithFile:**(const char *)*filename* Initializes the new object from TIFF data in *filename*
- **initWithStream:**(NXStream *)*stream* Initializes the new object from TIFF data in *stream*
- **initWithData:**(unsigned char *)*data* Initializes the new object using data read from an image
 fromRect:(const NXRect *)*rect*
- **initWithData:**(unsigned char *)*data* Initializes the new object from raw bitmap data
 pixelsWide:(int)*width*
 pixelsHigh:(int)*height*
 bitsPerSample:(int)*bps*
 samplesPerPixel:(int)*spp*
 hasAlpha:(BOOL)*alpha*
 isPlanar:(BOOL)*config*
 colorSpace:(NXColorSpace)*space*
 bytesPerRow:(int)*rowBytes*
 bitsPerPixel:(int)*pixelBits*
- **initWithDataPlanes:**(unsigned char **)*planes* Initializes the new object from raw bitmap data in the
 planes data buffers
 pixelsWide:(int)*width*
 pixelsHigh:(int)*height*
 bitsPerSample:(int)*bps*
 samplesPerPixel:(int)*spp*
 hasAlpha:(BOOL)*alpha*
 isPlanar:(BOOL)*config*
 colorSpace:(NXColorSpace)*space*
 bytesPerRow:(int)*rowBytes*
 bitsPerPixel:(int)*pixelBits*

Creating a List of NXBitmapImageReps

- + (List *)**newListFromSection:**(const char *)*name* Returns a List of NXBitmapImageReps from *name* data
- + (List *)**newListFromSection:**(const char *)*name*
 zone:(NXZone *)*aZone* Returns a List of NXBitmapImageReps from *name* data
- + (List *)**newListFromFile:**(const char *)*filename* Returns a List of NXBitmapImageReps from *filename*
- + (List *)**newListFromFile:**(const char *)*filename*
 zone:(NXZone *)*aZone* Returns a List of NXBitmapImageReps from *filename*

- + (List *)**newListFromStream**:(NXStream *)*stream* Returns a List of NXBitmapImageReps from *stream* data
- + (List *)**newListFromStream**:(NXStream *)*stream*
zone:(NXZone *)*aZone* Returns a List of NXBitmapImageReps from *stream* data

Reading Information from a Rendered Image

- + (int)**sizeImage**:(const NXRect *)*rect* Returns the number of bytes in bitmap for the *rect* image
- + (int)**sizeImage**:(const NXRect *)*rect*
pixelsWide:(int *)*width*
pixelsHigh:(int *)*height*
bitsPerSample:(int *)*bps*
samplesPerPixel:(int *)*spp*
hasAlpha:(BOOL *)*alpha*
isPlanar:(BOOL *)*config*
colorSpace:(NXColorSpace *)*space* Provides information about the image bounded by the *rect* rectangle

Copying and Freeing an NXBitmapImageRep

- **copyFromZone**:(NXZone *)*zone* Returns a copy of the NXBitmapImageRep
- **free** Deallocates the NXBitmapImageRep

Getting Information about the Image

- (int)**bitsPerPixel** Returns how many bits are needed to specify one pixel
- (int)**samplesPerPixel** Returns the number of samples (components) in the data
- (BOOL)**isPlanar** Returns YES if in planar configuration, NO if meshed
- (int)**numPlanes** Returns the number of data planes
- (int)**bytesPerPlane** Returns the number of bytes in each data plane
- (int)**bytesPerRow** Returns the number of bytes in a scan line
- (NXColorSpace)**colorSpace** Returns how bitmap data is to be interpreted

Getting Image Data

- (unsigned char *)**data** Returns a pointer to the bitmap data
- **getDataPlanes**:(unsigned char *)*planes* Provides pointers to each plane of bitmap data

Drawing the Image

- (BOOL)**draw** Draws the image at (0.0, 0.0) in current coordinates
- (BOOL)**drawIn**:(const NXRect *)*rect* Modifies coordinates so image is drawn in *rect* rectangle

Producing a TIFF Representation of the Image

- **writeTIFF:**(NXStream *)*stream* Writes a TIFF representation of the image to *stream*
- **writeTIFF:**(NXStream *)*stream*
usingCompression:(int)*compression* Writes a TIFF representation of the image to *stream*
- **writeTIFF:**(NXStream *)*stream*
usingCompression:(int)*compression*
andFactor:(float)*factor* Writes a TIFF representation of the image to *stream*

Setting and Checking Compression Types

- + (void)**getTIFFCompressionTypes:**(const int **)*list*
count:(int *)*numTypes* Returns all available compression types
- + (const char *)**localizedNameForTIFFCompressionType:**(int)*compression*
Returns the localized name for the compression type
- (BOOL)**canBeCompressedUsing:**(int)*compression*
YES if the image can be compressed using *compression*
- (void)**getCompression:**(int *)*compression*
andFactor:(float *)*factor* Returns the compression type and compression factor
- (void)**setCompression:**(int)*compression*
andFactor:(float)*factor* Sets the compression type and compression factor

Archiving

- **read:**(NXTypedStream *)*stream* Reads the NXBitmapImageRep from *stream*
- **write:**(NXTypedStream *)*stream* Writes the NXBitmapImageRep to *stream*

NXBrowser

Inherits From: Control : View : Responder : Object

Initializing and Freeing an NXBrowser

- **initWithFrame:**(const NXRect *)*frameRect* Initializes a new NXBrowser within *frameRect*
- **free** Frees the NXBrowser and its Matrices, NXBrowserCells and other objects (but not the delegate)

Setting the Delegate

- **setDelegate:***anObject*
- **delegate**

Sets the NXBrowser’s delegate to *anObject*
Returns the NXBrowser’s delegate

Target and Action

- **setAction:**(SEL)*aSelector*
- (SEL)**action**
- **setTarget:***anObject*
- **target**
- **setDoubleAction:**(SEL)*aSelector*
- (SEL)**doubleAction**

Sets the NXBrowser’s action method to *aSelector*
Returns the NXBrowser’s action method
Sets the NXBrowser’s target object to *anObject*
Returns the NXBrowser’s target object
Sets the NXBrowser’s double-click action to *aSelector*
Returns the NXBrowser’s double-click action method

Setting Component Classes

- **setMatrixClass:***classId*
- **setCellClass:***classId*
- **setCellPrototype:***aCell*
- **cellPrototype**

Sets the class of Matrix used in the NXBrowser’s columns
Sets the class of Cell used in the columns of NXBrowser
Sets the Cell instance copied to display items in the columns of NXBrowser
Returns the NXBrowser’s prototype Cell

Setting NXBrowser Behavior

- **setMultipleSelectionEnabled:**(BOOL)*flag*
- (BOOL)**isMultipleSelectionEnabled**
- **setBranchSelectionEnabled:**(BOOL)*flag*
- (BOOL)**isBranchSelectionEnabled**
- **setEmptySelectionEnabled:**(BOOL)*flag*
- (BOOL)**isEmptySelectionEnabled**
- **reuseColumns:**(BOOL)*flag*
- **setEnabled:**(BOOL)*flag*
- (BOOL)**acceptsFirstResponder**
- **acceptArrowKeys:**(BOOL)*acceptFlag*
 andSendActionMessages:(BOOL)*sendFlag*
- **getTitleFromPreviousColumn:**(BOOL)*flag*

Sets whether the user can select multiple items
Returns whether the user can select multiple items
Sets whether the user can select branch items when multiple selection is enabled
Returns whether the user can select branch items when multiple selection is enabled
Sets whether there can be nothing selected
Returns whether there can be nothing selected
Prevents Matrices from being freed when their columns are unloaded, so they can be reused
Sets whether the NXBrowser reacts to events
Enables arrow keys for scrolling and sending action messages
Sets whether the title of a column is set to the title of the selected Cell in the previous column

Configuring Controls

- **useScrollBars:**(*BOOL*)*flag* Sets whether Scrollers are used to scroll columns
- **useScrollButtons:**(*BOOL*)*flag* Sets whether buttons are used to scroll columns
- **setHorizontalScrollButtonsEnabled:**(*BOOL*)*flag* Sets whether buttons are used to scroll horizontally
- (*BOOL*)**areHorizontalScrollButtonsEnabled** Returns whether buttons are used to scroll horizontally
- **setHorizontalScrollerEnabled:**(*BOOL*)*flag* Sets whether Scroller is used to scroll horizontally
- (*BOOL*)**isHorizontalScrollerEnabled** Returns whether Scroller is used to scroll horizontally

Setting the NXBrowser's Appearance

- **setMinColumnWidth:**(*int*)*columnWidth* Sets the minimum column width
- (*int*)**minColumnWidth** Returns the minimum column width
- **setMaxVisibleColumns:**(*int*)*columnCount* Sets the maximum number of columns displayed
- (*int*)**maxVisibleColumns** Returns the maximum number of visible columns
- (*int*)**numVisibleColumns** Returns the number of columns visible
- (*int*)**firstVisibleColumn** Returns the index of the first visible column
- (*int*)**lastVisibleColumn** Returns the index of the last visible column
- (*int*)**lastColumn** Returns the index of the last column loaded
- **separateColumns:**(*BOOL*)*flag* Sets whether to separate columns with beveled borders
- (*BOOL*)**columnsAreSeparated** Returns whether columns are separated by beveled borders

Manipulating Columns

- **loadColumnZero** Loads column zero; unloads previously loaded columns
- (*BOOL*)**isLoading** Returns whether column zero is loaded
- **addColumn** Adds a column to the right of the last column
- **reloadColumn:**(*int*)*column* Reloads *column* if it is loaded; sets it as the last column
- **displayColumn:**(*int*)*column* Updates to display columns through index *column*
- **displayAllColumns** Updates the NXBrowser to display all loaded columns
- **setLastColumn:**(*int*)*column* Sets the last column to *column*
- **selectAll:***sender* Selects all Cells in the last column of the NXBrowser
- (*int*)**selectedColumn** Returns the index of the last column with a selected item
- (*int*)**columnOf:***matrix* Returns the column number in which *matrix* is located
- **validateVisibleColumns** Invokes delegate method **browser:columnIsValid:** for visible columns

Manipulating Column Titles

- **setTitle:(BOOL)flag** Sets whether columns display titles
- **(BOOL)isTitled** Returns whether columns display titles
- **setTitle:(const char *)aString ofColumn:(int)column** Sets the title of the column at index *column* to *aString*
- **(const char *)titleOfColumn:(int)column** Returns the title displayed for the column at index *column*
- **(NXRect *)getTitleFrame:(NXRect *)theRect ofColumn:(int)column** Returns the bounds of the title frame for the column at index *column*
- **(NXCoord)titleHeight** Returns the height of column titles
- **drawTitle:(const char *)title inRect:(const NXRect *)aRect ofColumn:(int)column** Draws the title for the column at index *column*
- **clearTitleInRect:(const NXRect *)aRect ofColumn:(int)column** Clears the title for the column at index *column*

Scrolling an NXBrowser

- **scrollColumnsLeftBy:(int)shiftAmount** Scrolls columns left by *shiftAmount* columns
- **scrollColumnsRightBy:(int)shiftAmount** Scrolls columns right by *shiftAmount* columns
- **scrollColumnToVisible:(int)column** Scrolls to make the column at index *column* visible
- **scrollUpOrDown:sender** Scrolls a column up or down
- **scrollViaScroller:sender** Scrolls columns left or right based on a Scroller
- **reflectScroll:clipView** Updates scroll buttons to reflect column contents
- **updateScroller** Updates the horizontal Scroller to reflect column positions

Event Handling

- **mouseDown:(NXEvent *)theEvent** Handles mouse-down events in the NXBrowser
- **keyDown:(NXEvent *)theEvent** Handles key-down events
- **doClick:sender** Responds to mouse clicks in a column of NXBrowser
- **doDoubleClick:sender** Responds to double-clicks in a column of NXBrowser

Getting Matrices and Cells

- **getLoadedCellAtRow:(int)row inColumn:(int)column** Loads if necessary and returns the Cell at *row* in *column*
- **matrixInColumn:(int)column** Returns the matrix located in *column*
- **selectedCell** Returns the last selected Cell (rightmost and lowest)
- **getSelectedCells:aList** Returns in *aList* all the rightmost selected Cells

Getting Column Frames

- (NXRect *)**getFrame:(NXRect *)theRect ofColumn:(int)column** Returns the rectangle containing the column at index *column*
- (NXRect *)**getFrame:(NXRect *)theRect ofInsideOfColumn:(int)column** Returns the rectangle containing the column at index *column*, not including borders

Manipulating Paths

- **setPathSeparator:(unsigned short)charCode** Sets the path separator to *charCode*
- **setPath:(const char *)path** Parses *path* and selects corresponding items in columns
- (char *)**getPath:(char *)thePath toColumn:(int)column** Returns string representing path from the first column to the column at index *column*

Drawing

- **drawSelf:(const NXRect *)rects :(int)rectCount** Draws the NXBrowser

Resizing the NXBrowser

- **sizeTo:(NXCoord)width :(NXCoord)height** Resizes the NXBrowser to *width* and *height*
- **sizeToFit** Resizes the NXBrowser to fit all its contents

Arranging an NXBrowser's Components

- **tile** Adjusts the NXBrowser's components

Methods Implemented by the Delegate

- (BOOL)**browser:sender columnIsValid:(int)column** Returns whether the contents of the column are valid
- **browserDidScroll:sender** Notifies the delegate when the NXBrowser has scrolled
- (int)**browser:sender fillMatrix:matrix inColumn:(int)column** Returns the number of rows in a column and loads NXBrowserCells in *matrix*
- (int)**browser:sender getNumRowsInColumn:(int)column** Returns the number of rows of data in the column at index *column*
- **browser:sender loadCell:cell atRow:(int)row inColumn:(int)column** Requests the delegate to load Cell at *row* in the column at index *column*
- (BOOL)**browser:sender selectCell:(const char *)title inColumn:(int)column** Requests the delegate to select the Cell with title *title* in the column at index *column*

- (const char *)**browser:sender**
titleOfColumn:(int)column
- **browserWillScroll:sender**

Queries the delegate for the title to display above the column at index *column*

Notifies the delegate when the NXBrowser will scroll

NXBrowserCell

Inherits From: Cell : Object

Initializing a NXBrowserCell

- **init** Initializes a new NXBrowserCell with “BrowserItem” as its title
- **initWithCell:(const char *)aString** Initializes a new NXBrowserCell with *aString* as its title

Determining Component Sizes

- **calcCellSize:(NXSize *)theSize**
inRect:(const NXRect *)aRect Calculates the size of the NXBrowserCell within *aRect*

Accessing Graphic Attributes

- (BOOL)**isOpaque** Returns YES, since an NXBrowserCell is opaque
- + **branchIcon** Returns the NXImage for branch NXBrowserCells
- + **branchIconH** Returns the NXImage for highlighted branches

Displaying

- **drawInside:(const NXRect *)cellFrame**
inView:aView Draws the inside of the NXBrowserCell in *aView*
- **drawSelf:(const NXRect *)cellFrame**
inView:aView Draws the entire NXBrowserCell in *aView*
- **highlight:(const NXRect *)cellFrame**
inView:aView lit:(BOOL)lit If *lit* is YES, highlights the NXBrowserCell in *aView*

Placing in Browser Hierarchy

- **setLeaf:(BOOL)flag** Sets whether the NXBrowserCell is a leaf or a branch
- (BOOL)**isLeaf** Returns whether the NXBrowserCell is a leaf or a branch

Determining Loaded Status

- **setLoaded:(BOOL)flag** Sets whether the NXBrowserCell is loaded and displayable
- **(BOOL)isLoading** Returns whether the NXBrowserCell is loaded

Setting State

- **set** Highlights the NXBrowserCell and sets its state to 1
- **reset** Unhighlights the NXBrowserCell and sets its state to 0

NXCachedImageRep

Inherits From: NXImageRep : Object

Initializing a New NXCachedImageRep

- **initWithWindow:(Window *)aWindow rect:(const NXRect *)aRect** Initializes the new NXCachedImageRep for an image to be drawn in *aWindow*
- **copyFromZone:(NXZone *)theZone** Creates and returns a copy of the receiver

Freeing an NXCachedImageRep

- **free** Deallocates the NXCachedImageRep

Getting the Representation

- **getWindow:(Window **)theWindow andRect:(NXRect *)theRect** Provides the Window and rectangle where the image is cached

Drawing the Image

- **(BOOL)draw** Reads the cached image and renders it

Archiving

- **read:(NXTypedStream *)stream** Reads the NXCachedImageRep from *stream*
- **write:(NXTypedStream *)stream** Writes the NXCachedImageRep to *stream*

NXColorPanel

Inherits From: Panel : Window : Responder : Object

Creating a New NXColorPanel

- | | |
|---|--|
| + newColorMask: (int) <i>colormask</i> | Returns the shared NXColorPanel |
| + newContent: (const NXRect *) <i>contentRect</i>
style: (int) <i>aStyle</i>
backing: (int) <i>bufferingType</i>
buttonMask: (int) <i>mask</i>
defer: (BOOL) <i>flag</i> | Returns the shared NXColorPanel |
| + newContent: (const NXRect *) <i>contentRect</i>
style: (int) <i>aStyle</i>
backing: (int) <i>bufferingType</i>
buttonMask: (int) <i>mask</i>
defer: (BOOL) <i>flag</i>
colorMask: (int) <i>colormask</i> | Returns the shared NXColorPanel |
| + sharedInstance: (BOOL) <i>create</i> | If YES, creates if necessary and returns the shared NXColorPanel |

Setting the NXColorPanel

- | | |
|---|---|
| - (int) colorMask | Returns the color mask of the NXColorPanel |
| - setColorMask: (int) <i>colormask</i> | Sets the color mask of the NXColorPanel |
| - setContinuous: (BOOL) <i>flag</i> | Sets the NXColorPanel to continuously send the action message to the target |
| - setMode: (int) <i>mode</i> | Sets the mode and returns the NXColorPanel |
| - setAccessoryView: <i>aView</i> | Sets the accessory view to <i>aView</i> |
| - setAction: (SEL) <i>aSelector</i> | Sets the action message sent to the target |
| - setShowAlpha: (BOOL) <i>flag</i> | Sets the NXColorPanel to show alpha values |
| - setTarget: <i>anObject</i> | Sets the target of the NXColorPanel |

Setting Color

- | | |
|--|--|
| - color: (NXColor *) <i>color</i> | Returns the currently selected color |
| - setColor: (NXColor) <i>color</i> | Sets the <i>color</i> of the NXColorPanel |
| + dragColor: (NXColor *) <i>color</i>
withEvent: (NXEvent *) <i>event</i>
fromView: <i>sourceView</i> | Drags <i>color</i> into a destination view from <i>sourceView</i> .
<i>event</i> is usually an NX_MOUSEUP |

NXColorPicker

Inherits From: Object

Conforms To: NXColorPickingDefault

Initialization

- **initWithPickerMask:(int)theMask
withColorPanel:thePanel** Initializes the receiver for the specified mask and color panel

Button Images

- **provideNewButtonImage** Returns a new button image for the color picker
- **insertNewButtonImage:newImage
in:newButtonCell** Override to customize *newImage* before insertion in *newButtonCell*

View Management

- **viewSizeChanged:sender** Does nothing. Override to respond to size change.

Alpha Control Check

- **alphaControlAddedOrRemoved:sender** Responds to change in color panel alpha control status

Order of Button Appearance

- (float)**insertionOrder** Returns the color picker button's insertion order

Using Color Lists

- **attachColorList:colorList** Override to attach a color list to a color picker
- **detachColorList:colorList** Override to detach a color list from a color picker

Mode

- **setMode:(int)mode** Override to set the color picker's mode

NXColorWell

Inherits From: Control: View: Responder: Object

New

– **initWithFrame:**(const NXRect *)*theFrame* Initializes and returns a new instance of NXColorWell

Event Handling

– (BOOL)**acceptsFirstResponder** Returns YES
– **mouseDown:**(NXEvent *)*theEvent* Responds to mouse down in the NXColorWell

Drawing

– **drawSelf:**(const NXRect *)*rects*
:(int)*rectCount* Draws the entire NXColorWell, including borders
– **drawWellInside:**(const NXRect *)*insideRect* Draws the colored area inside the NXColorWell, without drawing borders

Activating

– **deactivate** Deactivates and returns the NXColorWell
+ **deactivateAllWells** Deactivates all currently active NXColorWells
– **activate:**(int)*exclusive* Activates and returns the NXColorWell
– (BOOL)**isActive** Returns YES if the NXColorWell is active
– **setEnabled:**(BOOL)*enabled* Enables the NXColorWell

Managing Color

– **activeWellsTakeColorFrom:***sender* Changes color of all active wells to that of sender
– **activeWellsTakeColorFrom:***sender*
continuous:(BOOL)*continuously* Continuously changes color of all active, continuous wells to that of *sender*
– (NXColor)**color** Returns the color of the NXColorWell
– **takeColorFrom:***sender* Changes color of the well to that of *sender*
– **acceptColor:**(NXColor)*color*
atPoint:(NXPoint *)*aPoint* Changes color of the well to *color* when *aPoint* is a point in the bounds of the NXColorWell
– **setColor:**(NXColor)*color* Sets the color of the well to *color*
– **updateCustomColorList** Saves the current color list in NX_COLORLISTMODE

Target and Action

- (SEL)**action** Returns the NXColorWell’s action message
- **setAction:**(SEL) *aSelector* Sets the NXColorWell’s action message
- **setTarget:***anObject* Sets the NXColorWell’s target
- **target** Returns the NXColorWell’s target

Archiving

- **awake** Initializes the NXColorWell after unarchiving

NXCursor

Inherits From: Object

Initializing a New NXCursor Object

- **init** Initializes a new NXCursor, but doesn’t set the image
- **initWithImage:***image* Initializes a new NXCursor object with *image*

Defining the Cursor

- **setImage:***newImage* Sets the NXImage object that supplies the cursor image
- **image** Returns the NXImage object that has the cursor image
- **setHotSpot:**(const NXPoint *)*spot* Sets the point on the cursor that’s aligned with the mouse

Setting the Cursor

- **push** Makes the NXCursor the current cursor
- **pop** Restores the previous cursor
- + **pop** Restores the previous cursor
- **set** Sets the NXCursor to be the current cursor
- **setOnMouseEntered:**(BOOL)*flag* Determines whether **mouseEntered:** sets cursor
- **setOnMouseExited:**(BOOL)*flag* Determines whether **mouseExited:** sets cursor
- **mouseEntered:**(NXEvent *)*theEvent* Responds to a mouse-entered event
- **mouseExited:**(NXEvent *)*theEvent* Responds to a mouse-exited event
- + **currentCursor** Returns the current cursor

Archiving

- **read:**(NXTypedStream *)*stream*
- **write:**(NXTypedStream *)*stream*

Reads the NXCursor from the typed stream *stream*
Writes the NXCursor to the typed steam *stream*

NXCustomImageRep

Inherits From: NXImageRep : Object

Initializing a New NXCustomImageRep

- **initWithDrawMethod:**(SEL)*aSelector*
inObject:*anObject*

Initializes the new object so that *anObject*'s *aSelector* method will draw the image

Drawing the Image

- (BOOL)**draw**

Sends a message to draw the image

Archiving

- **read:**(NXTypedStream *)*stream*
- **write:**(NXTypedStream *)*stream*

Reads the NXCustomImageRep from *stream*
Writes the NXCustomImageRep to *stream*

NXEPSImageRep

Inherits From: NXImageRep : Object

Initializing a New NXEPSImageRep Instance

- **initWithSection:**(const char *)*name*
- **initWithFile:**(const char *)*filename*
- **initWithStream:**(NXStream *)*stream*

Initializes the new object from EPS code in the section
Initializes the new object from EPS code in *filename*
Initializes the new object from EPS code in *stream*

Creating a List of NXEPSImageReps

- + (List *)**newListFromSection**:(const char *)*name*
Returns a List of NXEPSImageReps from EPS in *name*
- + (List *)**newListFromSection**:(const char *)*name*
 zone:(NXZone *)*aZone*
Returns a List of NXEPSImageReps from EPS in *name*
- + (List *)**newListFromFile**:(const char *)*filename*
Returns a List of NXEPSImageReps from *filename* data
- + (List *)**newListFromFile**:(const char *)*filename*
 zone:(NXZone *)*aZone*
Returns a List of NXEPSImageReps from *filename* data
- + (List *)**newListFromStream**:(NXStream *)*stream*
Returns a List of NXEPSImageReps from EPS in *stream*
- + (List *)**newListFromStream**:(NXStream *)*stream*
 zone:(NXZone *)*aZone*
Returns a List of NXEPSImageReps from EPS in *stream*

Copying and Freeing an NXEPSImageRep

- **copyFromZone**:(NXZone *)*zone*
Returns a copy of the NXEPSImageRep
- **free**
Deallocates the NXEPSImageRep

Getting the Rectangle that Bounds the Image

- **getBoundingBox**:(NXRect *)*rect*
Copies the EPS bounding box into the *rect* rectangle

Getting Image Data

- **getEPS**:(char **)*theEPS* **length**:(int *)*numBytes*
Provides a pointer to the EPS code

Drawing the Image

- **prepareGState**
Implemented by subclasses to prepare the graphics state
- (BOOL)**draw**
Draws the image at (0.0, 0.0) in current coordinates
- (BOOL)**drawIn**:(const NXRect *)*rect*
Draws the image so it fits within the *rect* rectangle

Archiving

- **read**:(NXTypedStream *)*stream*
Reads the NXEPSImageRep from *stream*
- **write**:(NXTypedStream *)*stream*
Writes the NXEPSImageRep to *stream*

NXHelpPanel

Inherits From: Panel : Window : Responder : Object

Initializing and Freeing

- + **new** Creates, if necessary, and returns the NXHelpPanel object
- + **newForDirectory:(const char *)helpDirectory** Creates, if necessary, and returns the NXHelpPanel object
- **addSupplement:(const char *)helpDirectory
inPath:(const char *)supplementPath** Adds supplemental helpto the text displayed in the panel
- **free** Frees the NXHelpPanel and its storage

Attaching Help to Objects

- + **attachHelpFile:(const char *)filename
markerName:(const char *)markerName
to:anObject** Associates the help file at *markerName* with *anObject*
- + **detachHelpFrom:anObject** Removes any help information associated with *anObject*

Setting Click-for-Help

- + **(BOOL)isClickForHelpEnabled** Returns whether the click-for-help feature is enabled
- + **setClickForHelpEnabled:(BOOL)enabled** Sets whether the click-for-help feature is enabled

Printing

- **print:sender** Prints the currently displayed help text
- **printPanel:sender** Prints the currently displayed help text

Querying

- (NXAtom)**helpDirectory** Returns the absolute path of the help directory
- (NXAtom) **helpFile** Returns the path of the currently loaded help file

Showing Help

- **showFile:(const char *)filename
atMarker:(const char *)markerName** Causes the Help panel to display the help contained in *filename* at *markerName*
- **(BOOL)showHelpAttachedTo:anObject** Causes the Help panel to display help attached to *anObject*

NXImage

Inherits From: Object

Initializing a New NXImage Instance

- **init** Initializes the new NXImage without setting its size
- **initWithSize:(const NXSize *)aSize** Initializes the new NXImage to the specified size
- **initWithSection:(const char *)name** Initializes the new object from the data in *name* section
- **initWithFile:(const char *)filename** Initializes the new NXImage from the data in *filename*
- **initWithPasteboard:(Pasteboard *)pasteboard** Initializes the new NXImage from the data in *pasteboard*
- **initWithStream:(NXStream *)stream** Initializes the new NXImage from the data in *stream*
- **initWithImage:(NXImage *)image
 rect:(const NXRect *)rect** Initializes the new NXImage *to be a subimage of image*
- **copyFromZone:(NXZone *)zone** Creates and returns a copy of the NXImage in *zone*

Freeing an NXImage object

- **free** Frees the NXImage and its representations

Setting the Size of the Image

- **setSize:(const NXSize *)aSize** Sets the size of the image in base coordinates
- **getSize:(NXSize *)theSize** Provides the size of the image

Referring to Images by Name

- (BOOL)**setName:(const char *)string** Assigns *string* as the name of the NXImage object
- (const char *)**name** Returns the name of the NXImage object
- + **findImageNamed:(const char *)name** Returns the NXImage object with *name*

Specifying the Image

- (BOOL)**useDrawMethod:(SEL)aSelector
 inObject:anObject** Creates a representation that will use a delegated method to draw the image
- (BOOL)**useFromSection:(const char *)name** Creates representations for the data in the *name* section
- (BOOL)**useFromFile:(const char *)filename** Creates representations for the data in *filename* file
- (BOOL)**useRepresentation:(NXImageRep *)imageRep** Adds *imageRep* to the List of representations

- (BOOL)**useCacheWithDepth**:(NXWindowDepth)*depth* Creates an empty representation to draw in
- (BOOL)**loadFromStream**:(NXStream *)*stream* Creates representation for the data read from *stream*
- (BOOL)**loadFromFile**:(const char *)*fileName* Creates representation for the data read from *filename*
- (BOOL)**lockFocus** Prepares for drawing in the best representation
- (BOOL)**lockFocusOn**:(NXImageRep *)*imageRep* Prepares for drawing in *imageRep*
- **unlockFocus** Balances a previous **lockFocus** or **lockFocusOn**:

Using the Image

- **composite**:(int)*op* Composites the image to *aPoint*
 toPoint:(const NXPoint *)*aPoint*
- **composite**:(int)*op* Composites the *aRect* portion of the image to *aPoint*
 fromRect:(const NXRect *)*aRect*
 toPoint:(const NXPoint *)*aPoint*
- **dissolve**:(float)*delta* Composites the image using the **dissolve** operator
 toPoint:(const NXPoint *)*aPoint*
- **dissolve**:(float)*delta* Composites the image using the **dissolve** operator
 fromRect:(const NXRect *)*aRect*
 toPoint:(const NXPoint *)*aPoint*

Choosing Which Image Representation to Use

- **setColorMatchPreferred**:(BOOL)*flag* Determines whether color matches are preferred
- (BOOL)**isColorMatchPreferred** Returns whether color matches are preferred
- **setEPSUsedOnResolutionMismatch**:(BOOL)*flag* Sets whether to use EPS representations on mismatch
- (BOOL)**isEPSUsedOnResolutionMismatch** Returns whether to use EPS representations on mismatch
- **setMatchedOnMultipleResolution**:(BOOL)*flag* Sets whether resolution multiples match
- (BOOL)**isMatchedOnMultipleResolution** Returns whether resolution multiples match

Getting the Representations

- (NXImageRep *)**lastRepresentation** Returns the last representation added to the NXImage
- (NXImageRep *)**bestRepresentation** Returns the best representation for the deepest screen
- (List *)**representationList** Returns the List of all the representations
- **removeRepresentation**:(NXImageRep *)*imageRep* Removes *imageRep* from the List of representations

Determining How the Image is Stored

- **setUnique:(BOOL)flag** Sets whether representations are cached alone
- (BOOL)isUnique Returns whether representations are cached alone
- **setDataRetained:(BOOL)flag** Sets whether image data is retained by the object
- (BOOL)isDataRetained Returns whether image data is retained
- **setCacheDepthBounded:(BOOL)flag** Sets whether the default depth limit applies to caches
- (BOOL)isCacheDepthBounded Returns whether the default depth limit applies to caches
- **getImage:(NXImage **)image** Gets the image that the receiver is a subimage of
rect:(NXRect *)rect

Determining How the Image is Drawn

- **setFlipped:(BOOL)flag** Inverts the polarity of the y-axis for drawing the image
- (BOOL)isFlipped Returns whether the polarity of the y-axis is inverted
- **setScalable:(BOOL)flag** Determines whether representations are scaled to fit
- (BOOL)isScalable Returns whether representations are scaled to fit
- **setBackground-color:(NXColor)aColor** Sets the background color of the image
- (NXColor)backgroundColor Returns the background color of the image
- **(BOOL)drawRepresentation:(NXImageRep *)imageRep** Has *imageRep* draw the representation
inRect:(const NXRect *)rect
- **recache** Invalidates caches of all representations, so they will be redrawn

Assigning a Delegate

- **setDelegate:anObject** Makes *anObject* the delegate of the NXImage
- **delegate** Returns the delegate of the NXImage

Producing TIFF Data for the Image

- **writeTIFF:(NXTypedStream *)stream** Writes TIFF for the best representation to *stream*
- **writeTIFF:(NXTypedStream *)stream** Writes TIFF for all the representations to *stream*
allRepresentations:(BOOL)flag

Managing NXImageRep subclasses

- + (void)**registerImageRep:imageRepClass** Registers a new class for managing image data
- + (void)**unregisterImageRep:imageRepClass** Unregisters a class for managing image data
- + (Class)**imageRepForFileType:(const char *)type** Returns image rep that handles data of *type*

- + (Class)**imageRepForPasteboardType:(NXAtom)*type*** Returns image rep that handles data of *type*
- + (Class)**imageRepForStream:(NXStream *)*stream*** Returns image rep that handles data on *stream*

Testing Image Data Sources

- + (BOOL)**canInitFromPasteboard:(Pasteboard *)*pasteboard*** YES if NXImage can create a representation from *pasteboard*
- + (const char *const *)**imageFileTypes** Returns an array of supported image data file types
- + (const NXAtom *)**imagePasteboardTypes** Returns an array of supported pasteboard types

Archiving

- **read:(NXTypedStream *)*stream*** Reads the NXImage and its representations from *stream*
- **write:(NXTypedStream *)*stream*** Writes the NXImage and its representations to *stream*
- **finishUnarchiving** Replaces the NXImage with one having the same name

Methods Implemented by the Delegate

- (NXImage *)**imageDidNotDraw:sender
inRect:(NXRect *)*aRect*** Responds to message that image couldn't be composited

NXImageRep

Inherits From: Object

Initializing

- **initWithPasteboard:(Pasteboard *)*pasteboard*** Initializes the receiver from *pasteboard*

Checking data types

- + (BOOL)**canInitFromPasteboard:(Pasteboard *)*pasteboard*** YES if NXImageRep can initialize itself from *pasteboard*
- + (BOOL)**canLoadFromStream:(NXStream *)*stream*** YES if NXImageRep can initialize itself from *stream*

- + (const char *const *)**imageFileTypes** Returns an array of strings representing all file types
- + (const NXAtom *)**imagePasteboardTypes** Returns an array representing all pasteboard types
- + (const char *const *)**imageUnfilteredFileTypes** Returns an array representing directly supported file types
- + (const NXAtom *)**imageUnfilteredPasteboardTypes** Returns an array representing directly supported pasteboards

Setting the Size of the Image

- **setSize:(const NXSize *)aSize** Sets the size of the image
- **getSize:(NXSize *)theSize** Copies the size of the image into the *theSize* structure

Specifying Information about the Representation

- **setNumColors:(int)anInt** Informs the object that there are *anInt* color components
- (int)**numColors** Returns the number of color components
- **setAlpha:(BOOL)flag** Informs object whether there is a coverage component
- (BOOL)**hasAlpha** Returns whether there is a coverage component
- **setBitsPerSample:(int)anInt** Informs object there are *anInt* bits/pixel in a component
- (int)**bitsPerSample** Returns the number of bits per pixel in each component
- **setPixelsHigh:(int)anInt** Informs object that data is for an image *anInt* pixels high
- (int)**pixelsHigh** Returns the height specified in the image data
- **setPixelsWide:(int)anInt** Informs object that data is for an image *anInt* pixels wide
- (int)**pixelsWide** Returns the width specified in the image data

Drawing the Image

- (BOOL)**draw** Implemented by subclasses to draw the image
- (BOOL)**drawAt:(const NXPoint *)point** Modifies current coordinates so image is drawn at *point*
- (BOOL)**drawIn:(const NXRect *)rect** Modifies current coordinates so image is drawn in *rect*

Archiving

- **read:(NXTypedStream *)stream** Reads the NXImageRep from *stream*
- **write:(NXTypedStream *)stream** Writes the NXImageRep to *stream*

NXJournaler

Inherits From: Object

Initializing and Freeing a Journaler

- **init** Initializes a new NXJournaler
- **free** Deallocates the NXJournaler

Controlling Journaling

- **setEventStatus:(int)eventStatus
soundStatus:(int)soundStatus
eventStream:(NXStream *)stream
soundfile:(const char *)soundfile** Controls recording and playback
- **getEventStatus:(int *)eventStatusPtr
soundStatus:(int *)soundStatusPtr
eventStream:(NXStream **)streamPtr
soundfile:(char **)soundfilePtr** Provides status information about the NXJournaler
- **setRecordDevice:(int)device** Sets whether CODEC or DSP is used for sound input
- **(int)recordDevice** Returns NX_CODEC or NX_DSP

Identifying Associated Objects

- **speaker** Returns the NXJournaler's Speaker object
- **listener** Returns the NXJournaler's Listener object
- **setDelegate:anObject** Sets the NXJournaler's delegate
- **delegate** Returns the NXJournaler's delegate

Implemented by the delegate

- **journalerDidEnd:journaler** Informs the delegate that the session terminated
- **journalerDidUserAbort:journaler** Informs the delegate that the user aborted the session

NXPrinter

Inherits From: Object

Finding an NXPrinter

- + (NXPrinter *)**newForName:**(const char *)*name*
Returns the NXPrinter with the given name
- + (NXPrinter *)**newForName:**(const char *)*name*
host:(const char *)*hostName*
Returns the NXPrinter with the given name *and host*
- + (NXPrinter *)**newForName:**(const char *)*name*
host:(const char *)*hostName*
domain:(const char *)*domain*
includeUnavailable:(BOOL)*includeFlag*
Returns the NXPrinter with the given name, *host, and domain*
- + (NXPrinter *)**newForType:**(const char *)*type*
Returns an NXPrinter object for a given printer type
- + (char **)**printerTypes:**(BOOL)*normalFlag*
custom:(BOOL)*customFlag*
Returns the names of the recognized printer types

Printer Attributes

- (const char *)**domain**
Returns the name of the printer's domain
- (const char *)**host**
Returns the name of the printer's host computer
- (const char *)**name**
Returns the printer's name
- (const char *)**note**
Returns the note associated with the printer
- (const char *)**type**
Returns the name of the printer's type
- (BOOL)**isReallyAPrinter**
Returns whether the object corresponds to an actual printer

Retrieving Specific Information

- (BOOL)**acceptsBinary**
Returns YES if the printer accepts binary PostScript
- (NXRect)**imageRectForPaper:**(const char *)*paperType*
Returns the printing rectangle for the named paper type
- (NXSize)**pageSizeForPaper:**(const char *)*paperType*
Returns the size of the page for the named paper type
- (BOOL)**isColor**
Returns whether the printer can print color
- (BOOL)**isFontAvailable:**(const char *)*name*
Returns whether the named font is available to the printer
- (BOOL)**isValid**
Returns whether the NXPrinter is valid

- (int)**languageLevel** Returns the PostScript Language Level recognized by the printer
- (BOOL)**isOutputStackInReverseOrder** Returns whether the printer outputs pages in reverse page order

Querying the NXPrinter Tables

- (BOOL)**booleanForKey:(const char *)key
inTable:(const char *)table** Returns a boolean value for the given key in the given table
- (void *)**dataForKey:(const char *)key
inTable:(const char *)table
length:(int *)bytes** Returns untyped data for the key in the table
- (float)**floatForKey:(const char *)key
inTable:(const char *)table** Returns a float value for the key in the table
- (int)**intForKey:(const char *)key
inTable:(const char *)table** Returns an integer value for the key in the table
- (NXRect)**rectForKey:(const char *)key
inTable:(const char *)table** Returns an NXRect for the key in the table
- (NXSize)**sizeForKey:(const char *)key
inTable:(const char *)table** Returns an NXSize for the key in the table
- (const char *)**stringForKey:(const char *)key
inTable:(const char *)table** Returns a string for the key in the table
- (const char **)**stringListForKey:(const char *)key
inTable:(const char *)table** Returns an array of strings for the key in the table
- (int)**statusForTable:(const char *)table** Returns the status of the given table
- (BOOL)**isKey:(const char *)key
inTable:(const char *)table** Returns whether key is a key to table

NXSpellChecker

Inherits From: Object

Making A Checker Available

- + **sharedInstance** Returns the NXSpellChecker to use
- + **sharedInstance: (BOOL)flag** Returns the NXSpellChecker to use but creates a new one only when *flag* is YES

Managing The Spelling Panel

- **spellingPanel** Returns the NXSpellChecker’s panel
- **accessoryView** Returns the spell panel’s accessory view
- **setAccessoryView:***aView* Makes a view an accessory of the spell panel

Checking Spelling

- (BOOL) **checkSpelling:**(NXSpellCheckMode)*how*
of:(id <NXReadOnlyTextStream,
NXSelectRange>)*anObject* Starts the search for a misspelled word
- (BOOL) **checkSpelling:**(NXSpellCheckMode)*how*
of:(id <NXReadOnlyTextStream,
NXSelectRange>)*anObject* Starts the search for a misspelled word and the count of
wordCount:(int *)*theCount* words

Managing the Language Being Checked

- (const char*) **language** Returns the current spelling language
- **setLanguage:**(const char *)*aLanguage* Sets the current spelling language

Managing Ignored Words

- **closeSpellClient:***aClient* Notifies the NXSpellChecker that a document has closed
- (char **)**ignoredWordsForSpellClient:***aClient* Returns the list of ignored words for a document
- **setIgnoredWords:**(const char *const *)*someWords*
forSpellClient:(int)*tag* Initializes the list of ignored words for a document

NXSpellServer

Inherits From: Object

Checking in Your Service

- (BOOL)**registerLanguage:**(const char *)*language*
byVendor:(const char *)*vendor*

Assigning a Delegate

- **delegate** Returns the NXSpellServer’s delegate
- **setDelegate:***anObject* Makes the spelling service program the delegate of the NXSpellServer object

Running the Service

- **run** Starts the event loop in the NXSpellServer’s delegate

Checking User Dictionaries

- (BOOL)**isInUserDictionary:**(const char *)*word* Returns YES if the word is in any open user dictionary
caseSensitive:(BOOL)*flag*

Seeking alternative spellings

- **addGuess:**(const char *)*guess* Called by the delegate to append the guesses it has found

Methods Implemented by the Delegate

- (BOOL)**spellServer:**(NXSpellServer *)*sender* Searches for a misspelled word; return YES if one is found
findMisspelledWord:(int *)*start*
length:(int *)*length*
inLanguage:(const char *)*language*
inTextStream:(id <NXReadOnlyTextStream>)*textStream*
startingAt:(int)*startPosition*
wordCount:(int *)*number*
countOnly:(BOOL)*flag*
- (void)**spellServer:**(NXSpellServer *)*sender* Searches for alternatives to the misspelled word; returns
suggestGuessesForWord:(const char *)*word* guesses as a side effect, using **addGuess:**
inLanguage:(const char *)*language*
- (void)**spellServer:**(NXSpellServer *)*sender* Notifies the delagte of a word added to the user’s hidden
didLearnWord:(const char *)*word* wordlist
inLanguage:(const char *)*language*
- (void)**spellServer:**(NXSpellServer *)*sender* Notifies the delagte of a word removed from the user’s
didForgetWord:(const char *)*word* hidden wordlist
inLanguage:(const char *)*language;*

NXSplitView

Inherits From: View : Responder : Object

Initializing an NXSplitView

– **initWithFrame:**(const NXRect *)*frameRect* Initializes a new NXSplitView

Handling Events

– **mouseDown:**(NXEvent *)*theEvent* Handles mouse-down events
– **acceptsFirstMouse** Allows the NXSplitView to respond to the mouse event that makes its Window the key window

Managing Component Views

– **adjustSubviews** Adjusts the heights of the subviews
– **resizeSubviews:** Forces adjustment of the subviews
– **(NXCoord)dividerHeight** Returns the height of the divider
– **drawSelf:**(const NXRect *) *rects* :(int)*rectCount* Draws the NXSplitView
– **drawDivider:**(const NXRect *)*aRect* Draws the divider
– **setAutoresizeSubviews:**(BOOL)*flag* Ensures that the subviews are automatically resized

Assigning a Delegate

– **setDelegate:***anObject* Sets the NXSplitView's delegate
– **delegate** Returns the NXSplitView's delegate

Implemented by the Delegate

– **splitViewDidResizeSubviews:***sender* Informs the delegate that subviews were resized
– **splitView:***sender* Limits divider travel
 getMinY:(NXCoord *)*minY*
 getMaxY:(NXCoord *)*maxY*
 offsetSubviewAt:(int)*offset*
– **splitView:***sender* Allows custom resizing behavior
 resizeSubviews:(const NXSize *)*oldSize*

Object Additions

This method is declared in the Application Kit as an addition to the root Object class.

Sending Messages Determined at Run Time

- **perform:**(SEL)*aSelector* Sends an *aSelector* message to the receiver after *ms* delay
- with:***anObject*
- afterDelay:**(int)*ms*
- cancelPrevious:**(BOOL)*flag*

OpenPanel

Inherits From: SavePanel : Panel : Window : Responder : Object

Creating and Freeing an OpenPanel

- + **new** Returns the shared OpenPanel object
- + **newContent:**(const NXRect *)*contentRect* Returns the shared OpenPanel object
- style:**(int)*aStyle*
- backing:**(int)*bufferingType*
- buttonMask:**(int)*mask*
- defer:**(BOOL)*flag*
- **free** Deallocates the OpenPanel object

Setting the OpenPanel Class

- + **setOpenPanelFactory:***class* Sets class for initializing an OpenPanel

Filtering Files

- **allowMultipleFiles:**(BOOL)*flag* Sets whether the user can open multiple files

Querying the Chosen Files

- (const char *const *)**filenames** Gets the names of the selected files

Running the OpenPanel

- (int)**runModalForDirectory**:(const char *)*path* Displays the panel and begins its event loop
 file:(const char *)*name*
- (int)**runModalForDirectory**:(const char *)*path* Displays the panel and begins its event loop
 file:(const char *)*name*
 types:(const char *const *)*fileTypes*
- (int)**runModalForTypes**:(const char *const *)*fileTypes* Displays the panel and begins its event loop

PageLayout

Inherits From: Panel : Window : Responder : Object

Creating and Freeing a PageLayout Instance

- + **new** Returns a default PageLayout object
- + **newContent**:(const NXRect *)*contentRect* Used in PageLayout instantiation
 style:(int)*aStyle*
 backing:(int)*bufferingType*
 buttonMask:(int)*mask*
 defer:(BOOL)*flag*
- **free** Deallocates the PageLayout panel

Running the PageLayout Panel

- (int)**runModal** Displays the panel and begins its event loop

Customizing the PageLayout Panel

- **setAccessoryView**:*aView* Adds a View to the panel
- **accessoryView** Returns the PageLayout's accessory View

Updating the Panel's Display

- **pickedLayout**:*sender* Updates the panel when a new layout is selected
- **pickedOrientation**:*sender* Updates the panel with the selected orientation
- **pickedPaperSize**:*sender* Updates the panel when a paper size is selected

- **pickedUnits:***sender* Updates the panel when a new unit is selected
- **textDidEnd:***textObject*
 endChar:(unsigned short)*theChar* Updates the panel when the user finishes typing a page size
- (BOOL)**textWillChange:***textObject* Updates the panel when a page size is typed
- **convertOldFactor:**(float *)*old*
 newFactor:(float *)*new* Converts units for **pickedUnits:** method
- **pickedButton:***sender* Stops the event loop

Communicating with the PrintInfo Object

- **readPrintInfo** Reads the PageLayout’s values from the PrintInfo object
- **writePrintInfo** Writes the PageLayout’s values to the PrintInfo object

Panel

Inherits From: Window : Responder : Object

Initializing a New Panel

- **init** Initializes the new Panel with default values
- **initWithContent:**(const NXRect *)*contentRect*
 style:(int)*aStyle*
 backing:(int)*bufferingType*
 buttonMask:(int)*mask*
 defer:(BOOL)*flag* Initializes the new Panel as specified

Handling Events

- (BOOL)**commandKey:**(NXEvent *)*theEvent* Initiates **performKeyEquivalent:** messages
- **keyDown:**(NXEvent *)*theEvent* Convert key-down event to a **commandKey:** message

Determining the Panel Interface

- **setBecomeKeyOnlyIfNeeded:**(BOOL)*flag* Sets whether Panel waits to become key window
- (BOOL)**doesBecomeKeyOnlyIfNeeded** Returns whether Panel waits to become key window
- **setFloatingPanel:**(BOOL)*flag* Sets whether the Panel floats above other windows

- (BOOL)isFloatingPanel Returns whether the Panel floats above other windows
- setWorksWhenModal:(BOOL)flag Sets whether the Panel can operate on an attention panel
- (BOOL)worksWhenModal Returns whether Panel can operate on an attention panel

Pasteboard

Inherits From: Object

Creating and Freeing a Pasteboard

- + new Returns the selection Pasteboard object
- + newName:(const char *)name Returns the Pasteboard object named *name*
- + newUnique Creates a uniquely named Pasteboard
- free Releases the Pasteboard object's storage
- freeGlobally Frees the object and the domain for its name

Getting Data in Different Formats

- + newByFilteringFile:(const char *)filename Creates a pasteboard with all types for *filename*
- + newByFilteringData:(NXData *)data
ofType:(const char *)type Creates a pasteboard with all types for *data*
- + newByFilteringTypesInPasteboard:
(Pasteboard *)pboard Creates a pasteboard with all types filterable
from *pboard*
- + (NXAtom *)typesFilterableTo:
(const char *)type Returns all types *type* can be filtered to

Referring to a Pasteboard by Name

- + newName:(const char *)name Returns the Pasteboard object named *name*
- (const char *)name Returns the Pasteboard object's name

Writing Data

- declareTypes:(const char *const *)newTypes
num:(int)numTypes
owner:newOwner Sets data types and owner of the Pasteboard

- (int)**addTypes:(const char *const *)newTypes**
num:(int)numTypes
owner:newOwner Adds data types to the pasteboard
- **writeType:(const char *)dataType**
data:(const char *)theData
length:(int)numBytes Writes *theData* to the pasteboard server
- **writeType:(const char *)dataType**
fromStream:(NXStream *)stream Writes stream data to the pasteboard server
- (BOOL)**writeFileContents:**
(const char *)filename Writes data from *filename* to the pasteboard server

Discerning Types

- (const NXAtom *)**types** Returns an array of the Pasteboard's data types
- (const char *)**findAvailableTypeFrom:**
(const char *const *)types Returns first type in *types* that matches a pasteboard type

Reading Data

- (int)**changeCount** Returns the Pasteboard's change count
- **readType:(const char *)dataType**
data:(char **)theData
length:(int *)numBytes Reads data from the pasteboard server
- (NXStream *)**readTypeToStream:**
(const char *)dataType Returns a stream to pasteboard data
- (char *)**readFileContentsType:**
(const char *)type
toFile:(const char *)filename Writes pasteboard data to a file
- **deallocatePasteboardData:(char *)data**
length:(int)numBytes Deallocates data received from the pasteboard

Methods Implemented by the Owner

- **pasteboard:sender**
provideData:(NXAtom)type Implemented to write promised data to *sender* as *type*
- **pasteboardChangedOwner:sender** Notifies prior owner that ownership changed

PopUpList

Inherits From: Menu : Panel : Window : Responder : Object

Initializing a PopUpList

– **init** Initializes a new PopUpList

Setting Up the Items

– **addItem:(const char *)title** Adds an item with *title* as its title to the end of the list
– **insertItem:(const char *)title
at:(unsigned int)index** Inserts an item with *title* as its title at position *index*
– **removeItem:(const char *)title** Removes the item matching *title*
– **removeItemAt:(unsigned int)index** Removes the item at the specified *index*
– **(int)indexOfItem:(const char *)title** Returns the index of the item matching *title*
– **(unsigned int)count** Returns the number of items in the list

Interacting with the Trigger Button

– **changeButtonTitle:(BOOL)flag** Sets whether the PopUpList is a pop-up or a pull-down list
– **getButtonFrame:(NXRect *)bFrame** Gets the size needed for the Button that pops up the list

Activating the PopUpList

– **popUp:trigger** Pops the list up over *trigger*

Returning the User's Selection

– **(const char *)selectedItem** Returns the title of selected item

Modifying the Items

– **setFont:fontObject** Sets the Font used to draw the items
– **font** Returns the Font used to draw the items

Target and Action

- **setAction:**(SEL)*aSelector* Sets the PopUpList’s action method to *aSelector*
- (SEL)**action** Returns the PopUpList’s action method
- **setTarget:***anObject* Sets the PopUpList’s target object to *anObject*
- **target** Returns the PopUpList’s target object

Resizing the PopUpList

- **sizeWindow:**(NXCoord)*width* :(NXCoord)*height* Resizes the PopUpList to *width*, *height*

PrintInfo

Inherits From: Object

Initializing and Freeing a PrintInfo Instance

- **init** Initializes the PrintInfo instance after it’s allocated
- **free** Deallocates the PrintInfo object

Defining the Printing Rectangle

- **setMarginLeft:**(NXCoord)*leftMargin*
right:(NXCoord)*rightMargin*
top:(NXCoord)*topMargin*
bottom:(NXCoord)*bottomMargin* Sets the margins
- **getMarginLeft:**(NXCoord *)*leftMargin*
right:(NXCoord *)*rightMargin*
top:(NXCoord *)*topMargin*
bottom:(NXCoord *)*bottomMargin* Returns the margins by reference
- **setOrientation:**(char)*mode*
andAdjust:(BOOL)*flag* Sets the orientation as portrait or landscape
- (char)**orientation** Returns the orientation is portrait or landscape
- **setPaperRect:**(const NXRect *)*aRect*
andAdjust:(BOOL)*flag* Sets the width and height of the paper
- (const NXRect *)**paperRect** Returns the rectangle for the paper size
- **setPaperType:**(const char *)*type*
andAdjust:(BOOL)*flag* Sets the paper type
- (const char *)**paperType** Returns the paper type

Page Range

- **setFirstPage:**(int)*anInt* Sets the page number of first page to be printed
- (int)**firstPage** Returns the page number of the first page to be printed
- **setLastPage:**(int)*anInt* Sets the page number of last page to be printed
- (int)**lastPage** Returns the page number of the last page to be printed
- **setAllPages:**(BOOL)*flag* Sets whether all the pages are to be printed
- (BOOL)**isAllPages** Returns whether all the pages are to be printed
- (int)**currentPage** Returns the page number of the page being printed

Pagination and Scaling

- **setHorizPagination:**(int)*mode* Sets the horizontal pagination mode
- (int)**horizPagination** Returns the horizontal pagination mode
- **setVertPagination:**(int)*mode* Sets the vertical pagination mode
- (int)**vertPagination** Returns the vertical pagination mode
- **setScalingFactor:**(float)*aFloat* Sets the scaling factor
- (float)**scalingFactor** Returns the scaling factor

Positioning the Image on the Page

- **setHorizCentered:**(BOOL)*flag* Sets whether the image is centered horizontally
- (BOOL)**isHorizCentered** Returns whether the image is centered horizontally
- **setVertCentered:**(BOOL)*flag* Sets whether the image is centered vertically
- (BOOL)**isVertCentered** Returns whether the image is centered vertically
- **setPagesPerSheet:**(short)*aShort* Sets the number of pages printed per sheet of paper
- (short)**pagesPerSheet** Returns the number of pages printed per sheet of paper

Print Job Attributes

- **initializeJobDefaults** Invoked automatically to initialize printing defaults
- **setJobFeature:**(const char *)*feature*
 toValue:(const char *)*value* Sets the value of the given printing job feature
- (const char *)**valueForJobFeature:**(const char *)*feature*
Returns the value for the given printing job feature
- **removeJobFeature:**(const char *)*key* Removes the given printing job feature
- (const char **)**jobFeatures** Returns the keys to the job features table
- **setPageOrder:**(char)*mode* Sets the order in which pages will be printed
- (char)**pageOrder** Returns the order in which pages will be printed
- **setReversePageOrder:**(BOOL)*flag* Sets whether the page order is reversed
- (BOOL)**reversePageOrder** Returns whether the page order is reversed

- **setCopies:**(int)*anInt* Sets the number of copies to be printed
- (int)**copies** Returns the number of copies to be printed
- **setPaperFeed:**(const char *)*paperFeedSlot* Sets the paper feed slot used during printing
- (const char *)**paperFeed** Returns the paper feed slot used during printing

Specifying the Printer

- + **setDefaultPrinter:**(NXPrinter *)*printer* Sets the user's default printer
- + (NXPrinter *)**getDefaultPrinter** Returns the user's default printer
- **setPrinter:**(NXPrinter *)*aPrinter* Sets the printer that's used in subsequent printing jobs
- (NXPrinter *)**printer** Returns the NXPrinter that's used for printing

Spooling

- **setOutputFile:**(const char *)*aString* Sets the output file for printing
- (const char *)**outputFile** Returns the output file for printing
- **setContext:**(DPSContext)*aContext* Sets the DPS context used for printing
- (DPSContext)**context** Returns the DPS context used for printing

Archiving

- **read:**(NXTypedStream *)*stream* Reads the PrintInfo from the typed stream
- **write:**(NXTypedStream *)*stream* Writes the PrintInfo to the typed stream

PrintPanel

Inherits From: Panel : Window : Responder : Object

Creating and Freeing a PrintPanel

- + **new** Returns a default PrintPanel object
- + **newContent:**(const NXRect *)*contentRect* Returns a PrintPanel object
 - style:**(int)*aStyle*
 - backing:**(int)*bufferingType*
 - buttonMask:**(int)*mask*
 - defer:**(BOOL)*flag*
- **free** Deallocates the PrintPanel

Customizing the PrintPanel

- **setAccessoryView:***aView* Adds a View to the panel
- **accessoryView** Returns the accessory View

Running the Panel

- **(int)runModal** Displays the Print panel and begins its event loop
- **pickedButton:***sender* Stops the event loop

Updating the Panel's Display

- **pickedAllPages:***sender* Updates the panel when the user chooses all pages
- **(BOOL)textWillChange:***textObject* Updates the panel when user types pages to print

Communicating with the PrintInfo Object

- **updateFromPrintInfo** Reads PrintPanel's values from the PrintInfo object
- **finalWritePrintInfo** Writes PrintPanel's values to the PrintInfo object

Responder

Inherits From: Object

Managing the NeXT Responder

- **setNextResponder:***aResponder* Makes *aResponder* the receiver's next responder
- **nextResponder** Returns the receiver's next responder

Determining the First Responder

- **(BOOL)acceptsFirstResponder** Returns NO to refuse first responder status
- **becomeFirstResponder** Notifies the receiver it's the first responder
- **resignFirstResponder** Notifies the receiver it's not the first responder

Aiding Event Processing

- (BOOL)performKeyEquivalent:(NXEvent *)*theEvent*
Returns NO to indicate *theEvent* isn't handled
- (BOOL)tryToPerform:(SEL)*anAction*
with:*anObject*
Aids in dispatching action messages

Forwarding Event Messages

- mouseDown:(NXEvent *)*theEvent*
Passes the message to the receiver's next responder
- rightMouseDown:(NXEvent *)*theEvent*
Passes the message to the receiver's next responder
- mouseDragged:(NXEvent *)*theEvent*
Passes the message to the receiver's next responder
- rightMouseDown:(NXEvent *)*theEvent*
Passes the message to the receiver's next responder
- mouseUp:(NXEvent *)*theEvent*
Passes the message to the receiver's next responder
- rightMouseUp:(NXEvent *)*theEvent*
Passes the message to the receiver's next responder
- mouseMoved:(NXEvent *)*theEvent*
Passes the message to the receiver's next responder
- mouseEntered:(NXEvent *)*theEvent*
Passes the message to the receiver's next responder
- mouseExited:(NXEvent *)*theEvent*
Passes the message to the receiver's next responder
- keyDown:(NXEvent *)*theEvent*
Passes the message to the receiver's next responder
- keyUp:(NXEvent *)*theEvent*
Passes the message to the receiver's next responder
- flagsChanged:(NXEvent *)*theEvent*
Passes the message to the receiver's next responder
- noResponderFor:(const char *)*eventType*
Prints warning message to syslog if debugging

Services Menu Support

- validRequestorForSendType:(NXAtom)*typeSent*
andReturnTypes:(NXAtom)*typeReturned*
Implemented by subclasses to determine available services

Archiving

- read:(NXTypedStream *)*stream*
Reads the Responder from the typed stream *stream*
- write:(NXTypedStream *)*stream*
Writes the Responder to the typed stream *stream*

SavePanel

Inherits From: Panel : Window : Responder : Object

Creating and Freeing a SavePanel

- + **newContent:**(const NXRect *)*contentRect* Creates and returns a SavePanel object
 style:(int)*aStyle*
 backing:(int)*bufferingType*
 buttonMask:(int)*mask*
 defer:(BOOL)*flag*
- **free** Deallocates the SavePanel

Setting the SavePanel Class

- + **setSavePanelFactory:***class* Sets class for initializing an SavePanel

Customizing the SavePanel

- **setAccessoryView:***aView* Adds application-customized view to the panel
- **accessoryView** Returns the application-customized view
- **setTitle:**(const char *)*title* Sets the title of the SavePanel to *title*
- **setPrompt:**(const char *)*prompt* Sets the title of the file name form field

Setting Directory and File Type

- **setDirectory:**(const char *)*path* Sets the current directory of the SavePanel
- **setRequiredFileType:**(const char *)*type* Sets the required file type (if any)
- (const char *)**requiredFileType** Gets the required file type (if any)

Running the SavePanel

- (int)**runModalForDirectory:**(const char *)*path* Displays the SavePanel and begins its event loop
 file:(const char *)*name*
- (int)**runModal** Displays the SavePanel and begins its event loop

Reading Save Information

- (const char *)**directory** Returns directory chosen file resides in
- (const char *)**filename** Returns full name of file to be saved

Completing a Partial Filename

- (BOOL)commandKey:(NXEvent *)theEvent Enables command-space to do filename completion

Target and Action Methods

- ok:sender Method invoked by the OK button
- cancel:sender Method invoked by the Cancel button

Responding to User Input

- selectText:sender Called when TAB is pressed in the form
- textDidEnd:textObject
endChar:(unsigned short)endChar Determines whether TAB or BACKTAB was pressed
- textDidGetKeys:textObj isEmpty:(BOOL)flag Determines whether there's any text in the form

Setting the Delegate

- setDelegate:anObject Makes anObject the SavePanel's delegate

Methods implemented by the Delegate

- (int)panel:sender
compareFileNames:(const char *)fileName1
:(const char *)fileName2
checkCase:(BOOL)flag Returns 1 if *fileName1* precedes *fileName2*, -1 in the opposite case, 0 if the two are equivalent
- (BOOL)panel:sender
filterFile:(const char *)filename
inDirectory:(const char *)directory YES if *filename* can be saved in directory
- (BOOL)panelValidateFileNames:sender YES if the filename is acceptable to the delegate

Scroller

Inherits From: Control : View : Responder : Object

Initializing a Scroller

- initWithFrame:(const NXRect *)frameRect Initializes a new Scroller

Laying out the Scroller

- (NXRect *)**calcRect**:(NXRect *)*aRect*
 forPart:(int)*partCode* Gets the rectangle that encloses *partCode*
- **checkSpaceForParts** Checks for room for knob and scroll buttons
- **setArrowsPosition**:(int)*where* Sets position of scroll buttons in Scroller

Setting Scroller values

- (float)**floatValue** Returns Scroller's float value
- **setFloatValue**:(float)*aFloat* Sets value; positions knob
- **setFloatValue**:(float)*aFloat* :(float)*percent* Sets value; positions and sizes knob

Resizing the Scroller

- **sizeTo**:(NXCoord)*width* :(NXCoord)*height* Sizes the Scroller

Displaying

- **drawArrow**:(BOOL)*whichButton* :(BOOL)*flag* Draws highlighted and unhighlighted arrows
- **drawKnob** Draws the knob
- **drawParts** Caches Bitmaps for knob and scroll arrows
- **drawSelf**:(const NXRect *)*rects* :(int)*rectCount* Draws the Scroller
- **highlight**:(BOOL)*flag* Highlights scroll button that's under mouse

Target and Action

- **setAction**:(SEL)*aSelector* Sets the Scroller's action to *aSelector*
- (SEL)**action** Returns the Scroller's action
- **setTarget**:*anObject* Sets the Scroller's target to *anObject*
- **target** Returns the Scroller's target

Handling Events

- (BOOL)**acceptsFirstMouse** Makes the Scroller respond to the first mouse event
- (int)**hitPart** Returns Scroller part that received mouse-down
- **mouseDown**:(NXEvent *)*theEvent* Responds to mouse-down events
- (int)**testPart**:(const NXPoint *)*thePoint* Returns Scroller part that's under *thePoint*
- **trackKnob**:(NXEvent *)*theEvent* Responds to mouse-down events on the knob
- **trackScrollButtons**:(NXEvent *)*theEvent* Responds to mouse-down events on buttons

Archiving

- **awake** Ensures that Scroller's Bitmaps are created
- **read:(NXTypedStream *)stream** Reads the Scroller from the typed stream
- **write:(NXTypedStream *)stream** Writes the Scroller to the typed stream

ScrollView

Inherits From: View : Responder : Object

Initializing a ScrollView

- **initWithFrame:(const NXRect *)frameRect** Initializes a new ScrollView

Determining Component Sizes

- **getContentSize:(NXSize *)contentViewSize** Gets the content view's size
- **getDocVisibleRect:(NXRect *)aRect** Gets the visible portion of the document view

Laying Out the ScrollView

- + **getContentSize:(NXSize *)cSize** Gets the content view size for the given ScrollView size
 forFrameSize:(const NXSize *)fSize
 horizScroller:(BOOL)hFlag
 vertScroller:(BOOL)vFlag
 borderType:(int)aType
- + **getFrameSize:(NXSize *)fSize** Gets the ScrollView size for the given content view size
 forContentSize:(const NXSize *)cSize
 horizScroller:(BOOL)hFlag
 vertScroller:(BOOL)vFlag
 borderType:(int)aType
- **resizeSubviews:(const NXSize *)oldSize** Retiles the ScrollView after a **sizeTo::**
- **setHorizScrollerRequired:(BOOL)flag** Makes space for a horizontal scroller
- **setVertScrollerRequired:(BOOL)flag** Makes space for a vertical scroller
- **tile** Retiles the scrollers and content view

Managing Component Views

- **setDocView:***aView* Makes *aView* the ScrollView's document view
- **docView** Returns the current document view
- **setHorizScroller:***anObject* Sets the horizontal Scroller object
- **horizScroller** Returns the horizontal Scroller
- **setVertScroller:***anObject* Sets the vertical Scroller object
- **vertScroller** Returns the vertical Scroller
- **reflectScroll:***cView* Updates the Scrollers

Modifying Graphic Attributes

- **setBorderType:**(int)*aType* Determines the border type of the ScrollView
- (int)**borderType** Returns the border type
- **setBackgroundColor:**(NXColor)*color* Sets the ScrollView's background color
- (NXColor) **backgroundColor** Returns the ScrollView's background color
- **setBackgroundGray:**(float)*value* Sets the ScrollView's background gray
- (float)**backgroundGray** Returns the ScrollView's background gray

Setting Scrolling Behavior

- **setCopyOnScroll:**(BOOL)*flag* Sets how newly exposed areas are redrawn
- **setDisplayOnScroll:**(BOOL)*flag* Sets how the doc view is displayed during scrolling
- **setDynamicScrolling:**(BOOL)*flag* Sets how the doc view is displayed during scrolling
- **setLineScroll:**(float)*value* Sets the amount to scroll when scrolling a line
- **setPageScroll:**(float)*value* Sets the amount of overlap for a page scroll

Displaying

- **drawSelf:**(const NXRect *)*rects* :(int)*rectCount* Draws the ScrollView

Managing the Cursor

- **setDocCursor:***anObj* Sets the cursor for the document view

Archiving

- **read:**(NXTypedStream *)*stream* Reads the ScrollView from the typed stream
- **write:**(NXTypedStream *)*stream* Writes the ScrollView to the typed stream

SelectionCell

Inherits From: Cell : Object

Creating a SelectionCell

- **init** Initializes a new SelectionCell with “ListItem” as its title
- **initWithCell:(const char *)aString** Initializes a new SelectionCell with *aString* as its title

Determining Component Sizes

- **calcCellSize:(NXSize *)theSize
inRect:(const NXRect *)aRect** Calculates the size of the SelectionCell within *aRect*

Accessing Graphic Attributes

- **setLeaf:(BOOL)flag** Sets whether SelectionCell is a leaf or a branch
- **(BOOL)isLeaf** Returns whether the SelectionCell is a leaf or a branch
- **(BOOL)isOpaque** Returns YES, since SelectionCells are opaque

Displaying

- **drawSelf:(const NXRect *)cellFrame
inView:aView** Draws the SelectionCell in *cellFrame* within *aView*
- **drawInside:(const NXRect *)cellFrame
inView:aView** Draws the inside of the SelectionCell in *aView*
- **highlight:(const NXRect *)cellFrame
inView:aView
lit:(BOOL)flag** Highlights the SelectionCell within *cellFrame* in *controlView*

Archiving

- **awake** Reinitializes the SelectionCell when it’s unarchived

Slider

Inherits From: Control : View : Responder : Object

Setting Slider's Cell Class

+ **setCellClass:***classId* Sets the subclass of SliderCell used by Slider

Initializing a new Slider

– **initWithFrame:**(const NXRect *)*frameRect* Initializes a new Slider in *frameRect*

Modifying a Slider's appearance

– **setKnobThickness:**(NXCoord)*aFloat* Sets the knob's thickness to *aFloat*
– (NXCoord)**knobThickness** Returns the knob's thickness
– **setImage:***image* Sets the background image to *image*
– **image** Returns the background image
– **setTitle:**(const char *)*aString* Sets the background title to a copy of *aString*
– **setTitleNoCopy:**(const char *)*aString* Sets the background title to *aString*
– (const char *)**title** Returns the background title
– **setTitleCell:***aCell* Sets the Cell used to draw the background title
– **titleCell** Returns the Cell used to draw the background title
– **setTitleFont:***fontObject* Sets the Font used to draw the background title
– **titleFont** Returns the Font used to draw the background title
– **setTitleColor:**(NXColor)*aColor* Sets the color of text in the background title to *aColor*
– (NXColor)**titleColor** Returns the color of text in the background title
– **setTitleGray:**(float)*aFloat* Sets the gray of text in the background title to *aFloat*
– (float)**titleGray** Returns the gray of text in the background title
– (int)**isVertical** Returns 1 if vertical, 0 if horizontal, –1 if unknown

Setting Value Limits

– **setMinValue:**(double)*aDouble* Sets the Slider's minimum value to *aDouble*
– (double)**minValue** Returns the Slider's minimum value
– **setMaxValue:**(double)*aDouble* Sets the Slider's maximum value to *aDouble*
– (double)**maxValue** Returns the Slider's maximum value

Resizing the Slider

– **sizeToFit**

Modifies the Slider's size to fit its Cell

Handling Events

– (BOOL)**acceptsFirstMouse**

Returns YES, since Sliders always accept first mouse

– **setEnabled:(BOOL)flag**

Sets whether the Slider reacts to events

– **mouseDown:(NXEvent *)theEvent**

Responds to mouse-down by initiating tracking

SliderCell

Inherits From: ActionCell : Cell : Object

Initializing a new SliderCell

– **init**

Initializes a new SliderCell

Determining Component Sizes

– **calcCellSize:(NXSize *)theSize
inRect:(const NXRect *)aRect**

Returns the size of the SliderCell

– **getKnobRect:(NXRect*)knobRect
flipped:(BOOL)flipped**

Gets the rectangle the knob will be drawn in

Setting Value Limits

– **setMinValue:(double)aDouble**

Sets the SliderCell's minimum value to *aDouble*

– (double)**minValue**

Returns the SliderCell's minimum value

– **setMaxValue:(double)aDouble**

Sets the maximum value of the SliderCell to *aDouble*

– (double)**maxValue**

Returns the SliderCell's maximum value

Setting Values

– **setDoubleValue:(double)aDouble**

Sets the SliderCell's value to *aDouble*

– (double)**doubleValue**

Returns the SliderCell's value as a **double**

– **setFloatValue:(float)aFloat**

Sets the SliderCell's value to *aFloat*

– (float)**floatValue**

Returns the SliderCell's value as a **float**

– **setIntValue:(int)anInt**

Sets the SliderCell's value to *anInt*

- (int)**intValue** Returns SliderCell’s value as an **int**
- **setStringValue:**(const char *)*aString* Sets the SliderCell’s value to a number represented by *aString*
- (const char *)**stringValue** Returns the the SliderCell’s value as a string

Modifying Graphic Attributes

- **setKnobThickness:**(NXCoord)*aFloat* Sets the knob’s thickness to *aFloat*
- (NXCoord)**knobThickness** Returns the knob’s thickness
- **setImage:***image* Sets the background image to *image*
- **image** Returns the background image
- **setTitle:**(const char *)*aString* Sets the background title to a copy of *aString*
- **setTitleNoCopy:**(const char *)*aString* Sets the background title to *aString*
- (const char *)**title** Returns the background title
- **setTitleCell:***aCell* Sets the Cell used to draw the background title
- **titleCell** Returns the Cell used to draw the background title
- **setTitleFont:***fontObject* Sets the Font used to draw the background title
- **titleFont** Returns the Font used to draw the background title
- **setTitleColor:**(NXColor)*aColor* Sets the color of text in the background title to *aColor*
- (NXColor)**titleColor** Returns the color of text in the background title
- **setTitleGray:**(float)*aFloat* Sets the gray of text in the background title to *aFloat*
- (float)**titleGray** Returns the gray of text in the background title
- (BOOL)**isOpaque** Returns YES (SliderCells are always opaque)
- (int)**isVertical** Returns 1 if vertical, 0 if horizontal, –1 if unknown

Displaying the SliderCell

- **drawSelf:**(const NXRect *)*cellFrame*
inView:*controlView* Draws the SliderCell’s bar and knob in *controlView*
- **drawInside:**(const NXRect *)*cellFrame*
inView:*controlView* Draws the inside of the SliderCell in *controlView*
- **drawBarInside:**(const NXRect *)*aRect*
flipped:(BOOL)*flipped* Draws the SliderCell’s bar
- **drawKnob** Draws the SliderCell’s knob
- **drawKnob:**(const NXRect*)*knobRect* Draws the SliderCell’s knob in *knobRect*

Modifying Behavior

- **setEnabled:**(BOOL)*flag* Sets whether the SliderCell reacts to events
- **setContinuous:**(BOOL)*flag* Sets whether the Slider is continuous
- (BOOL)**isContinuous** Returns whether the Slider is continuous

- **setAltIncrementValue:**(double)*incValue* Sets how far the SliderCell moves when the knob is dragged one pixel with the Alternate key held down
- (double)**altIncrementValue** Returns how far the SliderCell moves when alt-dragged

Tracking the Mouse

- + (BOOL)**prefersTrackingUntilMouseUp** Returns YES, since SliderCells must track even when the mouse leaves their bounds
- (BOOL)**trackMouse:**(NXEvent *)*theEvent* Tracks the mouse
 inRect:(const NXRect *)*cellFrame*
 ofView:*controlView*
- (BOOL)**startTrackingAt:**(const NXPoint *)*startPoint* Begins a tracking session
 inView:*controlView*
- (BOOL)**continueTracking:**(const NXPoint *)*lastPoint* Continues tracking the mouse
 at:(const NXPoint *)*currentPoint*
 inView:*controlView*
- **stopTracking:**(const NXPoint *)*lastPoint* Ends the current tracking session
 at:(const NXPoint *)*stopPoint*
 inView:*controlView*
 mouseIsUp:(BOOL)*flag*

Archiving

- **read:**(NXTypedStream *)*stream* Reads the SliderCell from *stream*
- **write:**(NXTypedStream *)*stream* Writes the SliderCell to *stream*
- **awake** Caches knob icons when the SliderCell is unarchived

Speaker

Inherits From: Object

Initializing a New Speaker Instance

- **init** Initializes the Speaker after it has been allocated

Freeing a Speaker

- **free** Deallocates the Speaker (but not its ports)

Setting Up a Speaker

- **setSendTimeout:(int)ms** Sets how long to wait for messages to be delivered
- **(int)sendTimeout** Returns how long to wait for messages to be delivered
- **setReplyTimeout:(int)ms** Sets how long Speaker will wait for a reply
- **(int)replyTimeout** Returns how long Speaker will wait for a reply

Managing the Ports

- **setSendPort:(port_t)aPort** Makes *aPort* the port messages will be sent to
- **(port_t)sendPort** Returns the port the Speaker will send messages to
- **setReplyPort:(port_t)aPort** Makes *aPort* the port where replies are received
- **(port_t)replyPort** Returns the port where Speaker receives replies

Standard Remote Methods

- **(int)openFile:(const char *)fullPath
ok:(int *)flag** Sends a remote message to open *fullPath* file
- **(int)openTempFile:(const char *)fullPath
ok:(int *)flag** Sends a remote message to open *fullPath* file

Providing for Program Control

- **(int)msgCalc:(int *)flag** Sends message to update the current window
- **(int)msgCopyAsType:(const char *)aType
ok:(int *)flag** Sends message to copy selection as *aType* data
- **(int)msgCutAsType:(const char *)aType
ok:(int *)flag** Sends message to cut selection as *aType* data
- **(int)msgDirectory:(char *const *)fullPath
ok:(int *)flag** Sends message requesting the current directory
- **(int)msgFile:(char *const *)fullPath
ok:(int *)flag** Sends message requesting the current document
- **(int)msgPaste:(int *)flag** Sends message to paste data from pasteboard
- **(int)msgPosition:(char *const *)aString
posType:(int *)anInt
ok:(int *)flag** Sends message requesting selection information
- **(int)msgPrint:(const char *)fullPath
ok:(int *)flag** Sends message to print *fullPath* file
- **(int)msgQuit:(int *)flag** Sends remote message for application to quit

- (int)**msgSelection**:(char *const *)*bytes*
 - length**:(int *)*numBytes*
 - asType**:(const char *)*aType*
 - ok**:(int *)*flag*

Sends message requesting the current selection
- (int)**msgSetPosition**:(const char *)*aString*
 - posType**:(int)*anInt*
 - andSelect**:(int)*sflag*
 - ok**:(int *)*flag*

Sends message to scroll so *aString* is visible
- (int)**msgVersion**:(char *const *)*aString*
 - ok**:(int *)*flag*

Sends message requesting version information

Sending Remote Messages

- (int)**performRemoteMethod**:(const char *)*methodName*

Sends remote *methodName* message
- (int)**performRemoteMethod**:(const char *)*methodName*
 - with**:(const char *)*data*
 - length**:(int)*numBytes*

Sends remote message with *numBytes* of *data*
- (int)**selectorRPC**:(const char *)*methodName*
 - paramTypes**:(char *)*params*,

Sends remote message with variable arguments
- (int)**sendOpenFileMsg**:(const char *)*fullPath*
 - ok**:(int *)*flag*
 - andDeactivateSelf**:(BOOL)*deactivateFirst*

Sends an **openFile:ok**: remote message
- (int)**sendOpenTempFileMsg**:(const char *)*fullPath*
 - ok**:(int *)*flag*
 - andDeactivateSelf**:(BOOL)*deactivateFirst*

Sends an **openTempFile:ok**: remote message

Assigning a Delegate

- **setDelegate**:*anObject*

Makes *anObject* the Speaker’s delegate
- **delegate**

Returns the Speaker’s delegate

Archiving

- **read**:(NXTypedStream *)*stream*

Reads the Speaker from *stream*
- **write**:(NXTypedStream *)*stream*

Writes the Speaker to *stream*

Text

Inherits From: View : Responder : Object

Conforms To: NXChangeSpelling
NXIgnoreMisspelledWords
NXReadOnlyTextStream
NXSelectText

Initializing the Class Object

+ setDefaultFont: <i>anObject</i>	Makes <i>anObject</i> the default Font object for Text
+ getDefaultFont	Returns the default Font object for Text
+ excludeFromServicesMenu: (BOOL) <i>flag</i>	Controls whether Text objects register for services
+ registerDirective: (const char *) <i>directive</i> forClass: <i>class</i>	Associates an RTF control word with a class object
+ initialize	Performed automatically at startup

Initializing a New Text Object

- initWithFrame: (const NXRect *) <i>frameRect</i>	Initialize a new Text object
- initWithFrame: (const NXRect *) <i>frameRect</i> text: (const char *) <i>theText</i> alignment: (int) <i>mode</i>	Initialize a new Text object

Freeing a Text Object

- free	Frees the Text object and its storage
---------------	---------------------------------------

Modifying the Frame Rectangle

- setMaxSize: (const NXSize *) <i>newMaxSize</i>	Sets maximum size of the Text object
- getMaxSize: (NXSize *) <i>theSize</i>	Gets maximum size of the Text object
- setMinSize: (const NXSize *) <i>newMinSize</i>	Sets minimum size of the Text object
- getMinSize: (NXSize *) <i>theSize</i>	Gets minimum size of the Text object
- setVertResizable: (BOOL) <i>flag</i>	Sets whether frame height can change
- (BOOL) isVertResizable	Returns whether frame height can change
- setHorizResizable: (BOOL) <i>flag</i>	Sets whether frame width can change
- (BOOL) isHorizResizable	Returns whether frame width can change
- sizeTo: (NXCoord) <i>width</i> :(NXCoord) <i>height</i>	Resizes the Text object to <i>width</i> and <i>height</i>

- **sizeToFit**
- **resizeText:**(const NXRect *)*oldBounds*
:(const NXRect *)*maxRect*
- **moveTo:**(NXCoord)*x* :(NXCoord)*y*

Resizes the frame to accommodate the text
Used by Text object to resize and redisplay itself

Moves the Text object to (*x*, *y*)

Laying Out the Text

- **setMarginLeft:**(NXCoord)*leftMargin*
right:(NXCoord)*rightMargin*
top:(NXCoord)*topMargin*
bottom:(NXCoord)*bottomMargin*
- **getMarginLeft:**(NXCoord *)*leftMargin*
right:(NXCoord *)*rightMargin*
top:(NXCoord *)*topMargin*
bottom:(NXCoord *)*bottomMargin*
- **getMinWidth:**(NXCoord *)*width*
minHeight:(NXCoord *)*height*
maxWidth:(NXCoord)*widthMax*
maxHeight:(NXCoord)*heightMax*
- **setAlignment:**(int)*mode*
- (int)**alignment**
- **alignSelLeft:***sender*
- **alignSelCenter:***sender*
- **alignSelRight:***sender*
- **setSelProp:**(NXParagraphProp)*prop*
to:(NXCoord)*val*
- **changeTabStopAt:**(NXCoord)*oldX*
to:(NXCoord)*newX*
- (int)**calcLine**
- **setCharWrap:**(BOOL)*flag*
- (BOOL)**charWrap**
- **setNoWrap**
- **setParaStyle:**(void *)*paraStyle*
- (void *)**defaultParaStyle**
- (void *)**calcParagraphStyle:***fontId*
:(int)*alignment*
- **setLineHeight:**(NXCoord)*value*
- (NXCoord)**lineHeight**
- **setDescentLine:**(NXCoord)*value*
- (NXCoord)**descentLine**

Adjusts margins around the text

Gets dimensions of margins around the text

Calculates area needed to display the text

Sets how text is aligned at margins

Returns how text is aligned at margins

Aligns the text to the left margin

Aligns the text between the margins

Aligns the text to the right margin

Sets the paragraph style for one or more paragraphs

Resets the position of the specified tab stop

Calculates line breaks

Returns whether extra long words are wrapped

Sets whether extra long words are wrapped

Disables word wrap

Sets paragraph style for the entire text

Returns the default paragraph style

Recalculates paragraph style

Sets height of a line of text

Returns height of a line of text

Sets distance from base line to bottom of line

Returns distance from base line to bottom of line

Reporting Line and Position

- (int)**lineFromPosition**:(int)*position* Converts character position to line number
- (int)**positionFromLine**:(int)*line* Converts line number to character position
- (int)**offsetFromPosition**:(int)*position* Returns the byte offset corresponding to *position*
- (int)**positionFromOffset**:(int)*offset* Returns the position corresponding to the byte offset

Setting, Reading, and Writing the Text

- **setText**:(const char *)*aString* Replaces current text with *aString*
- **readText**:(NXStream *)*stream* Replaces current text with text from *stream*
- **startReadingRichText** Sent before Text object begins reading RTF data
- **readRichText**:(NXStream *)*stream* Replaces text with RTF data from *stream*
- **readRichText**:(NXStream *)*stream*
atPosition:(int)*position* Lets you add RTF data to *stream*
- **finishReadingRichText** Sent after Text object reads RTF data
- (NXRTFDError)**openRTFDFrom**:(const char *)*path*
Opens the RTFD file package specified by *path*
- (NXRTFDError)**saveRTFDTo**:(const char *)*path*
removeBackup:(BOOL)*removeBackup*
errorHandler:*errorHandler* Saves the contents (text and images) of the Text object to the file package specified by *path*
- **writeText**:(NXStream *)*stream* Writes all the text to *stream*
- **writeRichText**:(NXStream *)*stream* Writes all the text to *stream* using RTF
- **writeRichText**:(NXStream *)*stream*
from:(int)*start*
to:(int)*end* Writes text to *stream* using RTF
- **writeRTFDSelectionTo**:(NXStream *)*stream* Writes the selection—text and images—to *stream*
- **writeRTFDTo**:(NXStream *)*stream* Writes all the text and images to *stream*
- (NXStream *)**stream** Returns stream access to Text object’s text
- (NXTextBlock *)**firstTextBlock** Returns pointer to first text block
- **getParagraph**:(int)*prNumber*
start:(int *)*startPos*
end:(int *)*endPos*
rect:(NXRect *)*paragraphRect* Gets position, length, and size of a paragraph
- (int)**getSubstring**:(char *)*buf*
start:(int)*startPos*
length:(int)*numChars* Copies *numChars* at *startPos* to *buf*
- (int)**byteLength** Returns length of the Text object’s contents in bytes
- (int)**charLength** Returns number of characters in the text
- (int)**textLength** Returns number of characters in the text

Setting Editability

- **setEditable:(BOOL)flag**
- **(BOOL)isEditable**

Sets whether the text can be edited
Returns whether the text can be edited

Allowing Multiple Fonts and Paragraph Styles

- **setMonoFont:(BOOL)flag**
- **(BOOL)isMonoFont**

Controls whether multiple fonts and parastyles are OK
Returns whether only one font and parastyle is permitted

Editing the Text

- **copy:sender**
- **copyFont:sender**
- **copyRuler:sender**
- **paste:sender**
- **pasteFont:sender**
- **pasteRuler:sender**
- **cut:sender**
- **delete:sender**
- **clear:sender**
- **selectAll:sender**
- **selectText:sender**

Copies selected text to the pasteboard
Copies selected text's font to the pasteboard
Copies selected text's style to the pasteboard
Replaces selection with pasteboard's contents
Replaces selection's font with pasteboard's contents
Replaces selection's style with pasteboard's contents
Deletes selected text; copies it to pasteboard
Deletes selected text
Deletes selected text
Makes receiver the first responder; selects all text
Makes receiver the first responder; selects all text

Managing the Selection

- **subscript:sender**
- **superscript:sender**
- **unscript:sender**
- **underline:sender**
- **showCaret**
- **hideCaret**
- **setSelectable:(BOOL)flag**
- **(BOOL)isSelectable**
- **selectError**
- **selectNull**
- **setSel:(int)start :(int)end**
- **getSel:(NXSelPt *)start :(NXSelPt *)end**
- **replaceSel:(const char *)aString**

Subscripts the current selection
Superscripts the current selection
Removes sub/super script in the current selection
Toggles the underline attribute of text
Displays the previously hidden caret
Removes the caret from the text display
Sets whether the text can be selected
Returns whether the text can be selected
Selects all the text
Deselects the current selection
Selects text from *start* through *end*
Gets *start* and *end* of the selection
Replaces the selection with *aString*

- **replaceSel:(const char *)aString length:(int)length** Replaces selection with *length* bytes of *aString*
- **replaceSel:(const char *)aString length:(int)length runs:(NXRunArray *)insertRuns** Replaces selection with *length* bytes of *aString*
- **replaceSelWithRichText:(NXStream *)stream** Replaces selection with RTF from *stream*
- **replaceSelWithRTFD:(NXStream *)stream** Replaces selection with RTFD data from *stream*
- **scrollSelToVisible** Brings the selection within the frame rectangle

Setting the Font

- **setFontPanelEnabled:(BOOL)flag** Sets whether the Font panel can affect text
- **(BOOL)isFontPanelEnabled** Sets whether the Font panel can affect text
- **changeFont:sender** Changes font of selection
- **setFont:fontObj** Sets Font object for the entire text
- **font** Returns a monofont Text object's font
- **setFont:fontObj paraStyle:(void *)paraStyle** Sets Font and paragraph style for all text
- **setSelFont:fontId** Sets Font object for the selection
- **setSelFontFamily:(const char *)fontName** Sets font family for the selection
- **setSelFontSize:(float)size** Sets font size for the selection
- **setSelFontStyle:(NXFontTraitMask)traits** Sets font style for the selection
- **setSelFont:fontId paraStyle:(void *)paraStyle** Sets font and paragraph style for the selection

Checking Spelling

- **checkSpelling:sender** Searches for a misspelled word in the text
- **showGuessPanel:sender** Displays panel suggesting spelling corrections

Managing the Ruler

- **toggleRuler:sender** Controls the display of the ruler
- **(NXColor)isRulerVisible** Returns whether the ruler is visible in the superview

Finding Text

- **(BOOL)findText:(const char *)string ignoreCase:(BOOL)ignoreCaseFlag backwards:(BOOL)backwardsFlag wrap:(BOOL)wrapFlag** Searches for *string* in the text, starting at the insertion point

Modifying Graphic Attributes

- **setBackgroundGray:(float)value** Sets the gray value of the text background
- (float)**backgroundGray** Returns the gray value of the text background
- **setBackground-color:(NXColor)color** Sets background color of the text
- (NXColor)**background-color** Returns the background color of the text
- **setSelGray:(float)value** Sets the gray value of the selected text
- (float)**selGray** Returns the gray value of the selected text
- (float)**runGray:(NXRun *)run** Returns the gray value for the specified text run
- **setSelColor:(NXColor)color** Sets the color of the selected text
- (NXColor)**selColor** Returns the color of the selected text
- (NXColor)**runColor:(NXRun *)run** Returns the color of the specified text run
- **setTextGray:(float)value** Sets the gray value of the entire text
- (float)**textGray** Returns the gray value of the entire text
- **setTextColor:(NXColor)color** Sets the text color of the entire text
- (NXColor)**textColor** Returns the text color of the draw entire text

Reusing a Text Object

- **renewFont:newFontId**
text:(const char *)newText
frame:(const NXRect *)newFrame
tag:(int)newTag Resets Text object to draw different text
- **renewFont:(const char *)newFontName**
size:(float)newFontSize
style:(int)newFontStyle
text:(const char *)newText
frame:(const NXRect *)newFrame
tag:(int)newTag Resets Text object to draw different text
- **renewRuns:(NXRunArray *)newRuns**
text:(const char *)newText
frame:(const NXRect *)newFrame
tag:(int)newTag Resets Text object to draw different text
- **windowChanged:newWindow** Hides caret whenever the Text's window changes

Displaying

- **drawSelf:(const NXRect *)rects :(int)rectCount** Draws the Text object
- **setRetainedWhileDrawing:(BOOL)flag** Allows use of retained window when drawing
- (BOOL)**isRetainedWhileDrawing** Returns whether retained window is used for drawing

Assigning a Tag

- **setTag:(int)*anInt*** Makes *anInt* the Text object's tag
- **(int)tag** Returns the Text object's tag

Handling Event Messages

- **(BOOL)acceptsFirstResponder** Returns whether receiver can be the first responder
- **becomeFirstResponder** Informs Text object that it's becoming first responder
- **resignFirstResponder** Stops being the first responder, if delegate agrees
- **becomeKeyWindow** Activates caret if selection has width of 0
- **resignKeyWindow** Deactivates the caret
- **mouseDown:(NXEvent *)*theEvent*** Responds to mouse-down events
- **keyDown:(NXEvent *)*theEvent*** Responds to key-down events
- **moveCaret:(unsigned short)*theKey*** Moves the caret in response to arrow keys

Displaying Graphics within the Text

- + **registerDirective:(const char *)*directive*
forClass:*class*** Associates an RTF control word with a class object
- **replaceSelWithCell:*cell*** Replaces selection with image provided by *cell*
- **replaceSelWithView:*view*** Unimplemented
- **setLocation:(NXPoint *)*origin*
ofCell:*cell*** Sets origin of *cell*
- **getLocation:(NXPoint *)*origin*
ofCell:*cell*** Places coordinates of graphic object into *origin*
- **getLocation:(NXPoint *)*origin*
ofView:*view*** Unimplemented
- **setGraphicsImportEnabled:(BOOL)*flag*** Sets whether a Text object imports TIFF and EPS images
- **(BOOL)isGraphicsImportEnabled** Returns YES if the object imports TIFF and EPS images

Using the Services Menu

- + **excludeFromServicesMenu:(BOOL)*flag*** Controls whether Text objects use services menu
- **validRequestorForSendType:(NXAtom)*sendType*
andReturnType:(NXAtom)*returnType*** Determines which Service menu items are enabled
- **readSelectionFromPasteboard:*pboard*** Replaces selection with data from *pboard*
- **(BOOL)writeSelectionToPasteboard:*pboard*
types:(NXAtom *)*types*** Copies selection to *pboard*

Setting Tables and Functions

- **setCharFilter:**(NXCharFilterFunc)*aFunc* Makes *aFunc* the character filter function
- (NXCharFilterFunc)**charFilter** Returns the current character filter function
- **setTextFilter:**(NXTextFilterFunc)*aFunc* Makes *aFunc* the text filter function
- (NXTextFilterFunc)**textFilter** Returns the current text filter function
- **setBreakTable:**(const NXFSM *)*aTable* Sets table defining word boundaries
- (const NXFSM *)**breakTable** Gets table defining word boundaries
- **setPreSelSmartTable:**(const unsigned char *)*aTable* Sets cut and paste table for left word boundary
- (const unsigned char *)**preSelSmartTable** Gets cut and paste table for left word boundary
- **setPostSelSmartTable:**(const unsigned char *)*aTable* Sets cut and paste table for right word boundary
- (const unsigned char *)**postSelSmartTable** Gets cut and paste table for right word boundary
- **setCharCategoryTable:**(const unsigned char *)*aTable* Sets table defining character categories
- (const unsigned char *)**charCategoryTable** Returns table defining character categories
- **setClickTable:**(const NXFSM *)*aTable* Sets table defining double-click selection
- (const NXFSM *)**clickTable** Gets table defining double-click selection
- **setScanFunc:**(NXTextFunc)*aFunc* Makes *aFunc* the scan function
- (NXTextFunc)**scanFunc** Returns the current scan function
- **setDrawFunc:**(NXTextFunc)*aFunc* Makes *aFunc* the function that draws the text
- (NXTextFunc)**drawFunc** Returns the current draw function

Printing

- **adjustPageHeightNew:**(float *)*newBottom* Assists automatic pagination of text
top:(float)*oldTop*
bottom:(float)*oldBottom*
limit:(float)*bottomLimit*

Archiving

- **read:**(NXTypedStream *)*stream* Reads the Text object from the typed stream
- **write:**(NXTypedStream *)*stream* Writes the Text object to the typed stream

Assigning a Delegate

- **setDelegate:***anObject* Makes *anObject* the Text object's delegate
- **delegate** Returns the Text object's delegate

Implemented by the Delegate

- **textWillResize:***sender* Informs delegate of impending size change
- **textDidResize:***sender*
 oldBounds:(const NXRect *)*oldBounds*
 invalid:(NXRect *)*invalidRect* Reports size change to delegate
- (BOOL)**textWillChange:***sender* Informs delegate of impending text change
- **textDidChange:***sender* Alerts delegate to change in text
- (BOOL)**textWillEnd:***sender* Warns of impending loss of first responder status
- **textDidEnd:***sender*
 endChar:(unsigned short)*whyEnd* Reports to delegate loss of first responder status
- **textDidGetKeys:***sender isEmpty:(BOOL)**flag* Informs delegate of each text change
- **textWillSetSel:***sender toFont:font* Lets delegate intercede in the updating of the Font panel
- **textWillConvert:***sender*
 fromFont:*from*
 toFont:*to* Lets delegate intercede in selection's font change
- **textWillStartReadingRichText:***sender* Informs delegate that Text object will read RTF data
- **textWillFinishReadingRichText:***sender* Informs delegate that Text finished reading RTF data
- **textWillWrite:***sender*
 paperSize:(NXSize *)*paperSize* Lets the delegate specify paper size
- **textDidRead:***sender*
 paperSize:(NXSize *)*paperSize* Lets the delegate review paper size

Implemented by an Embedded Graphic Object

- **calcCellSize:**(NXSize *)*theSize* Provides the size of the object
- **drawSelf:**(const NXRect *)*rect*
 inView:*view* Draws the object
- **highlight:**(const NXRect *)*rect*
 inView:*view*
 lit:(BOOL)*flag* Highlights or unhighlights the object
- **readRichText:**(NXStream *)*stream*
 forView:*view* Reads representation from RTF data
- **writeRichText:**(NXStream *)*stream*
 forView:*view* Writes RTF representation to stream
- (BOOL)**trackMouse:**(NXEvent *)*theEvent*
 inRect:(const NXRect *)*rect*
 ofView:*view* Controls tracking of the mouse

TextField

Inherits From: Control : View : Responder : Object

Initializing the TextField Class

+ **setCellClass:***classId* Sets the Cell class used by TextField

Initializing a new TextField

– **initWithFrame:**(const NXRect *)*frameRect* Initializes a new TextField object with no text

Enabling the TextField

– **setEnabled:**(BOOL)*flag* Sets whether the TextField reacts to events

Setting User Access to Text

– **setSelectable:**(BOOL)*flag* Sets whether the TextField's text is selectable
– (BOOL)**isSelectable** Returns whether the TextField's text is selectable
– **setEditable:**(BOOL)*flag* Sets whether the TextField's text is editable
– (BOOL)**isEditable** Returns whether the TextField's text is editable

Editing Text

– **selectText:***sender* Selects all of the text if it's selectable or editable

Setting Tab Key Behavior

– **setNextText:***anObject* Sets the object selected when the user presses Tab
– **nextText** Sets the object selected when the user presses Tab
– **setPreviousText:***anObject* Sets the object selected when the user types Shift-Tab
– **previousText** Sets the object selected when the user types Shift-Tab

Assigning a Text Delegate

– **setTextDelegate:***anObject* Sets the delegate for messages from the field editor
– **textDelegate** Returns the delegate for messages from field editor

Text Object Delegate Methods

- (BOOL)**textWillChange:***textObject* Responds to a message from the field editor
- **textDidGetKeys:***textObject*
 isEmpty:(BOOL)*flag* Responds to a message from the field editor
- **textDidChange:***textObject* Responds to a message from the field editor
- (BOOL)**textWillEnd:***textObject* Responds to a message from the field editor
- **textDidEnd:***textObject*
 endChar:(unsigned short)*whyEnd* Responds to a message from the field editor

Modifying Graphic Attributes

- **setTextColor:**(NXColor)*aColor* Sets the color of the TextField’s text to *aColor*
- (NXColor)**textColor** Returns the color of the TextField’s text
- **setTextGray:**(float)*value* Sets the gray of the TextField’s text to *value*
- (float)**textGray** Returns the gray of the TextField’s text
- **setBackground-color:**(NXColor)*aColor* Sets the color of the background to *aColor*
- (NXColor)**background-color** Returns the color of the background
- **setBackgroundGray:**(float)*value* Sets the gray of the background to *value*
- (float)**backgroundGray** Returns the gray of the background
- **setBackgroundTransparent:**(BOOL)*flag* Sets whether the TextField background is transparent
- (BOOL)**isBackgroundTransparent** Returns whether the TextField background is transparent
- **setBezeled:**(BOOL)*flag* Sets whether the TextField has a bezeled border
- (BOOL)**isBezeled** Returns whether the TextField has a bezeled border
- **setBordered:**(BOOL)*flag* Sets whether the TextField has a plain border
- (BOOL)**isBordered** Returns whether the TextField has a plain border

Target and Action

- **setErrorAction:**(SEL)*aSelector* Sets the action method sent for an invalid value entered
- (SEL)**errorAction** Returns the action method sent for an invalid value

Resizing a TextField

- **sizeTo:**(float)*width* :(float)*height* Resizes the TextField to *width* and *height*

Handling Events

- (BOOL)**acceptsFirstResponder** Returns YES if text is editable or selectable
- **mouseDown:**(NXEvent *)*theEvent* Responds to a mouse-down event

Archiving

- **read:**(NXTypedStream *)*stream* Reads the TextField from *stream*
- **write:**(NXTypedStream *)*stream* Writes the TextField to *stream*

TextFieldCell

Inherits From: ActionCell : Cell : Object

Initializing a new TextFieldCell

- **init** Initializes a new TextFieldCell with text “Field”
- **initWithCell:**(const char *)*aString* Initializes a new TextFieldCell with text *aString*

Copying a TextFieldCell

- **copyFromZone:**(NXZone *)*zone* Returns a copy of the TextFieldCell allocated from *zone*

Modifying Graphic Attributes

- **setTextColor:**(NXColor)*aColor* Sets the color of the text to *aColor*
- (NXColor)**textColor** Returns the color of the text
- **setTextGray:**(float)*value* Sets the gray of the text to *value*
- (float)**textGray** Returns the gray of the text
- **setBackground-color:**(NXColor)*aColor* Sets the color of the background to *aColor*
- (NXColor)**background-color** Returns the color of the background
- **setBackgroundGray:**(float)*value* Sets the gray of the background to *value*
- (float)**backgroundGray** Returns the gray of the background
- **setBackgroundTransparent:**(BOOL)*flag* Sets whether the background is transparent
- (BOOL)**isBackgroundTransparent** Returns whether the background is transparent
- **setTextAttributes:***textObject* Sets the gray values of the background and text to those of *textObject*
- **setBezeled:**(BOOL)*flag* Sets whether TextFieldCell has a bezeled border
- (BOOL)**isOpaque** Returns whether the cell is opaque

- **replaceSubview:***oldView* **with:***newView*
- **subviews**
- **superview**
- **window**
- **windowChanged:***newWindow*

Replaces *oldView* with *newView*
 Returns a List of the View’s subviews
 Returns the receiving View’s superview
 Returns the Window in which the View is displayed
 Notifies the View that the Window it’s in is changing

Modifying the Frame Rectangle

- (float)**frameAngle**
- **getFrame:**(NXRect *)*theRect*
- **moveBy:**(NXCoord)*deltaX* :(NXCoord)*deltaY*
- **moveTo:**(NXCoord)*x* :(NXCoord)*y*
- **rotateBy:**(NXCoord)*deltaAngle*
- **rotateTo:**(NXCoord)*angle*
- **setFrame:**(const NXRect *)*frameRect*
- **sizeBy:**(NXCoord)*deltaWidth* :(NXCoord)*deltaHeight*
- **sizeTo:**(NXCoord)*width* :(NXCoord)*height*

Returns the angle of frame rectangle rotation
 Gets the View’s frame rectangle
 Moves the View by *deltaX* and *deltaY*
 Moves the View to (*x*, *y*)
 Rotates the View’s frame rectangle by *deltaAngle*
 Rotates the View’s frame rectangle to *angle*
 Assigns the View a new frame rectangle
 Resizes the View by *deltaWidth* and *deltaHeight*
 Resizes the View to *width* and *height*

Modifying the Coordinate System

- (float)**boundsAngle**
- **drawInSuperview**
- **getBounds:**(NXRect *)*theRect*
- (BOOL)**isFlipped**
- (BOOL)**isRotatedFromBase**
- (BOOL)**isRotatedOrScaledFromBase**
- **rotate:**(NXCoord)*angle*
- **setDrawRotation:**(NXCoord)*angle*
- **scale:**(NXCoord)*x* :(NXCoord)*y*
- **setDrawSize:**(NXCoord)*width* :(NXCoord)*height*
- **translate:**(NXCoord)*x* :(NXCoord)*y*
- **setDrawOrigin:**(NXCoord)*x* :(NXCoord)*y*
- **setFlipped:**(BOOL)*flag*

Returns the rotation of the View’s coordinate system
 Makes the View use its superview’s coordinate system
 Gets the View’s bounds rectangle
 Returns whether the View is flipped
 Returns whether the View is rotated
 Returns whether the View is rotated or scaled
 Rotates the View’s coordinate system *by angle*
 Rotates the View’s coordinate system to *angle*
 Scales the View’s coordinate system
 Resizes the View’s coordinate system to *width* and *height*
 Shifts the View’s coordinate system to (*x*, *y*)
 Sets the View’s origin to (*x*, *y*)
 Sets whether polarity of y-axis is reversed

Converting Coordinates

- **centerScanRect:**(NXRect *)*aRect*

Converts the rectangle to lie on center of pixels

- **convertPoint:(NXPoint *)aPoint
fromView:aView** Converts the point to the receiver’s coordinates
- **convertPoint:(NXPoint *)aPoint toView:aView** Converts the point to aView’s coordinates
- **convertPointFromSuperview:(NXPoint *)aPoint** Converts the point to the receiver’s coordinates
- **convertPointToSuperview:(NXPoint *)aPoint** Converts the point to the superview’s coordinates
- **convertRect:(NXRect *)aRect fromView:aView** Converts the rectangle to the receiver’s coordinates
- **convertRect:(NXRect *)aRect toView:aView** Converts the rectangle to aView’s coordinates
- **convertRectFromSuperview:(NXRect *)aRect** Converts the rectangle to the receiver’s coordinates
- **convertRectToSuperview:(NXRect *)aRect** Converts the rectangle to the superview’s coordinates
- **convertSize:(NXSize *)aSize fromView:aView** Converts the size to the receiver’s coordinates
- **convertSize:(NXSize *)aSize toView:aView** Converts the size to aView’s coordinates

Notifying Ancestor Views

- **descendantFlipped:sender** Notifies that *sender*’s y-axis has flipped
- **descendantFrameChanged:sender** Notifies that *sender*’s frame rectangle changed
- **notifyAncestorWhenFrameChanged:(BOOL)flag** Sets whether to notify ancestors of frame change
- **notifyWhenFlipped:(BOOL)flag** Sets whether to notify ancestors of flipped y-axis
- **suspendNotifyAncestorWhenFrameChanged:(BOOL)flag** Starts/stops temporary suspension of ancestor notification

Resizing Subviews

- **resizeSubviews:(const NXSize *)oldSize** Initiates **SuperviewSizeChanged:** messages
- **setAutoresizeSubviews:(BOOL)flag** Sets whether to notify subviews of resizing
- **setAutosizing:(unsigned int)mask** Determines automatic resizing behavior
- **(unsigned int)autosizing** Returns the View’s autosizing mask
- **SuperviewSizeChanged:(const NXSize *)oldSize** Notifies subviews that superview changed size

Graphics State Objects

- **allocateGState** Allocates a graphics state object (when next focused upon)
- **freeGState** Frees the View’s graphics state object
- **(int)gState** Returns the View’s graphics state object
- **initGState** Initializes the View’s graphics state object
- **renewGState** Reinitializes the View’s graphics state object
- **notifyToInitGState:(BOOL)flag** Determines whether **initGState** message is sent

Focusing

- **clipToFrame:**(const NXRect *)*frameRect* Clips to the frame rectangle during focusing
- **setClipping:**(BOOL)*flag* Sets whether the View is clipped to its frame rectangle
- (BOOL)**doesClip** Returns whether the View clips to its frame rectangle
- (BOOL)**isFocusView** Returns whether the View is currently in focus
- (BOOL)**lockFocus** Brings the View into focus
- **unlockFocus** Unfocuses the View

Displaying

- (BOOL)**canDraw** Returns whether the View can draw
- **display** Displays the View and its subviews
- **display:**(const NXRect *)*rects* :(int)*rectCount* Displays the View and its subviews
- **display:**(const NXRect *)*rects* :(int)*rectCount* :(BOOL)*clipFlag* Displays the View and its subviews
- **displayFromOpaqueAncestor:**(const NXRect *)*rects* :(int)*rectCount* :(BOOL)*clipFlag* Displays underlying ancestors and the View
- **displayIfNeeded** Conditionally displays the View and its subviews
- **drawSelf:**(const NXRect *)*rects* :(int)*rectCount* Implemented by subclasses to supply drawing instructions
- (BOOL)**getVisibleRect:**(NXRect *)*theRect* Gets the View's visible portion
- (BOOL)**isAutodisplay** Returns whether the View automatically updates
- **setAutodisplay:**(BOOL)*flag* Determines whether **update** *redisplay*s the View
- (BOOL)**isOpaque** Returns whether the View is registered as opaque
- **setOpaque:**(BOOL)*flag* Registers the View as opaque
- (BOOL)**needsDisplay** Returns whether the View needs to be redisplayed
- **setNeedsDisplay:**(BOOL)*flag* Marks the View as changed, needing redisplay
- (BOOL)**shouldDrawColor** Returns whether the View should be drawn in color
- **update** Conditionally redisplayes the View

Scrolling

- **adjustScroll:**(NXRect *)*newVisible* Lets the View adjust the visible rectangle
- **autoscroll:**(NXEvent *)*theEvent* Scrolls in response to a mouse-dragged event
- (BOOL)**calcUpdateRects:**(NXRect *)*rects* :(int *)*rectCount* :(NXRect *)*enclRect* :(NXRect *)*goodRect* Calculates the area to be redisplayed

- **invalidate:**(const NXRect *)*rects* :(int)*rectCount*
Marks parts of the View as needing to be redrawn
- **scrollPoint:**(const NXPoint *)*aPoint*
Aligns *aPoint* with the content view’s origin
- **scrollRect:**(const NXRect *)*aRect*
by:(const NXPoint *)*delta*
Shifts the rectangle by *delta*
- **scrollRectToVisible:**(const NXRect *)*aRect*
Scrolls the View so the rectangle is visible
- (float)**backgroundGray**
Returns the View’s background gray (used by a
ScrollView’s document view only)

Managing the Cursor

- **addCursorRect:**(const NXRect *)*aRect*
cursor:*anObj*
Adds a cursor rectangle to the View
- **discardCursorRects**
Removes all cursor rectangles in the View
- **removeCursorRect:**(const NXRect *)*aRect*
cursor:*anObj*
Removes a cursor rectangle from the View
- **resetCursorRects**
Resets the View’s cursor rectangles

Assigning a Tag

- **findViewWithTag:**(int)*aTag*
Returns the subview with *aTag* as its tag
- (int)**tag**
Returns the View’s tag

Aiding Event Handling

- (BOOL)**acceptsFirstMouse**
Returns NO to refuse first mouse-down event
- **hitTest:**(NXPoint *)*aPoint*
Returns the lowest subview containing the point
- (BOOL)**mouse:**(NXPoint *)*aPoint*
inRect:(NXRect *)*aRect*
Returns whether the point lies inside the rectangle
- (BOOL)**performKeyEquivalent:**(NXEvent *)*theEvent*
Returns whether a subview handled *theEvent*
- (BOOL)**shouldDelayWindowOrderingForEvent:**(NXEvent *)*anEvent*
Returns whether the View’s Window is brought forward
normally (mouse-down) or delayed (mouse-up)

Dragging

- **registerForDraggedTypes:**(const char *const *)*pbTypes* **count:**(int)*count*
Registers the Pasteboard types that the Window will accept
in an image-dragging session
- **unregisterDraggedTypes**
Unregisters the Window as a recipient of dragged images

- **dragImage:***anImage*
 - at:**(NXPoint *)*location*
 - offset:**(NXPoint *)*initialOffset*
 - event:**(NXEvent *)*event*
 - pasteboard:**(Pasteboard *)*pboard*
 - source:***sourceObject*
 - slideBack:**(BOOL)*slideFlag*
- **dragFile:**(const char *)*filename*
 - fromRect:**(NXRect *)*rect*
 - slideBack:**(BOOL) *aFlag*
 - event:**(NXEvent *)*event*

Instigates an image-dragging session

Instigates a file-dragging session

Printing

- **printPSCode:***sender*
 - numberList:**(const char *const *)*numbers*
 - sendAt:**(time_t)*time*
 - wantsCover:**(BOOL)*coverFlag*
 - wantsNotify:**(BOOL)*notifyFlag*
 - wantsHires:**(BOOL)*hiresFlag*
 - faxName:**(const char *)*string*
- **copyPSCodeInside:**(const NXRect *)*rect*
 - to:**(NXStream *)*stream*
- **writePSCodeInside:**(const NXRect *)*rect*
 - to:***pasteboard*
- **openSpoolFile:**(char *)*filename*
- **spoolFile:**(const char *)*filename*
- (BOOL)**canPrintRIB**

Prints the View and its subviews

Faxes the View and its subviews

Faxes the View and its subviews

Generates PostScript code for the rectangle

Places PostScript code for the rectangle on the pasteboard

Opens *filename* for print spooling

Spools *filename* to the printer

Indicates whether the View can print RIB files

Setting Up Pages

- (BOOL)**knowsPagesFirst:**(int *)*firstPageNum*
 - last:**(int *)*lastPageNum*
- (BOOL)**getRect:**(NXRect *)*theRect*
 - forPage:**(int)*page*
- **placePrintRect:**(const NXRect *)*aRect*
 - offset:**(NXPoint *)*location*
- (float)**heightAdjustLimit**
- (float)**widthAdjustLimit**

Returns whether the View paginates itself

Provides how much of the View will print on *page*

Locates the printing rectangle on the page

Returns how much of a page can go on the next page

Returns how much of a page can go on the next page

Writing Conforming PostScript

- **beginPSOutput** Initializes the printing environment
- **beginPrologueBBox:(const NXRect *)boundingBox**
 creationDate:(const char *)dateCreated Writes the beginning of the prologue for a print job
 createdBy:(const char *)anApplication
 fonts:(const char *)fontNames
 forWhom:(const char *)user
 pages:(int)numPages
 title:(const char *)aTitle
- **endHeaderComments** Writes the end of the header
- **endPrologue** Writes the end of the prologue
- **beginSetup** Writes the beginning of the document setup section
- **endSetup** Writes the end of the document setup section
- **adjustPageWidthNew:(float *)newRight** Assists automatic pagination of the View
 left:(float)oldLeft
 right:(float)oldRight
 limit:(float)rightLimit
- **adjustPageHeightNew:(float *)newBottom** Assists automatic pagination of the View
 top:(float)oldTop
 bottom:(float)oldBottom
 limit:(float)bottomLimit
- **beginPage:(int)ordinalNum** Writes a page separator
 label:(const char *)aString
 bBox:(const NXRect *)pageRect
 fonts:(const char *)fontNames
- **beginPageSetupRect:(const NXRect *)aRect** Writes the beginning of a page setup section
 placement:(const NXPoint *)location
- **drawSheetBorder:(float)width :(float)height** Allows you to draw a sheet border
- **drawPageBorder:(float)width :(float)height** Allows you to draw a page border
- **addToPageSetup** Allows you to add scaling to PostScript code
- **endPageSetup** Writes the end of a page setup section
- **endPage** Writes the end of a page
- **beginTrailer** Writes the beginning of the trailer for the print job
- **endTrailer** Writes the end of the trailer
- **endPSOutput** Finishes the printing job

Archiving

- **awake** Initializes the View after reading
- **read:(NXTypedStream *)stream** Reads the View from the typed stream
- **write:(NXTypedStream *)stream** Writes the View to the typed stream

Window

Inherits From: Responder : Object

Initializing a New Window Object

- **init** Initializes the new Window with default parameters
- **initWithContent:(const NXRect *)contentRect style:(int)aStyle backing:(int)bufferingType buttonMask:(int)mask defer:(BOOL)flag** Initializes the new Window object as specified
- **initWithContent:(const NXRect *)contentRect style:(int)aStyle backing:(int)bufferingType buttonMask:(int)mask defer:(BOOL)flag screen:(const NXScreen *)aScreen** Initializes the new Window object for *screen* as specified

Freeing a Window Object

- **free** Frees the Window object and its Views

Computing Frame and Content Rectangles

- + **getFrameRect:(NXRect *)frame forContentRect:(const NXRect *)content style:(int)aStyle** Gets frame rectangle for given content rectangle
- + **getContentRect:(NXRect *)content forFrameRect:(const NXRect *)frame style:(int)aStyle** Gets content rectangle for given frame rectangle
- + **(NXCoord)minFrameWidth:(const char *)aTitle forStyle:(int)aStyle buttonMask:(int)aMask** Returns minimum frame width needed for *aTitle*

Accessing the Frame Rectangle

- **getFrame:(NXRect *)theRect** Gets the Window's frame rectangle
- **getFrame:(NXRect *)theRect andScreen:(const NXScreen **)theScreen** Gets the Window's frame rectangle and screen

- (BOOL)setFrameUsingName:(const char *)*name* Sets the frame rectangle from the named default
- (void)saveFrameUsingName:(const char *)*name* Saves the frame rectangle as a system default
- + (void)removeFrameUsingName:(const char *)*name* Removes the named frame data from the system defaults
- (BOOL)setFrameAutosaveName:(const char *)*name* Sets the name that’s used to autosave the frame rectangle as a system default
- (const char *)frameAutosaveName Returns the name that’s used to autosave the frame rectangle as a system default
- (void)setFrameFromString:(const char *)*string* Sets the frame rectangle from *string*
- saveFrameToString:(const char *)*string* Saves the frame rectangle data to *string*

Accessing the Content View

- setContentView:*aView* Makes *aView* the Window’s content view
- contentView Returns the Window’s content view

Querying Window Attributes

- (int>windowNum Returns the window number
- (int)buttonMask Returns the mask identifying Window controls
- (int)style Returns the Window’s border and title bar style
- (BOOL)worksWhenModal Returns NO

Window Graphics

- setTitle:(const char *)*aString* Makes *aString* the Window’s title
- setTitleAsFilename:(const char *)*aString* Formats *aString* and makes it the Window’s title
- (const char *)title Returns the Window’s title string
- setBackgroundColor:(NXColor)*color* Sets *the* Window’s background color to *color*
- (NXColor)backgroundColor Returns the Window’s background color
- setBackgroundGray:(float)*value* Sets the gray *value* for Window’s background gray
- (float)backgroundGray Returns the Window’s background gray

Window Device Attributes

- setBackingType:(int)*backing* Sets the type of window device backing
- (int)backingType Returns the window device backing type
- setOneShot:(BOOL)*flag* Sets whether memory for the window should be freed

- (BOOL)isOneShot
- setFreeWhenClosed:(BOOL)flag

Returns whether memory for the window is freed
Sets whether closing the Window also frees it

Graphics State Objects

- (int)gState

Returns the graphics state object for the Window

The Miniwindow

- counterpart
- setMiniwindowIcon:(const char *)name
- (const char *)miniwindowIcon
- setMiniwindowImage:image
- (NXImage *)miniwindowImage
- setMiniwindowTitle:(const char *)title
- (const char *)miniwindowTitle

Returns the receiver's miniwindow/Window companion
Sets the icon that's displayed in the miniwindow
Returns the icon that's displayed in the miniwindow
Sets the image that's displayed in the miniwindow
Returns the image that's displayed in the miniwindow
Sets the title that's displayed in the miniwindow
Sets the title that's displayed in the miniwindow

The Field Editor

- getFieldEditor:(BOOL)flag for:anObject
- endEditingFor:anObject

Returns the Window's field editor
Ends the field editor's editing assignment

Window Status

- makeKeyWindow
- makeKeyAndOrderFront:sender
- becomeKeyWindow
- (BOOL)isKeyWindow
- resignKeyWindow
- (BOOL)canBecomeKeyWindow
- becomeMainWindow
- (BOOL)isMainWindow
- resignMainWindow
- (BOOL)canBecomeMainWindow

Makes the receiver the key window
Makes Window the key window and brings it forward
Records Window's new status as the key window
Returns whether Window is the key window
Records that Window no longer is the key window
Returns whether Window can be the key window
Records Window's new status as the main window
Returns whether Window is the main window
Records that Window no longer is the main window
Returns whether Window can be the main window

Moving and Resizing the Window

- moveTo:(NXCoord)x :(NXCoord)y
- moveTo:(NXCoord)x
:(NXCoord)y
screen:(const NXScreen *)aScreen

Moves the Window to (x, y)
Moves the Window to (x, y) relative to aScreen

- **moveTopLeftTo:**(NXCoord)x :(NXCoord)y Moves top left corner of the Window to (x, y)
- **moveTopLeftTo:**(NXCoord)x
:(NXCoord)y
screen:(const NXScreen *)*aScreen* Moves top left corner of the Window relative to *aScreen*
- **dragFrom:**(float)x :(float)y **eventNum:**(int)*num* Lets user drag the Window from (x, y)
- **constrainFrameRect:**(NXRect *)*frameRect*
toScreen:(const NXScreen *)*screen* Constrains the Window to *screen*
- **placeWindow:**(const NXRect *)*frameRect* Resizes the Window to new frame rectangle
- **placeWindow:**(const NXRect *)*frameRect*
screen:(const NXScreen *)*aScreen* Resizes the Window relative to *aScreen*
- **placeWindowAndDisplay:**(const NXRect *)*frameRect*
Resizes the Window while redisplaying its Views
- **sizeWindow:**(NXCoord)*width* :(NXCoord)*height*
Resizes the Window's content area
- **center** Centers the Window on the screen
- **setMinSize:**(const NXSize *)*aSize* Sets the Window's minimum size
- **getMinSize:**(NXSize *)*aSize* Returns the Window's minimum size
- **setMaxSize:**(const NXSize *)*aSize* Sets the Window's maximum size
- **getMaxSize:**(NXSize *)*aSize* Returns the Window's maximum size
- (int)**resizeFlags** Returns the event flags during resizing

Reordering the Window

- **makeKeyAndOrderFront:***sender* Makes the Window the key window and brings it forward
- **orderFront:***sender* Puts the Window at the front of its tier
- **orderBack:***sender* Puts the Window at the back of its tier
- **orderOut:***sender* Removes the Window from the screen list
- **orderWindow:**(int)*place relativeTo*:(int)*otherWin*
Repositions the Window in screen list
- **orderFrontRegardless** Puts the Window at the front even if the application is inactive
- (BOOL)**isVisible** Returns whether the Window is in the screen list
- **setHideOnDeactivate:**(BOOL)*flag* Sets whether deactivation hides the Window
- (BOOL)**doesHideOnDeactivate** Returns whether deactivation hides the Window

Converting Coordinates

- **convertBaseToScreen:**(NXPoint *)*aPoint* Converts point from base to screen coordinates
- **convertScreenToBase:**(NXPoint *)*aPoint* Converts point from screen to base coordinates

Managing the Display

- **display** Displays all the Window's Views
- **displayIfNeeded** Displays all the Window's Views that need it
- **disableDisplay** Inhibits Views from drawing in the Window
- **(BOOL)isDisplayEnabled** Returns whether Views can draw in the Window
- **reenableDisplay** Reenables drawing by Views in the Window
- **flushWindow** Flushes the Window's buffer to the screen
- **flushWindowIfNeeded** Conditionally flushes the Window's buffer to the screen
- **disableFlushWindow** Disables flushing for a buffered Window
- **reenableFlushWindow** Reenables flushing for a buffered Window
- **(BOOL)isFlushWindowDisabled** Returns whether flushing is disabled
- **displayBorder** Displays the border and title bar
- **useOptimizedDrawing:(BOOL)flag** Sets whether Window's Views should optimize drawing
- **update** Implemented by subclasses (see Menu)

Screens and Window Depths

- **(const NXScreen *)screen** Returns the screen that (most of) the Window is on
- **(const NXScreen *)bestScreen** Returns the deepest screen that the Window is on
- + **(NXWindowDepth)defaultDepthLimit** Returns the maximum depth for the current context
- **setDepthLimit:(NXWindowDepth)limit** Sets the Window's depth limit to *limit*
- **(NXWindowDepth)depthLimit** Returns the Window's depth limit
- **setDynamicDepthLimit:(BOOL)flag** Sets whether the depth limit will depend on the screen
- **(BOOL)hasDynamicDepthLimit** Returns whether the depth limit depends on the screen
- **(BOOL)canStoreColor** Returns whether Window is deep enough to store colors

Cursor Management

- **addCursorRect:(const NXRect *)aRect
cursor:anObject
forView:aView** Adds a new cursor rectangle to the Window
- **removeCursorRect:(const NXRect *)aRect
cursor:anObject
forView:aView** Removes a cursor rectangle from the Window
- **invalidateCursorRectsForView:aView** Marks cursor rectangles invalid for *aView*
- **disableCursorRects** Disables all cursor rectangles in the Window
- **enableCursorRects** Enables cursor rectangles in the Window
- **discardCursorRects** Removes all cursor rectangles in the Window
- **resetCursorRects** Resets cursor rectangles for the Window

Handling User Actions and Events

- **close** Closes the Window
- **performClose:***sender* Simulates user clicking the close button
- **miniaturize:***sender* Hides the Window and displays its miniwindow
- **performMiniaturize:***sender* Simulates user clicking the miniaturize button
- **deminiaturize:***sender* Hides the miniwindow and redisplay the Window
- **setDocEdited:**(BOOL)*flag* Sets whether the Window's document has been edited
- (BOOL)**isDocEdited** Returns whether Window's document has been edited
- **windowExposed:**(NXEvent *)*theEvent* Redisplay exposed part of the Window
- **windowMoved:**(NXEvent *)*theEvent* Updates the frame rectangle
- **screenChanged:**(NXEvent *)*theEvent* Adjusts depth limit of Windows with a dynamic limit

Setting the Event Mask

- (int)**setEventMask:**(int)*newMask* Sets the Window's event mask to *newMask*
- (int)**addToEventMask:**(int)*newEvents* Adds *newEvents* to the event mask
- (int)**removeFromEventMask:**(int)*oldEvents* Removes *oldEvents* from the event mask
- (int)**eventMask** Returns the Window's event mask

Aiding Event Handling

- **getMouseLocation:**(NXPoint *)*thePoint* Provides current location of the cursor
- **setTrackingRect:**(const NXRect *)*aRect*
inside:(BOOL)*insideFlag*
owner:*anObject*
tag:(int)*trackNum*
left:(BOOL)*leftDown*
right:(BOOL)*rightDown* Sets a tracking rectangle within the Window
- **discardTrackingRect:**(int)*trackNum* Clears tracking rectangle within the Window
- **makeFirstResponder:***aResponder* Makes *aResponder* the first responder
- **firstResponder** Returns the first responder
- **sendEvent:**(NXEvent *)*theEvent* Dispatches mouse and keyboard events
- **rightMouseDown:**(NXEvent *)*theEvent* Handles right mouse-down events
- (BOOL)**commandKey:**(NXEvent *)*theEvent* Handles Command key-down events
- (BOOL)**tryToPerform:**(SEL)*anAction*
with:*anObject* Aids in dispatching action messages
- **setAvoidsActivation:**(BOOL)*flag* Establishes whether the application will become active when the user clicks in the Window
- (BOOL)**avoidsActivation** Returns the value set by **setAvoidsActivation:**

Dragging

- **registerForDraggedTypes:(const char *const *)pbTypes count:(int)count**
Registers the Pasteboard types that the Window will accept in an image-dragging session
- **unregisterDraggedTypes**
Unregisters the Window as a recipient of dragged images
- **dragImage:anImage**
at:(NXPoint *)location
offset:(NXPoint *)initialOffset
event:(NXEvent *)event
pasteboard:(Pasteboard *)pboard
source:sourceObject
slideBack:(BOOL)slideFlag
Instigates an image-dragging session

Services and Windows Menu Support

- **validRequestorForSendType:(NXAtom)typeSent andReturntype:(NXAtom)typeReturned**
Invoked by clicking a Services menu item
- **setExcludedFromWindowsMenu:(BOOL)flag**
Sets whether Window is left out of the Windows menu
- **(BOOL)isExcludedFromWindowsMenu**
Returns whether Window is left out of Windows menu

Printing

- **printPSCode:sender**
Prints all the Window's Views
- **smartPrintPSCode:sender**
Prints all the Window's Views
- **faxPSCode:sender**
Faxes all the Window's Views
- **smartFaxPSCode:sender**
Faxes all the Window's Views
- **openSpoolFile:(char *)filename**
Opens *filename* for print spooling
- **spoolFile:(const char *)filename**
Spools *filename* to the printer
- **copyPSCodeInside:(const NXRect *)rect to:(NXStream *)stream**
Writes PostScript code for the rectangle to *stream*
- **(BOOL)knowsPagesFirst:(int *)firstPageNum last:(int *)lastPageNum**
Returns whether the Window paginates itself
- **(BOOL)getRect:(NXRect *)theRect forPage:(int)page**
Provides how much of the Window fits on *page*
- **placePrintRect:(const NXRect *)aRect offset:(NXPoint *)location**
Locates the printing rectangle on the page
- **(float)heightAdjustLimit**
Returns how much of a page can go on next page
- **(float)widthAdjustLimit**
Returns how much of a page can go on next page
- **beginPSOutput**
Initializes the printing environment

- **beginPrologueBBox:**(const NXRect *)*boundingBox*
creationDate:(const char *)*dateCreated*
createdBy:(const char *)*anApplication*
fonts:(const char *)*fontNames*
forWhom:(const char *)*user*
pages:(int)*numPages*
title:(const char *)*aTitle*
Writes the beginning of the prologue for a print job
- **endHeaderComments**
Writes the end of a PostScript comment section
- **endPrologue**
Writes the end of the prologue
- **beginSetup**
Writes the beginning of the document setup section
- **endSetup**
Writes the end of the document setup section
- **beginPage:**(int)*ordinalNum*
label:(const char *)*aString*
bBox:(const NXRect *)*pageRect*
fonts:(const char *)*fontNames*
Writes a page separator
- **beginPageSetupRect:**(const NXRect *)*aRect*
placement:(const NXPoint *)*location*
Writes the beginning of a page setup section
- **endPageSetup**
Writes the end of a page setup section
- **endPage**
Writes the end of a page description
- **beginTrailer**
Writes the beginning of trailer for the print job
- **endTrailer**
Writes the end of the trailer
- **endPSOutput**
Finishes the printing job

Archiving

- **read:**(NXTypedStream *)*stream*
Reads the Window from the typed stream
- **write:**(NXTypedStream *)*stream*
Writes the Window to the typed stream
- **awake**
Redisplays and reinitializes the Window

Assigning a Delegate

- **setDelegate:***anObject*
Makes *anObject* the Window's delegate
- **delegate**
Returns the Window's delegate

Implemented by the Delegate

- **windowWillClose:***sender*
Notifies delegate that the Window is about to close
- **windowWillReturnFieldEditor:***sender*
toObject:*client*
Lets delegate provide another Text object
- **windowWillResize:***sender*
toSize:(NXSize *)*frameSize*
Lets delegate constrain resizing

Protocols

NXDraggingDestination

Adopted By: no NeXTSTEP classes

Before the Image is Released

- (NXDragOperation)**draggingEntered:**(id <NXDraggingInfo>)*sender*
Invoked when the dragged image enters the destination
- (NXDragOperation)**draggingUpdated:**(id <NXDraggingInfo>)*sender*
Invoked periodically while the image is over the destination
- **draggingExited:**(id <NXDraggingInfo>)*sender* Invoked when the dragged image exits the destination

After the Image is Released

- (BOOL)**prepareForDragOperation:**(id <NXDraggingInfo>)*sender*
Invoked when the image is released
- (BOOL)**performDragOperation:**(id <NXDraggingInfo>)*sender*
Gives the destination an opportunity to perform the dragging operation
- **concludeDragOperation:**(id <NXDraggingInfo>)*sender*
Invoked when the dragging operation is complete.

NXDraggingInfo

Adopted By: no NeXTSTEP classes

Dragging-Session Information

- (BOOL)**isDraggingSourceLocal** Returns whether the source and destination are in the same application
- **draggingSource** Returns the source of the dragged image

- (NXDragOperation)**draggingSourceOperationMask** Returns the operation mask declared by the source
- **draggingDestinationWindow** Returns the destination’s Window
- (Pasteboard *)**draggingPasteboard** Returns the Pasteboard that holds the dragged data
- (int)**draggingSequenceNumber** Returns a number that identifies the dragging session
- (NXPoint)**draggingLocation** Returns the cursor’s location

Image Information

- (NXImage *)**draggedImage** Returns the NXImage object that’s being dragged
- (NXImage *)**draggedImageCopy** Returns a copy of the NXImage object that’s being dragged
- (NXPoint)**draggedImageLocation** Returns the current location of the dragged image’s origin

Sliding the Image

- **slideDraggedImageTo:(NXPoint *)screenPoint** Slides the image to the given location in the screen coordinate system

NXDraggingSource (informal protocol)

Category Of: Object

Querying the Source

- (NXDragOperation)**draggingSourceOperationMaskForLocal:(BOOL)isLocal**
Returns a mask giving the operations that can be performed on the dragged image’s data

Informing the Source

- **draggedImage:(NXImage *)image
beganAt:(NXPoint *)screenPoint** Invoked when the dragged image is displayed but before it starts following the mouse
- **draggedImage:(NXImage *)image
endedAt:(NXPoint *)screenPoint
deposited:(BOOL)didDeposit** Invoked after the dragged image has been released and the dragging destination has been given a chance to operate

NXNibNotification

(informal protocol)

Category Of: Object

Response on Loading

– `awakeFromNib` Implemented to respond to notifications sent after objects have been loaded from a nib file

NXPrintingUserInterface

(informal protocol)

Category Of: Object

Printer Panel

– (BOOL)`shouldRunPrintPanel:aView` Implemented to indicate whether the Print panel (or Fax panel) should be displayed to the user

NXRTFDErrorHandler

Adopted By: No NeXTSTEP Classes

Notification of Overwrite

– (BOOL)`attemptOverwrite:(const char *)filename` Notifies the receiver that the user is attempting to save an RTFD document in a location for which the user doesn't have search permission

Application Kit Functions

Rectangle Functions

Modify a rectangle:

```
void          NXSetRect(NXRect *aRect, NXCoord x, NXCoord y, NXCoord width,
                       NXCoord height)
void          NXOffsetRect(NXRect *aRect, NXCoord dx, NXCoord dy)
void          NXInsetRect(NXRect *aRect, NXCoord dx, NXCoord dy)
void          NXIntegralRect(NXRect *aRect)
NXRect *     NXDivideRect(NXRect *aRect, NXRect *bRect, NXCoord slice, int edge)
```

Test graphic relationships:

```
BOOL         NXMouseInRect(const NXPoint *aPoint, const NXRect *aRect, BOOL flipped)
BOOL         NXPointInRect(const NXPoint *aPoint, const NXRect *aRect)
BOOL         NXIntersectsRect(const NXRect *aRect, const NXRect *bRect)
BOOL         NXContainsRect(const NXRect *aRect, const NXRect *bRect)
BOOL         NXEqualRect(const NXRect *aRect, const NXRect *bRect)
BOOL         NXEmptyRect(const NXRect *aRect)
```

Compute third rectangle from two rectangles:

```
NXRect *     NXUnionRect(const NXRect *aRect, NXRect *bRect)
NXRect *     NXIntersectionRect(const NXRect *aRect, NXRect *bRect)
```

Optimize drawing:

```
void         NXRectClip(const NXRect *aRect)
void         NXRectClipList(const NXRect *rects, int count)
void         NXRectFill(const NXRect *aRect)
void         NXRectFillList(const NXRect *rects, int count)
void         NXRectFillListWithGrays(const NXRect *rects, const float *grays, int count)
void         NXEraseRect(const NXRect *aRect)
void         NXHighlightRect(const NXRect *aRect)
```

Draw a bordered rectangle:

```
void         NXDrawButton(const NXRect *aRect, const NXRect *clipRect)
void         NXDrawGrayBezel(const NXRect *aRect, const NXRect *clipRect)
```

void	NXDrawGroove (const NXRect *aRect, const NXRect *clipRect)
void	NXDrawWhiteBezel (const NXRect *aRect, const NXRect *clipRect)
NXRect *	NXDrawTiledRects (NXRect *boundsRect, const NXRect *clipRect, const int *sides, const float *grays, int count)
void	NXFrameRect (const NXRect *aRect)
void	NXFrameRectWithWidth (const NXRect *aRect, NXCoord frameWidth)

Query an NXRect structure:

NXCoord	NX_X (NXRect *aRect)
NXCoord	NX_Y (NXRect *aRect)
NXCoord	NX_WIDTH (NXRect *aRect)
NXCoord	NX_HEIGHT (NXRect *aRect)
NXCoord	NX_MAXX (NXRect *aRect)
NXCoord	NX_MAXY (NXRect *aRect)
NXCoord	NX_MIDX (NXRect *aRect)
NXCoord	NX_MIDY (NXRect *aRect)

Color Functions

Specify a color value:

NXColor	NXConvertRGBAToColor (float red, float green, float blue, float alpha)
NXColor	NXConvertCMYKAToColor (float cyan, float magenta, float yellow, float black, float alpha)
NXColor	NXConvertHSBAToColor (float hue, float saturation, float brightness, float alpha)
NXColor	NXConvertGrayAlphaToColor (float gray, float alpha)
NXColor	NXConvertRGBToColor (float red, float green, float blue)
NXColor	NXConvertCMYKToColor (float cyan, float magenta, float yellow, float black)
NXColor	NXConvertHSBToColor (float hue, float saturation, float brightness)
NXColor	NXConvertGrayToColor (float gray)

Convert a color value to its standard components:

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 <i>color</i> , float <i>*cyan</i> , float <i>*magenta</i> , float <i>*yellow</i> , float <i>*black</i>)
void	NXConvertColorToHSB (NXColor <i>color</i> , float <i>*hue</i> , float <i>*saturation</i> , float <i>*brightness</i>)
void	NXConvertColorToGray (NXColor <i>color</i> , float <i>*gray</i>)

Modify a color by changing one of its components:

NXColor	NXChangeRedComponent (NXColor <i>color</i> , float <i>red</i>)
NXColor	NXChangeGreenComponent (NXColor <i>color</i> , float <i>green</i>)
NXColor	NXChangeBlueComponent (NXColor <i>color</i> , float <i>blue</i>)
NXColor	NXChangeCyanComponent (NXColor <i>color</i> , float <i>cyan</i>)
NXColor	NXChangeMagentaComponent (NXColor <i>color</i> , float <i>magenta</i>)
NXColor	NXChangeYellowComponent (NXColor <i>color</i> , float <i>yellow</i>)
NXColor	NXChangeBlackComponent (NXColor <i>color</i> , float <i>black</i>)
NXColor	NXChangeHueComponent (NXColor <i>color</i> , float <i>hue</i>)
NXColor	NXChangeSaturationComponent (NXColor <i>color</i> , float <i>saturation</i>)
NXColor	NXChangeBrightnessComponent (NXColor <i>color</i> , float <i>brightness</i>)
NXColor	NXChangeGrayComponent (NXColor <i>color</i> , float <i>gray</i>)
NXColor	NXChangeAlphaComponent (NXColor <i>color</i> , float <i>alpha</i>)

Isolate one component of a color:

float	NXRedComponent (NXColor <i>color</i>)
float	NXGreenComponent (NXColor <i>color</i>)
float	NXBlueComponent (NXColor <i>color</i>)
float	NXCyanComponent (NXColor <i>color</i>)
float	NXMagentaComponent (NXColor <i>color</i>)
float	NXYellowComponent (NXColor <i>color</i>)
float	NXBlackComponent (NXColor <i>color</i>)
float	NXHueComponent (NXColor <i>color</i>)
float	NXSaturationComponent (NXColor <i>color</i>)
float	NXBrightnessComponent (NXColor <i>color</i>)
float	NXGrayComponent (NXColor <i>color</i>)
float	NXAlphaComponent (NXColor <i>color</i>)

Test whether two colors are the same:

BOOL	NXEqualColor (NXColor <i>oneColor</i> , NXColor <i>anotherColor</i>)
------	--

Get information about color space and window depth:

NXColorSpace	NXColorSpaceFromDepth (NXWindowDepth <i>depth</i>)
int	NXBPSFromDepth (NXWindowDepth <i>depth</i>)

int NXNumberOfColorComponents(NXColorSpace *space*)
BOOL NXGetBestDepth(NXWindowDepth **depth*, int *numColors*, int *bps*)

Read and write a color from a typed stream:

NXColor NXReadColor(NXTypedStream **stream*)
void NXWriteColor(NXTypedStream **stream*, NXColor *color*)

Read and write a color from a pasteboard:

NXColor NXReadColorFromPasteboard(id *pasteboard*)
void NXWriteColorToPasteboard(id *pasteboard*, NXColor *color*)

Set the current color:

void NXSetColor(NXColor *color*)

Reading the color at a screen position:

NXColor NXReadPixel(const NXPoint **location*)

Associate named colors with their color lists

const char * NXColorListName (NXColor *color*)
const char * NXColorName (NXColor *color*)
BOOL NXFindColorNamed (const char **colorList*, const char **colorName*,
 NXColor **color*)

Text Functions

Filter characters entered into Text object:

unsigned short NXFieldFilter(unsigned short *theChar*, int *flags*, unsigned short *charSet*)
unsigned short NXEditorFilter(unsigned short *theChar*, int *flags*, unsigned short *charSet*)

Calculate or draw a line of text (in Text object):

int NXScanALine(id *self*, NXLayoutInfo **layInfo*)
int NXDrawALine(id *self*, NXLayoutInfo **layInfo*)

Error-Handling Functions

Set and return an error handler:

```
void                NXDefaultTopLevelErrorHandler(NXHandler *errorState)
NXTopLevelErrorHandler * NXSetTopLevelErrorHandler(NXTopLevelErrorHandler *proc)
                    /* a macro */
NXTopLevelErrorHandler * NXTopLevelErrorHandler(void) /* a macro */
```

Manage error reporting:

```
void                NXRegisterErrorReporter(int min, int max, NXErrorReporter *proc)
void                NXRemoveErrorReporter(int code)
void                NXReportError(NXHandler *errorState)
```

Write a formatted error string:

```
void                NXLogError(const char *format, ...)
```

Typed Stream Functions

Read or write NeXT-defined data types from or to a typed stream:

```
void                NXReadPoint(NXTypedStream *typedStream, NXPoint *aPoint)
void                NXWritePoint(NXTypedStream *typedStream, const NXPoint *aPoint)
void                NXReadSize(NXTypedStream *typedStream, NXSize *aSize)
void                NXWriteSize(NXTypedStream *typedStream, const NXSize *aSize)
void                NXReadRect(NXTypedStream *typedStream, NXRect *aRect)
void                NXWriteRect(NXTypedStream *typedStream, const NXRect *aRect)
```

Remote Messaging Functions

Save data received in a remote message:

```
char *              NXCopiedInputData(int parameter)
char *              NXCopiedOutputData(int parameter)
```


Get send rights to an application port:

```
port_t      NXPortFromName(const char *name, const char *host)
port_t      NXPortNameLookup(const char *name, const char *host)
```

Match an Objective-C method and a receiver to a remote message:

```
NXRemoteMethod * NXRemoteMethodFromSel(SEL aSelector, NXRemoteMethod *methods)
id               NXResponsibleDelegate(id aListener, SEL aSelector)
```

Services Menu Functions

Determine whether an item is included in Services menus:

```
int          NXSetServicesMenuItemEnabled(const char *item, BOOL flag)
BOOL        NXIsServicesMenuItemEnabled(const char *item)
```

Programatically invoke a service:

```
BOOL        NXPerformService(const char *item, Pasteboard *pboard)
```

Force Services menu to update based on new services:

```
void        NXUpdateDynamicServices(void)
```

Event Function

Access event record in event queue:

```
NXEvent *    NXGetOrPeekEvent(DPSContext context, NXEvent *anEvent, int mask,
                             double timeout, int threshold, int peek)
```

Memory Allocation Functions

Macros to allocate memory:

```
type-name *  NX_MALLOC(type-name *var, type-name, int num)
type-name *  NX_REALLOC(type-name *var, type-name, int num)
void         NX_FREE(void *pointer)
```

Allocate a variable-sized array:

```
NXChunk *    NXChunkMalloc(int growBy, int initUsed)
NXChunk *    NXChunkRealloc(NXChunk *pc)
NXChunk *    NXChunkGrow(NXChunk *pc, int newUsed)
NXChunk *    NXChunkCopy(NXChunk *pc, NXChunk *dpc)
NXChunk *    NXChunkZoneMalloc(int growBy, int initUsed, NXZone *zone)
NXChunk *    NXChunkZoneRealloc(NXChunk *pc, NXZone *zone)
NXChunk *    NXChunkZoneGrow(NXChunk *pc, int newUsed, NXZone *zone)
NXChunk *    NXChunkZoneCopy(NXChunk *pc, NXChunk *dpc, NXZone *zone)
```

Macros to allocate zone memory:

```
type-name *  NX_ZONEMALLOC(NXZone *zone, type-name *var, type-name, int num)
type-name *  NX_ZONEREALLOC(NXZone *zone, type-name *var, type-name, int num)
```

Other Application Kit Functions

Get user's home directory and name:

```
const char *  NXHomeDirectory(void)
const char *  NXUserName(void)
```

Synchronize the application with the Window Server:

```
void          NXPing(void)
```

Find dimensions of specified paper type:

```
const NXSize * NXFindPaperSize(const char *paperName)
```

Play the system beep:

```
void          NXBeep(void)
```

Set up timer events:

```
NXTrackingTimer * NXBeginTimer(NXTrackingTimer *timer, double delay, double period)
void              NXEndTimer(NXTrackingTimer *timer)
```

Allow journaling during direct mouse tracking:

void NXJournalMouse(void)

Set or copy current graphics state object:

void NXSetGState(int *gstate*)
void NXCopyCurrentGState(int *gstate*)

Report user's request to abort:

BOOL NXUserAborted(void)
void NXResetUserAbort(void)

Return file-related pasteboard types:

NXAtom NXCreateFileContentsPboardType(const char **fileType*)
NXAtom NXCreateFilenamePboardType(const char **filename*)
const char * NXGetFileType(const char **pboardType*)
const char ** NXGetFileTypes(const char *const **pboardTypes*)

Find unique files from a path:

int NXCompleteFilename(char **path*, int *maxPathSize*)

Draw a distinctive outline around linked data:

void NXFrameLinkRect(const NXRect **aRect*, BOOL *isDestination*)
float NXLinkFrameThickness(void)

Get the amount of memory used by the Window Server:

int NXGetWindowServerMemory(DPSContext *context*, int **virtualMemory*,
 int **windowBackingMemory*, NXStream **windowDumpStream*)

Macro to write an error message:

void NX_ASSERT(int *exp*, char **msg*)

Macro for debugging Display PostScript:

void NX_PSDEBUG(void)

Types and Constants

Defined Types

NXAcknowledge

```
typedef struct _NXAcknowledge {  
    msg_header_t header;  
    msg_type_t sequenceType;  
    int sequence;  
    msg_type_t errorType;  
    int error;  
} NXAcknowledge
```

NXAppkitErrorTokens

```
typedef enum _NXAppkitErrorTokens {  
    NX_longLine = NX_APPKIT_ERROR_BASE,  
    NX_nullSel,  
    NX_wordTablesWrite,  
    NX_wordTablesRead,  
    NX_textBadRead,  
    NX_textBadWrite,  
    NX_powerOff,  
    NX_pasteboardComm,  
    NX_mallocError,  
    NX_printingComm,  
    NX_abortModal,  
    NX_abortPrinting,  
    NX_illegalSelector,  
    NX_appkitVMError,  
    NX_badRtfDirective,  
    NX_badRtfFontTable,  
    NX_badRtfStyleSheet,  
    NX_newerTypedStream,  
    NX_tiffError,  
    NX_printPackageError,  
    NX_badRtfColorTable,  
    NX_journalAborted,  
    NX_draggingError,
```

```
NX_colorUnknown,
NX_colorBadIO,
NX_colorNotEditable,
NX_badBitmapParams,
NX_windowServerComm,
NX_unavailableFont,
NX_PPDIncludeNotFound,
NX_PPDParseError,
NX_PPDIncludeStackOverflow,
NX_PPDIncludeStackUnderflow,
NX_rtfPropOverflow
} NXAppkitErrorTokens;
```

NXBreakArray

```
typedef struct _NXBreakArray {
    NXChunk chunk;
    NXLineDesc breaks[1];
} NXBreakArray;
```

NXCharArray

```
typedef struct _NXCharArray {
    NXChunk chunk;
    wchar text[1];
} NXCharArray;
```

NXCharFilterFunc

```
typedef unsigned short (*NXCharFilterFunc)
(unsigned short charCode,
 int flags,
 unsigned short charSet);
```

NXCharMetrics

```
typedef struct {
    short charCode;
    unsigned char numKernPairs;
    unsigned char reserved;
    float xWidth;
    int name;
    float bbox[4];
    int kernPairIndex;
} NXCharMetrics;
```

NXChunk

```
typedef struct _NXChunk {
    short growby;
    int allocated;
    int used;
} NXChunk;
```

NXColorSpace

```
typedef enum _NXColorSpace {
    NX_CustomColorSpace = -1,
    NX_OneIsBlackColorSpace = 0,
    NX_OneIsWhiteColorSpace = 1,
    NX_RGBColorSpace = 2,
    NX_CMYKColorSpace = 5
} NXColorSpace;
```

NXCompositeChar

```
typedef struct {
    int compCharIndex;
    int numParts;
    int firstPartIndex;
} NXCompositeChar;
```

NXCompositeCharPart

```
typedef struct {
    int partIndex;
    float dx;
    float dy;
} NXCompositeCharPart;
```

NXDataLinkDisposition

```
typedef enum _NXDataLinkDisposition {
    NX_LinkInDestination = 1,
    NX_LinkInSource = 2,
    NX_LinkBroken = 3
} NXDataLinkDisposition
```

NXDataLinkNumber

```
typedef int NXDataLinkNumber;
```

NXDataLinkUpdateMode

```
typedef enum _NXDataLinkUpdateMode {
    NX_UpdateContinuously = 1,
    NX_UpdateWhenSourceSaved = 2,
    NX_UpdateManually = 3,
    NX_UpdateNever = 4
} NXDataLinkUpdateMode
```

NXDragOperation

```
typedef enum _NXDragOperation {
    NX_DragOperationNone = 0,
    NX_DragOperationCopy = 1,
    NX_DragOperationLink = 2,
    NX_DragOperationGeneric = 4,
    NX_DragOperationPrivate = 8,
    NX_DragOperationAll = 15
} NXDragOperation;
```

NXEncodedLigature

```
typedef struct {
    unsigned char firstChar;
    unsigned char secondChar;
    unsigned char ligatureChar;
    unsigned char reserved;
} NXEncodedLigature;
```

NXErrorReporter

```
typedef void NXErrorReporter(NXHandler *errorState);
```

NXFaceInfo

```
typedef struct _NXFaceInfo {
    NXFontMetrics *fontMetrics;
    int flags;
    struct _fontFlags {
        unsigned int usedInDoc:1;
        unsigned int usedInPage:1;
        unsigned int usedInSheet:1;
        unsigned int _PADDING:13;
    } fontFlags;
    struct _NXFaceInfo *nextFInfo;
} NXFaceInfo;
```

NXFontMetrics

```
typedef struct _NXFontMetrics {
    char *formatVersion;
    char *name;
    char *fullName;
    char *familyName;
    char *weight;
    float italicAngle;
    char isFixedPitch;
    char isScreenFont;
    short screenFontSize;
    float fontBBox[4];
    float underlinePosition;
    float underlineThickness;
    char *version;
```



```

char *notice;
char *encodingScheme;
float capHeight;
float xHeight;
float ascender;
float descender;
short hasYWidths;
float *widths;
unsigned int widthsLength;
char *strings;
unsigned int stringsLength;
char hasXYKerns;
char reserved;
short *encoding;
float *yWidths;
NXCharMetrics *charMetrics;
int numCharMetrics;
NXLigature *ligatures;
int numLigatures;
NXEncodedLigature *encLigatures;
int numEncLigatures;
union {
    NXKernPair *kernPairs;
    NXKernXPair *kernXPairs;
} kerns;
int numKernPairs;
NXTrackKern *trackKerns;
int numTrackKerns;
NXCompositeChar *compositeChars;
int numCompositeChars;
NXCompositeCharPart *compositeCharParts;
int numCompositeCharParts;
} NXFontMetrics;

```

NXFontTraitMask

```
typedef unsigned int NXFontTraitMask;
```

NXFSM

```
typedef struct _NXFSM {
    const struct _NXFSM *next;
    short delta;
    short token;
} NXFSM;
```

NXHeightChange

```
typedef struct _NXHeightChange {
    NXLineDesc lineDesc;
    NXHeightInfo heightInfo;
} NXHeightChange;
```

NXHeightInfo

```
typedef struct _NXHeightInfo {
    NXCoord newHeight;
    NXCoord oldHeight;
    NXLineDesc lineDesc;
} NXHeightInfo;
```

NXJournalHeader

```
typedef struct {
    int version;
    unsigned int offsetToAppNames;
    unsigned int lastEventTime;
    unsigned int reserved1;
    unsigned int reserved2;
} NXJournalHeader
```

NXKernPair

```
typedef struct {
    int secondCharIndex;
    float dx;
    float dy;
} NXKernPair;
```

NXKernXPair

```
typedef struct {
    int secondCharIndex;
    float dx;
} NXKernXPair;
```

NXLay

```
typedef struct _NXLay {
    NXCoord x;
    NXCoord y;
    short offset;
    short chars;
    id font;
    void *paraStyle;
    NXRun *run;
    NXLayFlags IFlags;
} NXLay;
```

NXLayArray

```
typedef struct _NXLayArray {
    NXChunk chunk;
    NXLay lays[1];
} NXLayArray;
```

NXLayFlags

```
typedef struct {
    unsigned int mustMove:1;
    unsigned int isMoveChar:1;
    unsigned int RESERVED:14;
} NXLayFlags;
```

NXLayInfo

```
typedef struct _NXLayInfo {
    NXRect rect;
    NXCoord descent;
    NXCoord width;
    NXCoord left;
    NXCoord right;
    NXCoord rightIndent;
    NXLayArray *lays;
    NXWidthArray *widths;
    NXCharArray *chars;
    NXTextCache cache;
    NXRect *textClipRect;
    struct _IFlags {
        unsigned int horizCanGrow:1;
        unsigned int vertCanGrow:1;
        unsigned int erase:1;
        unsigned int ping:1;
        unsigned int endsParagraph:1;
        unsigned int resetCache:1;
        unsigned int RESERVED:10;
    } IFlags;
} NXLayInfo;
```

NXLigature

```
typedef struct {
    int firstCharIndex;
    int secondCharIndex;
    int ligatureIndex;
} NXLigature;
```

NXLineDesc

```
typedef short NXLineDesc;
```

NXLinkEnumerationState

```
typedef struct {
    void *a;
    void *b;
} NXLinkEnumerationState
```

NXMeasurementUnit

```
typedef struct _NXMeasurementUnit {
    NX_UnitInch = 0,
    NX_UnitCentimeter = 1,
    NX_UnitPoint = 2,
    NX_UnitPica = 3
} NXMeasurementUnit;
```

NXMessage

```
typedef struct _NXMessage {
    msg_header_t header;
    msg_type_t sequenceType;
    int sequence;
    msg_type_t actionType;
    char action[NX_MAXMESSAGE];
} NXMessage
```

NXModalSession

```
typedef struct _NXModalSession {
    id app;
    id window;
    struct _NXModalSession *prevSession;
    int oldRunningCount;
    BOOL oldDoesHide;
    BOOL freeMe;
    int winNum;
    NXHandler *errorData;
    int reserved1;
    int reserved2;
} NXModalSession;
```

NXParagraphProp

```
typedef enum {
    NX_LEFTALIGN = NX_LEFTALIGNED,
    NX_RIGHTALIGN = NX_RIGHTALIGNED,
    NX_CENTERALIGN = NX_CENTERED,
    NX_JUSTALIGN = NX_JUSTIFIED,
    NX_FIRSTINDENT,
    NX_INDENT,
```

```
NX_ADDTAB,  
NX_REMOVETAB,  
NX_LEFTMARGIN,  
NX_RIGHTMARGIN  
} NXParagraphProp;
```

NXParamValue

```
typedef union {  
    int ival;  
    double dval;  
    port_t pval;  
    struct _bval {  
        char *p;  
        int len;  
    } bval;  
} NXParamValue;
```

NXRect

```
typedef struct _NXRect {  
    NXPoint origin;  
    NXSize size;  
} NXRect
```

NXRemoteMethod

```
typedef struct _NXRemoteMethod {  
    SEL key;  
    char *types;  
} NXRemoteMethod
```

NXResponse

```
typedef struct _NXResponse {  
    msg_header_t header;  
    msg_type_t sequenceType;  
    int sequence;  
} NXResponse
```

NXRTFDError

```
typedef enum {  
    NX_RTFDErrorNone  
    NX_RTFDErrorSaveAborted,  
    NX_RTFDErrorUnableToWriteFile,  
    NX_RTFDErrorUnableToCloseFile,  
    NX_RTFDErrorUnableToCreatePackage  
    NX_RTFDErrorUnableToCreateBackup,  
    NX_RTFDErrorUnableToDeleteBackup,  
    NX_RTFDErrorUnableToDeleteTemp,  
    NX_RTFDErrorUnableToDeleteOriginal,  
    NX_RTFDErrorFileDoesntExist,  
    NX_RTFDErrorUnableToReadFile,  
    NX_RTFDErrorInsufficientAccess,  
    NX_RTFDErrorMalformedRTFD  
} NXRTFDError;
```

NXRun

```
typedef struct _NXRun {  
    id font;  
    int chars;  
    void *paraStyle;  
    float textGray;  
    int textRGBColor;  
    unsigned char superscript;  
    unsigned char subscript;  
    id info;  
    NXRunFlags rFlags;  
} NXRun;
```

NXRunArray

```
typedef struct _NXRunArray {  
    NXChunk chunk;  
    NXRun runs[1];  
} NXRunArray;
```

NXRunFlags

```
typedef struct {
    unsigned int underline:1;
    unsigned int dummy:1;
    unsigned int subclassWantsRTF:1;
    unsigned int graphic:1;
    unsigned int RESERVED:12;
} NXRunFlags;
```

NXScreen

```
typedef struct _NXScreen {
    int screenNumber;
    NXRect screenBounds;
    NXWindowDepth depth;
} NXScreen;
```

NXSelPt

```
typedef struct _NXSelPt {
    int cp;
    int line;
    NXCoord x;
    NXCoord y;
    int c1st;
    NXCoord ht;
} NXSelPt;
```

NXSpellCheckMode

```
typedef enum {
    NX_CheckSpelling,
    NX_CheckSpellingToEnd,
    NX_CheckSpellingFromStart,
    NX_CheckSpellingInSelection,
    NX_CountWords,
    NX_CountWordsToEnd,
    NX_CountWordsInSelection
} NXSpellCheckMode;
```


NXStreamSeekMode

```
typedef enum {
    NX_StreamStart,
    NX_StreamCurrent,
    NX_StreamEnd
} NXStreamSeekMode;
```

NXStringOrderTable

```
typedef struct {
    unsigned char primary[256];
    unsigned char secondary[256];
    unsigned char primaryCI[256];
    unsigned char secondaryCI[256];
} NXStringOrderTable;
```

NXTabStop

```
typedef struct _NXTabStop {
    short kind;
    NXCoord x;
} NXTabStop;
```

NXTextBlock

```
typedef struct _NXTextBlock {
    struct _NXTextBlock *next;
    struct _NXTextBlock *prior;
    struct _tbFlags {
        unsigned int malloced:1;
        unsigned int PAD:15;
    } tbFlags;
    short chars;
    wchar *text;
} NXTextBlock;
```

NXTextCache

```
typedef struct _NXTextCache {
    int curPos;
    NXRun *curRun;
    int runFirstPos;
    NXTextBlock *curBlock;
    int blockFirstPos;
} NXTextCache;
```

NXTextFilterFunc

```
typedef char *(*NXTextFilterFunc)
    (id self,
     unsigned char *insertText,
     int *insertLength,
     int position);
```

NXTextFunc

```
typedef int (*NXTextFunc)
    (id self,
     NXLayoutInfo *layInfo);
```

NXTextStyle

```
typedef struct _NXTextStyle {
    NXCoord indent1st;
    NXCoord indent2nd;
    NXCoord lineHt;
    NXCoord descentLine;
    short alignment;
    short numTabs;
    NXTabStop *tabs;
} NXTextStyle;
```

NXTopLevelErrorHandler

```
typedef void NXTopLevelErrorHandler(NXHandler *errorState);
```

NXTrackingTimer

```
typedef struct _NXTrackingTimer {
    double delay;
    double period;
    DPSTimedEntry te;
    BOOL freeMe;
    BOOL firstTime;
    NXHandler *errorData;
    int reserved1;
    int reserved2;
} NXTrackingTimer;
```

NXTrackKern

```
typedef struct {
    int degree;
    float minPointSize;
    float minKernAmount;
    float maxPointSize;
    float maxKernAmount;
} NXTrackKern;
```

NXWidthArray

```
typedef struct _NXWidthArray {
    NXChunk chunk;
    NXCoord widths[1];
} NXWidthArray;
```

NXWindowDepth

```
typedef enum _NXWindowDepth {
    NX_DefaultDepth = 0,
    NX_TwoBitGrayDepth = 258,
    NX_EightBitGrayDepth = 264,
    NX_TwelveBitRGBDepth = 516,
    NX_TwentyFourBitRGBDepth = 520
} NXWindowDepth;
```

wchar

```
typedef unsigned char wchar;
```

Symbolic Constants

Application Status Value

Kit-Defined Subtypes

NX_WINEXPOSED	0
NX_APPACT	1
NX_APPDEACT	2
NX_WINMOVED	4
NX_SCREENCHANGED	8

System-Defined Subtype

NX_POWEROFF	1
-------------	---

Error Base Constants Value

NX_APPKIT_ERROR_BASE	3000
NX_APP_ERROR_BASE	10000000

Bits per Character or Integer

	Value
NBITSCHAR	8
NBITSINT	(sizeof(int)*NBITSCHAR)

Boolean Constants Value

TRUE	1
FALSE	0

Box Borders

NX_NOBORDER
NX_LINE
NX_BEZEL
NX_GROOVE

Box Title Positions

NX_NOTITLE
NX_ABOVETOP
NX_ATTOP
NX_BELOWTOP
NX_ABOVEBOTTOM
NX_ATBOTTOM
NX_BELOWBOTTOM

Button and ButtonCell Highlight/Display Types

NX_MOMENTATYPUSH
NX_PUSHONPUSHOFF
NX_TOGGLE
NX_SWITCH
NX_RADIOBUTTON
NX_MOMENTARYCHANGE
NX_ONOFF

Button and ButtonCell Icon Positions

NX_TITLEONLY
NX_ICONONLY
NX_ICONLEFT
NX_ICONRIGHT
NX_ICONBELOW
NX_ICONABOVE
NX_ICONOVERLAPS

Cell and ButtonCell Parameter Constants

NX_CELLDISABLED
NX_CELLSTATE
NX_CELLEEDITABLE
NX_CELLHIGHLIGHTED
NX_LIGHTBYCONTENTS
NX_LIGHTBYGRAY
NX_LIGHTBYBACKGROUND
NX_ICONISKEYEQUIVALENT
NX_OVERLAPPINGICON
NX_ICONHORIZONTAL
NX_ICONLEFTORBOTTOM
NX_CHANGECONTENTS
NX_BUTTONINSET

Cell Data Entry Types

NX_ANYTYPE
NX_INTTYPE
NX_POSINTTYPE
NX_FLOATTYPE
NX_POSFLOATTYPE
NX_DOUBLETYPE
NX_POSDOUBLETYPE

Cell sendActionOn: Flag

NX_PERIODICMASK

Cell Types

NX_NULLCELL
NX_TEXTCELL
NX_ICONCELL

Color Panel Modes

NX_GRAYMODE
NX_RGBMODE
NX_CMYKMODE
NX_HSBMODE
NX_CUSTOMPALETTE
NX_CUSTOMCOLORMODE
NX_BEGINMODE

Color Panel Mode Masks

NX_GRAYMODEMASK
NX_RGBMODEMASK
NX_CMYKMODEMASK
NX_HSBMODEMASK
NX_CUSTOMPALETTE
NX_LISTMODEMASK
NX_WHEELMODEMASK
NX_ALLMODESMASK

**Color Picker Insertion
Order Constants**

	Value
NX_WHEEL_INSERTION	0.50
NX_SLIDERS_INSERTION	0.51
NX_CUSTOMPALETTE_INSERTION	0.52
NX_LIST_INSERTION	0.53

Drawing Activity States**Meaning**

NX_DRAWING	Drawing to the screen
NX_PRINTING	Spooling to a printer
NX_COPYING	Copying to a pasteboard

Event Thresholds State**Value**

NX_BASETHRESHOLD	1
NX_RUNMODALTHRESHOLD	5
NX_MODALRESPHRESHOLD	10

Figure Space Constant

NX_FIGSPACE

Font Attribute Constants

NX_FONTHEADER
NX_FONTMETRICS
NX_FONTWIDTHS
NX_FONTCHARDATA
NX_FONTKERNING
NX_FONTCOMPOSITES

**Font Conversion
Constants****Value**

NX_NOFONTCHANGE	0
NX_VIAPANEL	1
NX_ADDTRAIT	2
NX_SIZEUP	3
NX_SIZEDOWN	4
NX_HEAVIER	5
NX_LIGHTER	6
NX_REMOVETRAIT	7

Font Matrix Constants

NX_IDENTITYMATRIX
NX_FLIPPEDMATRIX

Font Trait Constants

	Value
NX_ITALIC	0x00000001
NX_BOLD	0x00000002
NX_UNBOLD	0x00000004
NX_NONSTANDARDCHARSET	0x00000008
NX_NARROW	0x00000010
NX_EXPANDED	0x00000020
NX_CONDENSED	0x00000040
NX_SMALLCAPS	0x00000080
NX_POSTER	0x00000100
NX_COMPRESSED	0x00000200

FontPanel View Tags

NX_FPPREVIEWFIELD
NX_FPSIZEFIELD
NX_FPVERTBUTTON
NX_FPPREVIEWBUTTON
NX_FPSETBUTTON
NX_FPSIZETITLE
NX_FPCURRENTFIELD

Gray Shades

	Value
NX_WHITE	1.0
NX_LTGRAY	2.0/3.0
NX_DKGRAY	1.0/3.0
NX_BLACK	0.0

Icon and Token Window Dimensions

	Value
NX_ICONWIDTH	48.0
NX_ICONHEIGHT	48.0
NX_TOKENWIDTH	64.0
NX_TOKENHEIGHT	64.0

Image Representation Device Matching Constant

NX_MATCHESDEVICE

Journaling Flags

Value

NX_JOURNALFLAG	31
NX_JOURNALFLAGMASK	(1 << NX_JOURNALFLAG)

Journaling Listener Name

Value

NX_JOURNALREQUEST	"NXJournalerRequest"
-------------------	----------------------

Journaling Recording Device

Value

NX_CODEC	0
NX_DSP	1

Journaling Status

Value

NX_STOPPED	0
NX_PLAYING	1
NX_RECORDING	2
NX_NONABORTABLEFLAG	31
NX_NONABORTABLEMASK	(1 << NX_NONABORTABLEFLAG)

Journaling Subevents

Value

NX_WINDRAGGED	0
NX_MOUSELOCATION	1
NX_LASTJRNEVENT	2

Journaling Window Encodings

Value

NX_KEYWINDOW	(-1)
NX_MAINWINDOW	(-2)
NX_MAINMENU	(-3)
NX_MOUSEDOWNWINDOW	(-4)
NX_APPICONWINDOW	(-5)
NX_UNKNOWNWINDOW	(-6)

Listener Maximum Message Size	Value
NX_MAXMESSAGE	(2048–sizeof(msg_header_t)– sizeof(msg_type_t)–sizeof(int)– sizeof(msg_type_t)–8)

Listener Maximum Parameters	Value
NX_MAXMSGPARAMS	20

Listener Position Types	Value
NX_TEXTPOSTYPE	0
NX_REGEXPRPOSTYPE	1
NX_LINENUMPOSTYPE	2
NX_CHARNUMPOSTYPE	3
NX_APPPOSTYPE	4

Listener Reserved Message Numbers	Value
NX_SELECTORPMSG	35555
NX_SELECTORFMSG	35556
NX_RESPONSEMSG	35557
NX_ACKNOWLEDGE	35558

Listener RPC Error Return Values Error	Value
NX_INCORRECTMESSAGE	–20000

Listener Timeout Default	Value
NX_SENDTIMEOUT	10000
NX_RCVTIMEOUT	10000

Mach Executable File Segment Names for Images	Segment Name
NX_EPSSEGMENT	“__EPS”
NX_TIFFSEGMENT	“__TIFF”
NX_ICONSEGMENT	“__ICON”

Matrix Selection Mode Constants

NX_RADIOMODE
NX_HIGHLIGHTMODE
NX_LISTMODE
NX_TRACKMODE

Modal Session

Return Constants	Value
NX_RUNSTOPPED	(-1000)
NX_RUNABORTED	(-1001)
NX_RUNCONTINUES	(-1002)

Obsolete Speaker Constants

NX_ISFILE,
NX_ISDIRECTORY,
NX_ISAPPLICATION,
NX_ISODMOUNT,
NX_ISNETMOUNT,
NX_ISSCSIMOUNT,
NX_ISFLOPPYMOUNT

Open Panel Tag Constants

Constants	Value
NX_OPICONBUTTON	NX_SPICONBUTTON
NX_OPTITLEFIELD	NX_SPTITLEFIELD
NX OPCANCELBUTTON	NX_SPCANCELBUTTON
NX_OPOKBUTTON	NX_SPOKBUTTON
NX_OPFORM	NX_SPFORM

Page Layout Panel Button Tags

NX_PLICONBUTTON
NX_PLTITLEFIELD
NX_PLPAPERSIZEBUTTON
NX_PLAYOUTBUTTON
NX_PLUNITSBUTTON
NX_PLWIDTHFORM
NX_PLHEIGHTFORM
NX_PLPORTLANDMATRIX
NX_PLSCALEFIELD
NX_PLCANCELBUTTON
NX_PLOKBUTTON

Page Order Modes

NX_DESCENDINGORDER
NX_SPECIALORDER
NX_ASCENDINGORDER
NX_UNKNOWNORDER

Page Orientation Constants

NX_PORTRAIT
NX_LANDSCAPE

Pagination Modes

NX_AUTOPAGINATION
NX_FITPAGINATION
NX_CLIPPAGINATION

Panel Button Tags	Value
--------------------------	--------------

NX_OKTAG	1
NX_CANCELTAG	0

Panel Return Constants	Value
-------------------------------	--------------

NX_ALERTDEFAULT	1
NX_ALERTALTERNATE	0
NX_ALERTOTHER	-1
NX_ALERTERROR	-2

Printer Table Key Length

NX_PRINTKEYMAXLEN

Printer Table States

NX_PRINTERTABLEOK
NX_PRINTERTABLENOTFOUND
NX_PRINTERTABLEERROR

Rectangle Sides

NX_XMIN
NX_YMIN
NX_XMAX
NX_YMAX

Save Panel Tag Constants

Value

NX_SPICONBUTTON	150
NX_SPTITLEFIELD	151
NX_SPBROWSER	152
NX_SPCANCELBUTTON	NX_CANCELTAG
NX_SPOKBUTTON	NX_OKTAG
NX_SPFORM	155

Scroller Arrow Positions

Value

NX_SCROLLARROWSMAXEND	0
NX_SCROLLARROWSMINEND	1
NX_SCROLLARROWSNONE	2

Scroller Part Identification Constants

Value

NX_NOPART	0
NX_DECPAGE	1
NX_KNOB	2
NX_INCPAGE	3
NX_DECLINE	4
NX_INCLINE	5
NX_KNOBSLOT	6
NX_JUMP	6

Scroller Usable Parts

Value

NX_SCROLLERNOPARTS	0
NX_SCROLLERONLYARROWS	1
NX_SCROLLERALLPARTS	2

Scroller Width and Height

NX_SCROLLERWIDTH	(18.0)
------------------	--------

Text Alignment Modes

NX_LEFTALIGNED
NX_RIGHTALIGNED
NX_CENTERED
NX_JUSTIFIED

Text Block Constant

NX_TEXTPER

Text Key Constants

NX_BACKSPACE
NX_CR
NX_DELETE
NX_BTAB
NX_ILLEGAL
NX_RETURN
NX_TAB
NX_BACKTAB
NX_LEFT
NX_RIGHT
NX_UP
NX_DOWN

Text Tab Stop Constant

NX_LEFTTAB

TIFF Compression Schemes

NX_TIFF_COMPRESSION_NONE
NX_TIFF_COMPRESSION_CCITTFAX3
NX_TIFF_COMPRESSION_CCITTFAX4
NX_TIFF_COMPRESSION_LZW
NX_TIFF_COMPRESSION_JPEG
NX_TIFF_COMPRESSION_PACKBITS

View Autoresize Constants

NX_NOTSIZABLE
NX_MINXMARGINSIZABLE
NX_WIDTHSIZABLE
NX_MAXXMARGINSIZABLE
NX_MINYMARGINSIZABLE
NX_HEIGHTSIZABLE
NX_MAXYMARGINSIZABLE

Window Button Masks

NX_CLOSEBUTTONMASK
NX_MINIATURIZEBUTTONMASK

Window Frame Description String Length

NX_MAXFRAMESTRINGLENGTH

Window Styles

NX_PLAINSTYLE
NX_TITLEDSTYLE
NX_MENUSTYLE
NX_MINIWINDOWSTYLE
NX_MINIWORLDSTYLE
NX_TOKENSTYLE
NX_RESIZEBARSTYLE
NX_FIRSTWINSTYLE
NX_LASTWINSTYLE
NX_NUMWINSTYLES

Window Tiers

Value

NX_NORMALLEVEL	0
NX_FLOATINGLEVEL	3
NX_DOCKLEVEL	5
NX_SUBMENULEVEL	10
NX_MAINMENULEVEL	20

Workspace Request Constants (File Operations)

	Value
WSM_MOVE_OPERATION	"move"
WSM_COPY_OPERATION	"copy"
WSM_LINK_OPERATION	"link"
WSM_COMPRESS_OPERATION	"compress"
WSM_DECOMPRESS_OPERATION	"decompress"
WSM_ENCRYPT_OPERATION	"encrypt"
WSM_DECRYPT_OPERATION	"decrypt"
WSM_DESTROY_OPERATION	"destroy"
WSM_RECYCLE_OPERATION	"recycle"
WSM_DUPLICATE_OPERATION	"duplicate"

Global Variables

Application Object

id NXApp;

Break Tables

```
const NXFSM *const NXEnglishBreakTable
const int NXEnglishBreakTableSize
const NXFSM *const NXEnglishNoBreakTable
const int NXEnglishNoBreakTableSize
const NXFSM *const NXCBreakTable
const int NXCBreakTableSize
```

Character Category Tables

```
const unsigned char *const NXEnglishCharCatTable
const unsigned char *const NXCCCharCatTable
```

Click Tables

```
const NXFSM *const NXEnglishClickTable
const int NXEnglishClickTableSize
const NXFSM *const NXCClickTable
const int NXCClickTableSize
```


Domain Name

const char *const NXSystemDomainName;

File Information

NXAtom NXPlainFileType;
NXAtom NXDirectoryFileType;
NXAtom NXApplicationFileType;
NXAtom NXFilesystemFileType;
NXAtom NXShellCommandFileType;

Filename Extension for DataLinks

NXAtom NXDataLinkFilenameExtension;

Null Object

int NXNullObject;

Pasteboard Names

NXAtom NXGeneralPboard;
NXAtom NXFontPboard;
NXAtom NXRulerPboard;
NXAtom NXFindPboard;
NXAtom NXDragPboard;

Pasteboard Types

NXAtom NXAsciiPboardType;
NXAtom NXPostScriptPboardType;
NXAtom NXTIFFPboardType;
NXAtom NXRTFPboardType;
NXAtom NXFilenamePboardType;
NXAtom NXTabularTextPboardType;
NXAtom NXFontPboardType;
NXAtom NXRulerPboardType;
NXAtom NXFileContentsPboardType;
NXAtom NXColorPboardType;

Pasteboard Types

```
NXAtom NXDataLinkPboardType;
```

Pasteboard Types

```
NXAtom NXSelectionPboardType;
```

Process

```
int NXProcessID;
```

Screen Dump Switch

```
BOOL NXScreenDump
```

Smart Cut and Paste Tables

```
const unsigned char *const NXEnglishSmartLeftChars  
const unsigned char *const NXEnglishSmartRightChars  
const unsigned char *const NXCSmartLeftChars  
const unsigned char *const NXCSmartRightChars
```

View Drawing Status

```
short NXDrawingStatus;
```

Workspace Name

```
const char *NXWorkspaceName;  
const char *const NXWorkspaceReplyName;  
#define NX_WORKSPACEREQUEST NXWorkspaceName  
#define NX_WORKSPACEREPLY NXWorkspaceReplyName
```

3 *Common Classes and Functions*

Classes

HashTable

Inherits From: Object

Initializing and Freeing a HashTable

- **init** Initializes a new, default HashTable
- **initKeyDesc:(const char *)aKeyDesc** Initializes a new HashTable
- **initKeyDesc:(const char *)aKeyDesc
valueDesc:(const char *)aValueDesc** Initializes a new HashTable
- **initKeyDesc:(const char *)aKeyDesc
valueDesc:(const char *)aValueDesc
capacity:(unsigned int)aCapacity** Initializes a new HashTable
- **free** Deallocates the HashTable
- **freeObjects** Deallocates the HashTable's objects
- **freeKeys:(void (*)(void *))keyFunc
values:(void (*)(void *))valueFunc** Conditionally frees the HashTable's associations
- **empty** Empties the HashTable but retains its capacity

Copying a HashTable

- **copyFromZone:(NXZone *) zone** Returns an empty copy of the HashTable

Manipulating Table Associations

- (unsigned int)**count** Returns the number of objects in the table
- (BOOL)**isKey**:(const void *)*aKey* Indicates whether *aKey* is in the table
- (void *)**valueForKey**:(const void *)*aKey* Returns the value mapped to *aKey*
- (void *)**insertKey**:(const void *)*aKey*
value:(void *)*aValue* Adds or updates *aKey/aValue* pair
- (void *)**removeKey**:(const void *)*aKey* Removes *aKey/aValue* pair

Iterating Over All Associations

- (NXHashState)**initState** Begins process of iteration through the HashTable
- (BOOL)**nextState**:(NXHashState *)*aState*
key:(const void **)*aKey*
value:(void **)*aValue* Moves to the next entry in the HashTable

Archiving

- **read**:(NXTypedStream *)*stream* Read the HashTable from the typed stream *stream*
- **write**:(NXTypedStream *)*stream* Writes the HashTable to the typed stream *stream*

List

Inherits From: Object

Initializing a New List Object

- **init** Initializes the new List object
- **initCount**:(unsigned int)*numSlots* Initializes the new List to hold at least *numSlots* objects

Copying and Freeing a List

- **copyFromZone**:(NXZone *)*zone* Returns a copy of the List allocated from *zone*
- **free** Deallocates the List object

Manipulating Objects by Index

- **insertObject**:*anObject* **at**:(unsigned int)*index* Puts *anObject* in the List at *index*

- **addObject:***anObject*
 - **removeObjectAt:**(unsigned int)*index*
 - **removeLastObject**
 - **replaceObjectAt:**(unsigned int)*index*
with:*newObject*
 - **objectAt:**(unsigned int)*index*
 - **lastObject**
 - (unsigned int)**count**
- Adds *anObject* at the end of the List
- Removes the object located at *index*
- Removes the object at the end of the List
- Puts *newObject* in place of the object at *index*
- Returns the object at *index*
- Returns the object at the end of the List
- Returns the number of objects in the List

Manipulating Objects by id

- **addObject:***anObject*
 - **addObjectIfAbsent:***anObject*
 - **removeObject:***anObject*
 - **replaceObject:***anObject* with:*newObject*
 - (unsigned int)**indexOf:***anObject*
- Adds *anObject* at the end of the List
- Adds *anObject* to the List, if it's not already in the List
- Removes first occurrence of *anObject* from the List
- Puts *newObject* in the List in place of *anObject*
- Returns the index of *anObject*

Comparing and Combining Lists

- (BOOL)**isEqual:***anObject*
 - **appendList:**(List *)*otherList*
- Returns whether the two Lists have the same contents
- Adds the objects in *otherList* to the receiving List

Emptying a List

- **empty**
 - **freeObjects**
- Empties the List of its contents, but doesn't free the objects
- Deallocates all the objects in the List

Sending Messages to the Objects

- **makeObjectsPerform:**(SEL)*aSelector*
 - **makeObjectsPerform:**(SEL)*aSelector*
with:*anObject*
- Sends an *aSelector* message to each object in the List
- Sends *aSelector* message with an argument to each object

Managing the Storage Capacity

- (unsigned int)**capacity**
 - **setAvailableCapacity:**(unsigned int)*numSlots*
- Returns the number of objects the List can store
- Sets the capacity of the List to at least *numSlots* objects

Archiving

- **write:**(NXTypedStream *)*stream*
 - **read:**(NXTypedStream *)*stream*
- Writes the List to the typed stream *stream*
- Reads the List from the typed stream *stream*

NXBundle

Inherits From: Object

Initializing a New NXBundle object

– **initWithDirectory:**(const char *)*fullPath* Initializes a new object for the *fullPath* directory

Getting and Freeing an NXBundle

+ **mainBundle** Returns the NXBundle for the directory of the executable
+ **bundleForClass:***classObject* Returns the NXBundle that loaded *classObject*
– **free** Frees the receiving NXBundle, if it can be freed

Getting a Bundled Class

– **principalClass** Returns the principal class loaded by the receiver
– **classNamed:**(const char *)*classname* Returns the class object for the *classname* class

Setting Which Resources To Use

+ **setSystemLanguages:**(const char * const *)*languageArray*
Informs the receiver of the user’s language preferences

Finding a Resource

– (BOOL)**getPath:**(char *)*path*
 forResource:(const char *)*filename*
 ofType:(const char *)*extension* Provides the full path to the *filename* resource
+ (BOOL)**getPath:**(char *)*path*
 forResource:(const char *)*filename*
 ofType:(const char *)*extension*
 inDirectory:(const char *)*directory*
 withVersion:(int)*version* Provides the full path to the *filename* resource

Getting the Bundle Directory

– (const char *)**directory** Returns the full pathname of the NXBundle’s directory

Setting the Version

- **setVersion:**(int)*version* Sets the version that resources must match
- (int)*version* Returns the version that resources must match

NXStringTable

Inherits From: HashTable : Object

Initializing and Freeing an NXStringTable

- **init** Initializes a new NXStringTable
- **free** Deallocates the NXStringTable

Querying an NXStringTable

- (const char *)**valueForKey:**(const char *)*aString* Returns the value that corresponds to *aString*

Writing Elements

- **readFromFile:**(const char *)*fileName* Reads the keys and values from *fileName*
- **writeToFile:**(const char *)*fileName* Writes the keys and values to *fileName*
- **readFromStream:**(NXStream *)*stream* Reads the keys and values from *stream*
- **writeToStream:**(NXStream *)*stream* Writes the keys and values to *stream*

Storage

Inherits From: Object

Initializing a New Storage Object

- **init** Initializes the Storage object

- **initCount:**(unsigned int)*count*
- **elementSize:**(unsigned int)*sizeInBytes*
- **description:**(const char *)*descriptor*

Initializes the new object to store at least *count* elements

Copying and Freeing Storage Objects

- **copyFromZone:**(NXZone *)*zone*
- **free**

Returns a copy of the Storage object allocated from *zone*
Deallocates the Storage object and its contents

Getting, Adding, and Removing Elements

- **addElement:**(void *)*anElement*
- **insertElement:**(void *)*anElement*
– **at:**(unsigned int)*index*
- **removeElementAt:**(unsigned int)*index*
- **removeLastElement**
- **replaceElementAt:**(unsigned int)*index*
– **with:**(void *)*anElement*
- **empty**
- (void *)**elementAt:**(unsigned int)*index*

Adds *anElement* at the end of the Storage array

Puts *anElement* in the Storage array at *index*

Removes the element located at *index*

Removes the last element

Replaces the element at *index* with *anElement*

Empties the Storage object but retains its capacity

Returns a pointer to the element at *index*

Comparing Storage Objects

- (BOOL)**isEqual:***anObject*

Returns whether two Storage objects are the same

Managing the Storage Capacity and Type

- (unsigned int)**count**
- (const char *)**description**
- **setAvailableCapacity:**(unsigned int)*numSlots*
- **setNumSlots:**(unsigned int)*numSlots*

Returns the number of elements currently stored

Returns the encoding for the type of elements stored

Sets the capacity of the Storage array to at least *numSlots*

Sets the number of elements stored to *numSlots* elements

Archiving

- **read:**(NXTypedStream *)*stream*
- **write:**(NXTypedStream *)*stream*

Reads the Storage object from the typed stream *stream*

Writes the Storage object to the typed stream *stream*

Functions

Character Classification Functions

Classify NeXTSTEP-Encoded Values:

int	<code>NXIsAlpha(unsigned int c)</code>
int	<code>NXIsUpper(unsigned int c)</code>
int	<code>NXIsLower(unsigned int c)</code>
int	<code>NXIsDigit(unsigned int c)</code>
int	<code>NXIsXDigit(unsigned int c)</code>
int	<code>NXIsAlNum(unsigned int c)</code>
int	<code>NXIsSpace(unsigned int c)</code>
int	<code>NXIsPunct(unsigned int c)</code>
int	<code>NXIsPrint(unsigned int c)</code>
int	<code>NXIsGraph(unsigned int c)</code>
int	<code>NXIsCntrl(unsigned int c)</code>
int	<code>NXIsAscii(unsigned int c)</code>

Convert NeXTSTEP-Encoded Characters:

unsigned char *	<code>NXToAscii(unsigned int c)</code>
int	<code>NXToLower(unsigned int c)</code>
int	<code>NXToUpper(unsigned int c)</code>

Defaults System Functions

Set or Read Default Parameters:

int	<code>NXRegisterDefaults(const char *owner, const NXDefaultsVector vector)</code>
const char *	<code>NXGetDefaultValue(const char *owner, const char *name)</code>
const char *	<code>NXReadDefault(const char *owner, const char *name)</code>
int	<code>NXRemoveDefault(const char *owner, const char *name)</code>
int	<code>NXSetDefault(const char *owner, const char *name, const char *value)</code>
const char *	<code>NXUpdateDefault(const char *owner, const char *name)</code>
void	<code>NXUpdateDefaults(void)</code>
int	<code>NXWriteDefault(const char *owner, const char *name, const char *value)</code>
int	<code>NXWriteDefaults(const char *owner, NXDefaultsVector vector)</code>
const char *	<code>NXSetDefaultsUser(const char *newUser)</code>

Error-Handling Functions

Macros to Raise an Exception:

```
void          NX_RAISE(int code, const void *data1, const void *data2)
void          NX_RERAISE(void)
val          NX_VALRETURN(val)
void          NX_VOIDRETURN
```

Set and Return an Exception Raiser:

```
void          NXDefaultExceptionRaiser(int code, const void *data1, const void *data2)
void          NXSetExceptionRaiser(NXExceptionRaiser *procedure)
NXExceptionRaiser * NXGetExceptionRaiser(void)
```

Macros to Handle Uncaught Exceptions:

```
void          NXSetUncaughtExceptionHandler(NXUncaughtExceptionHandler *proc)
NXUncaughtExceptionHandler * NXGetUncaughtExceptionHandler(void)
```

Manage the Error Data Buffer:

```
void          NXAllocErrorData(int size, void **data)
void          NXResetErrorData(void)
```

Stream Functions

Manipulate a Memory Stream:

```
NXStream *   NXOpenMemory(const char *address, int size, int mode)
NXStream *   NXMapFile(const char *pathName, int mode)
int          NXSaveToFile(NXStream *stream, const char *name)
void         NXCloseMemory(NXStream *stream, int option)
void         NXGetMemoryBuffer(NXStream *stream, char **streambuf, int *len,
                               int *maxLen)
```

Open a File Stream or a Mach Port Stream:

```
NXStream *   NXOpenFile(int fd, int mode)
NXStream *   NXOpenPort(port_t port, int mode)
```

Close a Stream:

void NXClose(NXStream *stream)

Read From or Write to a Stream:

int NXRead(NXStream *stream, void *buf, int count)
int NXWrite(NXStream *stream, const void *buf, int count)

Read or Write Formatted Data from or to a Stream:

int NXPutc(NXStream *stream, char c) /* a macro */
int NXGetc(NXStream *stream) /* a macro */
void NXUngetc(NXStream *stream)
int NXScanf(NXStream *stream, const char *format, ...)
void NXPrintf(NXStream *stream, const char *format, ...)
int NXVScanf(NXStream *stream, const char *format, va_list argList)
void NXVPrintf(NXStream *stream, const char *format, va_list argList)

Register a Procedure for Formatting Data Written to a Stream:

void NXRegisterPrintfProc(char formatChar, NXPrintfProc *proc, void *procData)

Flush a Stream:

int NXFlush(NXStream *stream)

Set or Report Current Position in a Stream:

void NXSeek(NXStream *stream, long offset, int ptrName)
long NXTell(NXStream *stream)
BOOL NXAtEOS(NXStream *stream) /* a macro */

Support a User-defined Stream:

NXStream * NXStreamCreate(int mode, int createBuf)
NXStream * NXStreamCreateFromZone(int mode, int createBuf, NXZone *zone)
void NXStreamDestroy(NXStream *stream)
int NXDefaultRead(NXStream *stream, void *buf, int count)
int NXDefaultWrite(NXStream *stream, const void *buf, int count)
int NXFill(NXStream *stream)
void NXChangeBuffer(NXStream *stream)

Typed Stream Functions

Open or Close a Typed Stream:

```
NXTypedStream* NXOpenTypedStream(NXStream *stream, int mode)
void           NXCloseTypedStream(NXTypedStream *stream)
NXTypedStream* NXOpenTypedStreamForFile(const char *fileName, int mode)
```

Read or Write Objective C Objects from or to a Typed Stream:

```
id           NXReadObject(NXTypedStream *stream)
void        NXWriteObject(NXTypedStream *stream, id object)
void        NXWriteObjectReference(NXTypedStream *stream, id object)
void        NXWriteRootObject(NXTypedStream *stream, id rootObject)
```

Read or Write Arbitrary Data from or to a Typed Stream:

```
void        NXReadType(NXTypedStream *stream, const char *type, void *data)
void        NXWriteType(NXTypedStream *stream, const char *type, const void *data)
void        NXReadTypes(NXTypedStream *stream, const char *types, ...)
void        NXWriteTypes(NXTypedStream *stream, const char *types, ...)
```

Read or Write Arrays from or to a Typed Stream:

```
void        NXReadArray(NXTypedStream *stream, const char *dataType, int count,
                        void *data)
void        NXWriteArray(NXTypedStream *stream, const char *dataType, int count,
                        const void *data)
```

Read or Write an Object from or to a Typed-Stream Memory Buffer:

```
id           NXReadObjectFromBuffer(const char *buffer, int length)
char *       NXWriteRootObjectToBuffer(id object, int *length)
void        NXFreeObjectBuffer(char *buffer, int length)
```

Determine Whether There's More Data to Be Read:

```
BOOL        NXEndOfTypedStream(NXTypedStream *stream)
```

Flush a Typed Stream

void NXFlushTypedStream(NXTypedStream *stream)

Get the Version Number of a Class

int NXTypedStreamClassVersion(NXTypedStream *stream, const char *className)

Get or Set the Zone for a Typed Stream

NXZone NXGetTypedStreamZone(NXTypedStream *stream)
void NXSetTypedStreamZone(NXTypedStream *stream, NXZone *zone)
id NXReadObjectFromBufferWithZone(const char *buffer, int length,
 NXZone *zone)

Memory Allocation Functions

Zone Memory Allocation

void * NXZoneMalloc(NXZone *zone, size_t size)
void * NXZoneCalloc(NXZone *zone, size_t numElems, size_t numBytes)
void * NXZoneRealloc(NXZone *zone, void *ptr, size_t size)
void NXZoneFree(NXZone *zone, void *ptr)
NXZone * NXDefaultMallocZone(void)
NXZone * NXCreateZone(size_t startSize, size_t granularity, int canFree)
NXZone * NXCreateChildZone(NXZone *parentZone, size_t startSize, size_t granularity,
 int canFree)
void NXMergeZone(NXZone *zone)
void NXDestroyZone(NXZone *zone)
NXZone * NXZoneFromPtr(void *ptr)
void NXZonePtrInfo(void *ptr)
int NXMallocCheck(void)
void NXNameZone(NXZone *zonep, const char *name)

Hash and String Table Functions

Create, Manipulate, and Free a Hash Table:

```
NXHashTable * NXCreateHashTable(NXHashTablePrototype prototype, unsigned capacity,
                                const void *info)
NXHashTable * NXCreateHashTableFromZone(NXHashTablePrototype prototype,
                                         unsigned capacity, const void *info, NXZone *zone)
void NXFreeHashTable(NXHashTable *table)
void NXEmptyHashTable(NXHashTable *table)
void NXResetHashTable(NXHashTable *table)
NXHashTable * NXCopyHashTable(NXHashTable *table)
BOOL NXCompareHashTables(NXHashTable *table1, NXHashTable *table2)
unsigned NXPtrHash(const void *info, const void *data)
unsigned NXStrHash(const void *info, const void *data)
int NXPtrIsEqual(const void *info, const void *data1, const void *data2)
int NXStrIsEqual(const void *info, const void *data1, const void *data2)
void NXNoEffectFree(const void *info, void *data)
void NXReallyFree(const void *info, void *data)
```

Manipulate the Elements of a Hash Table:

```
void * NXHashInsert(NXHashTable *table, const void *data)
void * NXHashInsertIfAbsent(NXHashTable *table, const void *data)
int NXHashMember(NXHashTable *table, const void *data)
void * NXHashGet(NXHashTable *table, const void *data)
void * NXHashRemove(NXHashTable *table, const void *data)
unsigned NXCountHashTable(NXHashTable *table)
NXHashState NXInitHashState(NXHashTable *table)
int NXNextHashState(NXHashTable *table, NXHashState *state, void **data)
```

String Functions

Get Localized Versions of Strings:

```
const char *    NXLocalizedString(const char *key, const char *value, comment)
const char *    NXLocalizedStringFromTable(const char *table, const char *key,
                                           const char *value, comment)
const char *    NXLocalizedStringFromTableInBundle(const char *table, NXBundle *bundle,
                                                   const char *key, const char *value, comment)
const char *    NXLoadLocalizedStringFromTableInBundle(const char *table,
                                                       NXBundle *bundle, const char *key, const char *value)
```

Create a Unique String:

```
NXAtom         NXUniqueString(const char *buffer)
NXAtom         NXUniqueStringWithLength(const char *buffer, int length)
NXAtom         NXUniqueStringNoCopy(const char *buffer)
char *         NXCopyStringBuffer(const char *buffer)
char *         NXCopyStringBufferFromZone(const char *buffer, NXZone *zone)
```

Miscellaneous Functions

Get a Pointer to the Objects Stored in a List:

```
id *           NX_ADDRESS(List *aList)
```

Search for and Read a File:

```
int            NXFilePathSearch(const char *envVarName, const char *defaultPath,
                                int leftToRight, const char *fileName, int (*funcPtr)(),
                                void *funcArg)
```


Types and Constants

Defined Types

NXAtom

```
typedef const char *NXAtom;
```

NXExceptionRaiser

```
typedef void NXExceptionRaiser(int code,  
    const void *data1,  
    const void *data2);
```

NXHandler

```
typedef struct _NXHandler {  
    jmp_buf jumpState;  
    struct _NXHandler *next;  
    int code;  
    const void *data1, *data2;  
} NXHandler;
```

NXHashState

```
typedef struct {  
    int i;  
    int j;  
} NXHashState;
```

NXHashTable

```
typedef struct {  
    const NXHashTablePrototype *prototype;  
    unsigned count;  
    unsigned nbBuckets;  
    void *buckets;  
    const void *info;  
} NXHashTable;
```

NXHashTablePrototype

```
typedef struct {
    unsigned (*hash)(const void *info, const void *data);
    int (*isEqual)(const void *info, const void *data1, const void *data2);
    void (*free)(const void *info, void *data);
    int style;
} NXHashTablePrototype;
```

NXUncaughtExceptionHandler

```
typedef void NXUncaughtExceptionHandler(int code,
    const void *data1,
    const void *data2);
```

NXZone

```
typedef struct _NXZone {
    void *(*realloc)(struct _NXZone *zonep, void *ptr, size_t size);
    void *(*malloc)(struct _NXZone *zonep, size_t size);
    void (*free)(struct _NXZone *zonep, void *ptr);
    void (*destroy)(struct _NXZone *zonep);
} NXZone;
```

Symbolic Constants

List Constants

`NX_NOT_IN_LIST`

NXStringTable Constants	Value
--------------------------------	--------------

<code>MAX_NXSTRINGTABLE_LENGTH</code>	1024
---------------------------------------	------

Zone Constants	Value
-----------------------	--------------

<code>NX_NOZONE</code>	(NXZone *)0
------------------------	-------------

Global Variables

HashTable Prototypes

```
const NXHashTablePrototype NXPtrPrototype;  
const NXHashTablePrototype NXStrPrototype;  
const NXHashTablePrototype NXPtrStructKeyPrototype;  
const NXHashTablePrototype NXStrStructKeyPrototype;
```

4 Database Kit

Classes

DBAssociation

Inherits From: Object

Initializing

- **initWithGroup:***aFetchGroup*
expression:*anExpr*
destination:*aDest* Initializes; associates *aFetchGroup* with *aDest*

Linking Expression and View

- **destination** The association's user interface object
- **fetchGroup** The DBFetchGroup that owns the association
- **expression** The DBExpression that selects the properties displayed
- **setDestination:***newDestination* Sets the user association's user interface object

Methods to be Re-Defined in a Subclass of Association

- **contentsDidChange** Notice to redisplay because the value changed
- **currentRecordDidDelete** Notice to redisplay because a record was deleted

- **endEditing** Notification that editing the destination must end
- **getValue:value** Gets an object containing the association's data
- **(unsigned int)selectedRowAfter:(unsigned int)previousRow**
The index of the next selected row
- **selectionDidChange** Notice that the user has changed the selection
- **setValue:value** Sets the association's data
- **validateEditing** Notice to validate changes the user has made

Methods to be Defined in the Destination of a Custom Association

- **association:association** Gets an object containing the association's data
getValue:(DBValue *)value
- **association:association** Sets the association's data
setValue:(DBValue *)value
- **associationContentsDidChange:association** Notice to redisplay because the value changed
- **associationCurrentRecordDidDelete:association** Notice to redisplay because a record was deleted
- **associationSelectionDidChange:association** Notice that the user has changed the selection

DBBinder

Inherits From: Object

Conforms To: DBCursorPositioning

Initializing

- **init** Initializes a new DBBinder instance
- **initWithDatabase:aDBDatabase** Initializes database, properties, qualifier; frees DBBinder
withProperties:(List *)propertyList
andQualifier:(DBQualifier *)aDBQualifier
- **free** Frees the space allocated to a DBBinder

Connecting to a Database

- **(DBDatabase *)database** The DBBinder's DBDatabase
- **setDatabase:(DBDatabase *)aDatabase** Sets the DBBinder's DBDatabase

Managing Properties

- (List *)**getProperties**:(List *)*aList* Gets and returns the DBBinder’s properties
- (List *)**setProperty**s:(List *)*aList* Sets and returns the DBBinder’s properties
- **addProperty**:*anObject* Adds an object to the list of properties
- **removePropertyAt**:(unsigned int)*index* Deletes one of the objects from the list of properties

Managing the Qualifier

- (DBQualifier *)**qualifier** The DBBinder’s qualifier
- **setQualifier**:(DBQualifier *)*aQualifier* Sets the DBBinder’s qualifier

Managing the Container

- (id <DBContainers>)**container** The DBBinder’s container
- **setContainer**:(id <DBContainers>)*anObject* Sets the DBBinder’s container
- **setFlushEnabled**:(BOOL)*flag* Sets whether flushing the DBBinder is permitted
- (BOOL)**isFlushEnabled** Reports whether flushing is enabled; default YES
- **setFreeObjectsOnFlush**:(BOOL)*flag* Sets whether the DBBinder is freed when flushed
- (BOOL)**areObjectsFreedOnFlush** YES if container objects freed when DBBinder is flushed

Managing the Record Prototype

- + **setDynamicRecordClassName**:(const char *)*aName* Assign a unique name to a class for unprototyped records
- + **setDynamicRecordSuperclassName**:(const char *)*aName* Identify (existing) superclass for unprototyped records
- **setRecordPrototype**:*anObject* Makes *anObject* the prototype for the DBBinder’s records
- **createRecordPrototype** Create default prototype object for the DBBinder’s records
- (BOOL)**ownsRecordPrototype** YES if **createRecordPrototype** will work (no prototype)
- **recordPrototype** The DBBinder’s record prototype
- **associateRecordIvar**:(const char *)*variableName* Makes *variableName* report the value of *aProperty*
 withProperty:(id <DBProperties>)*aProperty*
- **associateRecordSelectors**:(SEL)*set* Sets the selectors for storing and retrieving *aProperty*
 :(SEL)*get*
 withProperty:(id <DBProperties>)*aProperty*
- (DBValue *)**valueForProperty**:(id <DBProperties>)*aProperty*
 The value of aProperty for the current record

Ordering and Ignoring Records

- **addRetrieveOrder:**(DBRetrieveOrder)*anOrder* **for:**(id <DBProperties>)*aProperty*
Appends *aProperty* to the retrieve ordering criteria
- **removeRetrieveOrderFor:**(id <DBProperties>)*aProperty*
Removes *aProperty* from the list of ordering criteria
- (DBRetrieveOrder)**retrieveOrderFor:**(id <DBProperties>)*aProperty*
The direction in which *aProperty* is sorted on retrieve
- (unsigned int)**positionInOrderingsFor:**(id <DBProperties>)*aProperty*
The rank order of *aProperty* in the list of order criteria
- (BOOL)**ignoresDuplicateResults**
YES if duplicate records are ignored during select
- **setIgnoresDuplicateResults:**(BOOL)*flag*
Sets whether duplicate records will be ignored in select

Accessing the Database

- **fetch**
Fetches record; puts in record objects (in the container)
- **select**
Selects records and fetches them
- **selectWithoutFetching**
Selects records in the database for fetching
- **insert**
Inserts the DBBinder's record objects into the database
- **update**
Updates the database for each record object
- **delete**
Deletes record object from the database
- (BOOL)**evaluateString:**(const unsigned char *)*aString*
Tells the adaptor to evaluate *aString* (without qualifier)
- (BOOL)**adaptorWillEvaluateString:**(const unsigned char *)*aString*
YES if delegate permits evaluation of *aString*; default YES

Fetching in a Thread

- **fetchInThread**
Starts an asynchronous fetch to the container
- **cancelFetch**
Aborts an asynchronous fetch
- **checkThreadedFetchCompletion:**(double)*timeout*
Sends **binderDidFetch:** if an asynchronous fetch completes within *timeout* seconds

Limiting a Fetch

- **setMaximumRecordsPerFetch:**(unsigned int)*limit*
Sets maximum records per synchronous fetch
- (unsigned int)**maximumRecordsPerFetch**
Returns maximum records per fetch; default unlimited
- (BOOL)**recordLimitReached**
YES if the previous fetch stopped for the record limit

Using the Shared Cursors for Several Binders

- **setSharesContext:(BOOL)flag** Set whether this binder uses the shared cursor
- (BOOL)**sharesContext** YES if this binder uses the shared cursor

Managing General Resources

- **reset** Cancels any fetch, then flushes and frees objects
- (BOOL)**flush** If enabled, empties the container
- (NXZone *)**scratchZone** The zone the DBBinder is now using for allocations

Appointing a Delegate

- **delegate** The object that receives notification messages
- **setDelegate:anObject** Sets the object to receive notification messages

Archiving

- **read:(NXTypedStream *)stream** Creates an instance by reading from a typed stream
- **write:(NXTypedStream *)stream** Archives an instance by writing to a typed stream

Methods Implemented by the Delegate

- **binder:aBinder didEvaluateString:(const unsigned char *)aString** Notification that *aString* was evaluated by the adaptor
- (BOOL)**binder:aBinder willEvaluateString:(const unsigned char *)aString** Notification that *aString* will be sent the adaptor; YES lets evaluation proceed
- **binderDidDelete:aBinder** Notification that records were deleted from the database
- **binderDidFetch:aBinder** Notification that records were fetched from the database
- **binderDidInsert:aBinder** Notification that records were inserted in the database
- **binderDidSelect:aBinder** Notification that records were selected (but not fetched)
- **binderDidUpdate:aBinder** Notification that database records were updated
- (BOOL)**binderWillDelete:aBinder** YES permits deleting binder's records from the database
- (BOOL)**binderWillFetch:aBinder** YES permits records to be fetched from the database
- (BOOL)**binderWillInsert:aBinder** YES permits records to be inserted in the database
- (BOOL)**binderWillSelect:aBinder** YES permits records to be selected in the database
- (BOOL)**binderWillUpdate:aBinder** YES permits records to update the database

DBDatabase

Inherits From: Object

Initializing the Class

+ **initialize** Sent automatically; prepares class to respond to inquiries

Reporting What's Available

+ (const char **) **adaptorNames** List of the names of available adaptors
+ (const char **) **databaseNamesForAdaptor:(const char *)anAdaptorName**
List of database available through *anAdaptorName*

Initializing an Instance

- **initWithFile:(const char *)aPath** Initializes and loads information from a model file

Describing the Model Source

- (const char *) **directory** The directory from which the model was loaded
- (const char *) **name** The model's name in the class's name table
- (BOOL) **setName:(const char *)aString** Sets the model's name in the class's name table
- (const char *) **currentAdaptorName** The name of the current database adaptor
- (const char *) **defaultAdaptorName** The name of the model's default adaptor
- (const unsigned char *) **defaultLoginString** The model's default login string
- (const unsigned char *) **currentLoginString** The current login string
- (const unsigned char *) **loginStringForUser:(const char *)aUser**
The the model's login string for user *aUser*

Describing the Database Model

- (id <DBEntities>) **entityNamed:(const char *)aName**
Returns an object embodying entity *aName*
- (List *) **getEntities:(List *)aList**
Returns a list of the names of the model's entities

Revising the Data Dictionary

- **emptyDataDictionary** Frees the current data dictionary
- **loadDefaultDataDictionary** Replaces the data dictionary by querying the database

Connecting to the Database

- + **findDatabaseNamed:(const char *)*aName***
connect:(BOOL)*flag* Returns a DBDatabase instance, after loading model *aName* and (and, if flag is YES, connecting to it)
- (BOOL)**connect** Opens a connection to database using the default login
- (BOOL)**connectUsingString:(const unsigned char *)*aString***
Opens database connection to *database* by sending *aString*
- (BOOL)**connectUsingAdaptor:(const char *)*aClassname***
andString:(const unsigned char *)*aString* Opens database connection via *anAdaptor* and *aString*
- (BOOL)**disconnect** Disconnects from the database
- (BOOL)**disconnectUsingString:(const unsigned char *)*aString***
Disconnects from the database by sending it *aString*
- (BOOL)**isConnected** YES if there is a valid connection to the database
- (const unsigned char *)**connectionName** The name assigned to the current connection

Managing Transactions

- (BOOL)**beginTransaction** YES if a new transaction is successfully started
- (BOOL)**rollbackTransaction** YES if a the current transaction is successfully rolled back
- (BOOL)**commitTransaction** YES if a the current transaction is successfully committed
- (BOOL)**isTransactionInProgress** YES is a transaction is in progress
- (BOOL)**areTransactionsEnabled** YES if transactions are enabled
- (BOOL)**enableTransactions:(BOOL)*flag*** Enable/disable transaction; returns YES if successful

Using a Delegate

- **delegate** The object that receives notification messages
- **setDelegate:*anObject*** Sets the object that receives notification messages

Evaluating an Arbitrary String

- (BOOL)**evaluateString:*aString*** Returns YES if the adaptor evaluates the string

Controlling the User Interface

- (BOOL)**arePanelsEnabled** YES if UI panels can respond to problems
- **setPanelsEnabled:(BOOL)*flag*** Enable/disable response by UI panels

Archiving

- **read:(NXTypedStream *)*stream*** Creates an instance by reading from a typed stream
- **write:(NXTypedStream *)*stream*** Archives an instance by writing to a typed stream

Methods Implemented in the Delegate

- **db:***aDatabase*
log:(const char*)*fmt*, ...
Notification of log message sent by *aDatabase*
- (BOOL)**db:***aDatabase*
notificationFrom:*anObject*
message:(const unsigned char*)*msg*
code:(int)*n*
Notification of a message received from *aDatabase*;
Returns YES when the user acknowledges the notification.
- (BOOL)**db:***aDatabase*
willEvaluateString:(const char *)*aString*
usingBinder:(const char *)*aBinder*
Notice that *aString* will be evaluated; YES lets it proceed
- **dbDidRollbackTransaction:***sender*
Notification that database rolled back a transaction
- **dbDidCommitTransaction:***sender*
Notification that database committed a transaction
- **dbWillCommitTransaction:***sender*
Notification that database will commit a transaction
- **dbWillRollbackTransaction:***sender*
Notification that database will roll back a transaction

DBEditableFormatter

Inherits From: DBFormatter : Object

Initializing

- **init**
Initializes a new instance
- **free**
Frees the space an instance formerly used

Manipulating Font

- **font**
Returns the font used in the editable display
- **setFont:***aFont*
Sets the font used in the editable display

Displaying and Editing

- **drawFieldAt:**(unsigned int) *row*
: *column*
inside:(NXRect *) *frame*
inView: *aView*
withAttributes:(id <DBTableVectors>) *rowAttrs*
:(id <DBTableVectors>) *columnAttrs*
usePositions:(BOOL) *useRowPos*
:(BOOL) *useColumnPos*
Displays one field of the data source’s current record taken from position *row* or *column* of the dynamic axis, using *rowAttrs* or *colAttrs* to identify static attributes, and flags *useRowPos* and *useColumnPos* to select which
- (BOOL) **editFieldAt:**(unsigned int) *row*
:(unsigned int) *column*
inside:(NXRect *) *frame*
inView: *aView*
withAttributes:(id <DBTableVectors>) *rowAttrs*
:(id <DBTableVectors>) *columnAttrs*
usePositions:(BOOL) *useRowPos*
:(BOOL) *useColumnPos*
onEvent: *theEvent*
Displays and prepares to edit one field of the data source’s current record, taken from *row* or *column* of dynamic axis, using *rowAttrs* or *colAttrs* to identify static attributes, and flags *useRowPos* and *useColumnPos* to select which; returns YES if editing was permitted

Controlling Editing

- **abortEditing**
Forces an end to editing and discards changes; returns self
- (BOOL) **endEditing**
Ends editing when user clicks elsewhere Returns YES if that becomes first responder

Archiving

- **read:**(NXTypedStream *) *stream*
Creates an instance by reading from a typed stream
- **write:**(NXTypedStream *) *stream*
Archives an instance by writing to a typed stream
- **finishUnarchiving**
Automatically invoked final step in unarchiving

Methods Implemented by the Delegate

See DBFormatterValidation protocol.

DBExpression

Inherits From: Object

Conforms To: DBExpressionValues
DBProperties

Creating and Freeing a DBExpression

- **initForEntity:**(id <DBEntities>)*anEntity* Initializes for *anEntity*, with description string shown
 fromDescription:(const unsigned char *)*descriptionFormat*, ...
- **initForEntity:**(id <DBEntities>)*anEntity* Initializes *anEntity*, from property *aName*,
 fromName:(const char *)*aName* to have data type *aType*
 usingType:(const char *)*aType*
- **copyFromZone:**(NXZone *)*zone* Returns new copy of receiver, allocated from *zone*
- **free** Frees the space that an instance formerly used

Setting the Entity and Description

- **setEntity:**(id <DBEntities>)*anEntity* Sets *anEntity*, with the description string shown
 andDescription:(const unsigned char *)*descriptionFormat*, ...

Archiving

- **read:**(NXTypedStream *)*stream* Creates an instance by reading from a typed stream
- **write:**(NXTypedStream *)*stream* Archives an instance by writing to a typed stream

DBFetchGroup

Inherits From: Object

Initializing

- **initEntity:***anEntity* Initialize a new instance for links to *anEntity*
- **setName:**(const char *)*aName* Invoked automatically; matches the name to the attribute it
 fetches

Reporting Current Context

- (const char *)**name** Returns the name (set to match the attribute it fetches)
- **module** The DBModule that owns the fetch group
- **entity** The DBEntity for which the fetch group fetches
- **recordList** The DBRecordList in which fetched records are stored
- (unsigned int)**currentRecord** The index within the DBRecordList of the current record
- (unsigned int)**recordCount** The number of records in the DBRecordList

Controlling Current Selection

- **setAutoselect:**(BOOL)*flag* If YES, fetch selects first row, delete selects next rowf
- (BOOL)**doesAutoSelect** Returns flag set by – **setAutoselect::**; default YES
- **setCurrentRecord:**(unsigned int)*newIndex* Sets a position within the DBRecordList
- **clearCurrentRecord** Deselects current record
- (unsigned int)**selectedRowAfter:**(unsigned int)*previousRow* Index of first selected row after *previousRow*
- **redisplayEverything** Displays all of DBFetchGroup's DBAssociations

Manipulating Contents

- **deleteCurrentSelection** Deletes the selected records from the DBRecordList
- (BOOL)**insertNewRecordAt:**(unsigned int)*index* Inserts a (default) record in the DBRecordList at *index*
- **fetchContentsOf:***aSource* **usingQualifier:***aQualifier* Replaces all records by reading *aSource* using *aQualifier*

Dealing with Changes

- (BOOL)**hasUnsavedChanges** YES if the DBRecordList has been changed but not saved
- (BOOL)**validateCurrentRecord** YES unless delegates for editor or DDModule object
- **saveChanges** Saves changes in this or subordinate fetch groups
- **discardChanges** Discards changes in this and subordinate fetch groups

Using Associations

- **addExpression:***newExpression* Adds *newExpression* to the list of expressions to fetch
- **takeValueFromAssociation:***anAssociation* Puts the displayed value into the DBRecordList
- **addAssociation:***newAssociation* Adds *newAssociation* to the list of associations
- **removeAssociation:***anAssociation* Removes *anAssociation* from the list of associations

Using a Delegate

- **delegate**
- **setDelegate:***anObject*

The object that receives notification messages
Sets the object to receive notification messages

Methods Implemented by the Delegate

- **fetchGroup:***fetchGroup*
didInsertRecordAt:(int)*index* Notification of a new record in the DBRecordList
- (BOOL)**fetchGroup:***fetchGroup*
willValidateRecordAt:(int)*index* Notification of pending validation; YES lets it proceed
- **fetchGroup:***fetchGroup*
willDeleteRecordAt:(int)*index* Notification of pending deletion; YES lets it proceed
- (DBFailureResponse)**fetchGroup:***fetchGroup*
willFailForReason:(DBFailureCode)*code* Returns constant to indicate response to failure notice
- **fetchGroupDidFetch:***fetchGroup* Notification of new contents in DBRecordList
- **fetchGroupDidSave:***fetchGroup* Notification that DBRecordList has been saved
- **fetchGroupWillChange:***fetchGroup* Notification that user made changes in the DBRecordList
- **fetchGroupWillFetch:***fetchGroup* Notification that fetch will change the DBRecordList
- (BOOL)**fetchGroupWillSave:***fetchGroup* Notification of pending save; YES lets it proceed

DBFormatter

(abstract superclass)

Inherits From: Object

Controlling the Data Source

- **dataSource** Returns the DBRecordList (or other source)
- **setDataSource:***newDataSource* Makes *newDataSource* the place to get values for display

Getting the Value to be Formatted

- **getValueAt:**(unsigned int) *row*
: *column*
inside:(NXRect *)*frame*
inView:*aView*
withAttributes:(id <DBTableVectors>) *rowAttrs*
:(id <DBTableVectors>) *columnAttrs*
usePositions:(BOOL) *useRowPos*
:(BOOL) *useColumnPos*
- Returns a DBValue from the DBRecordList, taking it from position *row* or *column* of the dynamic axis, using *rowAttrs* or *colAttrs* to identify static attributes, and flags *useRowPos* and *useColumnPos* to select which

Formatting a Field

- **drawFieldAt:**(unsigned int) *row*
: *column*
inside:(NXRect *)*frame*
inView:*aView*
withAttributes:(id <DBTableVectors>) *rowAttrs*
:(id <DBTableVectors>) *columnAttrs*
usePositions:(BOOL) *useRowPos*
:(BOOL) *useColumnPos*
- Displays one field of the data source's current record taken from position *row* or *column* of the dynamic axis, using *rowAttrs* or *colAttrs* to identify static attributes, and flags *useRowPos* and *useColumnPos* to select which

Batching Format Requests

- **beginBatching:**(id <DBTableVectors>) *attrs*
 - **endBatching**
 - **resetBatching:**(id <DBTableVectors>) *attrs*
- Notification that format *attrs* apply to all following items
Marks the end of a block of items formatted the same way
Begin batching if not already started

Appointing a Delegate

- **delegate**
 - **setDelegate:***anObject*
- The object that receives notification messages
Sets the object to receive notification messages

DBImageFormatter

Inherits From: DBFormatter : Object

Initializing

- **init** Initializes a new instance
- **free** Frees the space an instance formerly used

Default

- **setDefaultImage:anImage** Set image to be shown when the data has none
- **defaultImage** The image displayed when the data has none

Display

- **drawFieldAt:(unsigned int) row**
:column
inside:(NXRect *)frame
in View:aView
withAttributes:(id <DBTableVectors>) rowAttrs
:(id <DBTableVectors>)columnAttrs
usePositions:(BOOL)useRowPos
:(BOOL)useColumnPos
- Displays one image from the data source's current record taken from position *row* or *column* of the dynamic axis, using *rowAttrs* or *colAttrs* to identify static attributes, and flags *useRowPos* and *useColumnPos* to select which

Archiving

- **read:(NXTypedStream *)stream** Creates an instance by reading from a typed stream
- **write:(NXTypedStream *)stream** Archives an instance by writing to a typed stream

DBImageView

Inherits From: Control : View : Responder : Object

Internals

- **initWithFrame:**(const NXRect *)*frameRect* Initializes the view in the frame coordinates
- **drawSelf:**(const NXRect *)*rects*
:(int)*rectCount* Called by **display** to draw the image

Getting/Setting the Image

- **image** Returns the image being displayed
- **setImage:***newImage* Makes *newImage* the image to display

Getting/Setting the Border

- **setStyle:**(int)*newStyle* Sets the style of border for the image
- **style** Returns a constant indicating the border style

Editing

- **isEditable** (BOOL) YES if the image can be deleted or replaced
- **setEditable:**(BOOL)*flag* Allow/prohibit deleting or replacing the image

DBModule

Inherits From: Object

Reporting the Context

- **database** The DBModule's DBDatabase
- **entity** The DBModule's DBEntity
- **rootFetchGroup** The DBModule's root DBFetchGroup
- **associationForObject:***anObject* The DBAssociation that handles UI object *anObject*

- **editingAssociation**
- **getFetchGroups:**(List *)*aList*
- **fetchGroupNamed:**(const char *)*aName*

The DBAssociation currently involved in editing
Returns a list of all the DBModule’s DBFetchGroups
Returns the DBFetchGroup for the property named *aName*

Initializing and Configuring

- **initDatabase:***aDatabase*
entity:*anEntity*
- **fetchContentsOf:***aSource*
usingQualifier:*aQualifier*
- **addFetchGroup:***aFetchGroup*

Initializes a new DBModule with the given DBDatabase and DBEntity,
Fetches records for the DBEntity or DBValue *aSource*, using *aQualifier* to select records
Invoked to add *aFetchGroup* to the list of fetch groups

Responding to User Actions

- **fetchAllRecords:***sender*
- **saveChanges:***sender*
- **discardChanges:***sender*
- **deleteRecord:***sender*
- **appendNewRecord:***sender*
- **insertNewRecord:***sender*
- **nextRecord:***sender*
- **previousRecord:***sender*
- **takeValueFrom:***sender*
- **textDidEnd:***textObject*
endChar:(unsigned short)*whyEnd*
- (BOOL)**textWillChange:***textObject*
- (BOOL)**textWillEnd:***textObject*

Fetches all records for the DBModule’s root fetch group
Saves in the database changes made to the fetched records
Discard changes proposed for the fetched records
Delete one of the fetched records
Append a new (default) record to those fetched
Insert a new (default) record at the current position
Select the next of the fetched records
Select the preceding of the fetched records
UI object has a new value, so fetched record is revised
User has finished editing a text field
User has entered an editable field; YES lets editing proceed
Notification that an editable field will relinquish first responder; YES lets the change proceed

Using a Delegate

- **delegate**
- **setDelegate:***anObject*

The object that receives notification messages
Sets the object to receive notification messages

Methods Implemented by the Delegate

- **moduleDidSave:***module*
- (BOOL)**moduleWillLoseChanges:***module*
- (BOOL)**moduleWillSave:***module*

Called when *module* has completed a save to the database
Called when *module* is about to discard user’s changes
Called when *module* is about to save to the database

DBQualifier

Inherits From: Object

Conforms To: DBExpressionValues

Initializing and Freeing

- + **initialize** Automatically invoked to initialize the class
- **initWithEntity:(id <DBEntities>)anEntity** Initializes a new instance to select from *anEntity*
- **initWithEntity:(id <DBEntities>)anEntity
fromDescription:(const unsigned char *)descriptionFormat, ...** Initializes to select from *anEntity* by *descriptionFormat*
- **copyFromZone:(NXZone*)z** Returns a copy of the DBQualifier, allocating from *z*
- **free** Frees space that a DBQualifier formerly used

Modifying

- **addDescription:(const unsigned char *)descriptionFormat, ...** Appends *descriptionFormat* to the qualifier descriptions
- **setEntity:(id <DBEntities>)anEntity
andDescription:(const unsigned char *)descriptionFormat, ...** Sets both *anEntity* and qualifying *descriptionFormat*
- (BOOL)**setName:(const char *)aName** Assigns the DBQualifier *aName* and returns YES
- (BOOL)**empty** Deletes the qualifying descriptions and returns YES

Querying

- (const char *)**name** Returns the name assigned to the DBQualifier
- (id <DBEntities>)**entity** Returns the DBQualifier's entity
- (BOOL)**isEmpty** Returns YES if the qualifying descriptions are empty

Archiving

- **read:(NXTypedStream *)stream** Creates an instance by reading from a typed stream
- **write:(NXTypedStream *)stream** Archives an instance by writing to a typed stream

DBRecordList

Inherits From: DBRecordStream

Conforms To: DBContainers
DBCursorPositioning

Initializing and Freeing

- **init** Initializes a new instance of DBRecordList
- **free** Frees the space a DBRecordList formerly used
- **clear** Empties the record list and lists of properties

Setting the Retrieval Mode

- **setRetrieveMode:(DBRecordListRetrieveMode)aMode** Sets the DBRecordList's retrieval strategy
- **(DBRecordListRetrieveMode)currentRetrieveMode** Returns a constant identifying the retrieval strategy

Fetching Data from the Database

- **fetchRecordForRecordKey:(DBValue *)aValue** Fetches records qualified by matching *aValue*
- **fetchUsingQualifier:(DBQualifier *)aQualifier** Empties, then fetches records selected by *aQualifier*
- **fetchUsingQualifier:(DBQualifier *)aQualifier
emptyFirst:emptyFirst** Fetches records selected by *aQualifier*;
if *emptyFirst* is YES, first empties the record list
- **(unsigned int)recordLimit** Returns the maximum number of records to fetch
- **setRecordLimit:(unsigned int)count** Sets the maximum number of records to fetch

Accessing Data in the DBRecordList

- **getValue:(DBValue *)aValue
forProperty:aProperty** Puts the current record's value for *aProperty* into *aValue*
- **getValue:(DBValue *)aValue
forProperty:aProperty
at:(unsigned int)index** Puts value of *aProperty* for the record at *index* into *aValue*
- **getRecordKeyValue:(DBValue *)aValue** Puts the key value for the current record into *aValue*
- **getRecordKeyValue:(DBValue *)aValue
at:(unsigned int)index** Puts the key value for the record at *index* into *aValue*

Modifying Data in the DBRecordList

- **setValue:(DBValue *)aValue
forProperty:aProperty** Sets the current record's value of *aProperty* to *aValue*
- **setValue:(DBValue *)aValue
forProperty:aProperty
at:(unsigned int)index** Sets value of *aProperty* for record at *index* to *aValue*
- **insertRecordAt:(unsigned int)index** Inserts a (default) record ahead of the record at *index*
- **appendRecord** Inserts a (default) record after the last one
- **newRecord** Inserts a (default) record to precede the current record
- **(BOOL)isNewRecord** YES if the current record is one that has been inserted
- **(BOOL)isNewRecordAt:(unsigned int)index** YES if the record at *index* had been inserted
- **deleteRecord** Deletes the current record
- **deleteRecordAt:(unsigned int)index** Deletes the record at *index*
- **(BOOL)isModified** YES if the current record has been changed or inserted
- **(BOOL)isModifiedAt:(unsigned int)index** YES if the record at *index* has been changed or inserted
- **(BOOL)isModifiedForProperty:aProperty
at:(unsigned int)index** YES if *aProperty* of the record at *index* has been changed

Using Record Indexes

- **(unsigned int)positionForRecordKey:(DBValue *)aValue** Returns the index of the record whose key is *aValue*
- **moveRecordAt:(unsigned int)sourceIndex
to:(unsigned int)destinationIndex** Moves record at *sourceIndex* to precede the record now at *destinationIndex*
- **swapRecordAt:(unsigned int)anIndex
withRecordAt:(unsigned int)anotherIndex** Transposes the positions of the two records

Saving Data

- **(unsigned int)saveModifications** Saves to the database any changes since the fetch; returns code for success, partial success, or failure

DBRecordStream

Inherits From: Object

Initializing and Freeing

- **init** Initializes a new instance
- **free** Frees space formerly used by a DBRecordStream

Setting up a DBRecordStream

- **addRetrieveOrder:(DBRetrieveOrder)*anOrder* for:(id <DBProperties>)*aProperty*** Appends *anOrder* (up/down) to sort criteria for *aProperty*
- **(List *)setProperties:(List *)*propertyList* ofSource:*aSource*** Sets/returns list of properties wanted from entity *aSource*
- **(List *)getProperties:(List *)*propertyList*** Returns and puts into *propertyList* the stream's properties
- **(List *)setKeyProperties:(List *)*propertyList*** Sets and returns *propertyList* as the stream's key properties
- **(List *)getKeyProperties:(List *)*keyList*** Returns/ puts into *propertyList* the stream's key properties

Fetching Data

- **fetchUsingQualifier:(DBQualifier *)*aQualifier*** Starts fetching records that pass *aQualifier*
- **cancelFetch** Stops fetching and sends **fetchDone** to DBDatabase
- **(DBRecordRetrieveStatus)currentRetrieveStatus** DB_Ready/NotReady, DB_FetchInProgress/Completed

Accessing Data

- **getValue:(DBValue *)*aValue* forProperty:*aProperty*** Puts current record's *aProperty*'s value into *aValue*
- **getRecordKeyValue:(DBValue *)*aValue*** Puts current record's key value into *aValue*
- **setNext** Makes next record available; **nil** if none left

Modifying Data

- **setValue:(DBValue *)*aValue* forProperty:*aProperty*** Sets the current record's *aProperty* to *aValue*
- **newRecord** Inserts new, empty record at the current record
- **deleteRecord** Deletes the current record
- **(BOOL)isNewRecord** YES if the current record is a new one

- (BOOL)isModified YES if the current record is new or has been modified
- (BOOL)isReadOnly YES if the record stream cannot be modified

Saving Modifications

- (unsigned int)saveModifications Writes current record’s modifications to the database

Resetting a DBRecordStream

- clear Resets everything except the delegate

Assigning Delegates

- delegate The object that receives notification messages
- setDelegate:*anObject* Sets the object that will receive notification messages
- binderDelegate The object that receives notification messages for binders
- setBinderDelegate:*anObject* Sets the object to receive notification messages for binders

Method Implemented by the Delegate

- (BOOL)recordStream:*sender*
willFailForReason:(DBFailureCode) *aCode* Invoked when changes can’t be saved; *aCode* tells why; YES acknowledges failure; NO tries to proceed with those records that are not affected
- (BOOL)recordStreamPrepareCurrentRecordForModification:*aRecordStream* Invoked when a record will be modified or deleted; YES permits modification to proceed

DBTableVector

- Inherits From: Object
- Conforms To: DBTableVectors

Creating the Object

- initWithIdentifier:*anIdentifier* Initialize a DBTableVector for property *anIdentifier*
- free Free the space formerly allocated to a DBTableVector

DBTableView

Inherits From: UIScrollView : View : Responder : Object

Initializing and Freeing

- **initWithFrame:**(const NXRect *)*newFrame* Initializes an instance located within *newFrame*
- **free** Frees space formerly used by a DBTableView

Setting up the DBTableView

- **setDataSource:***aSource* Sets the object that will provide data for the display
- **dataSource** The object that provides data for the display
- **setDelegate:***delegate* Sets the object that will receive notification messages
- **delegate** The object that receives notification messages

Displaying

- **drawSelf:**(const NXRect *) *rects* :(int) *count*

Setting and Reporting Formatting

- **formatterAt:**(unsigned int)*row* :(unsigned int)*column* Returns the DBFormatter for the field at *row* and *column*
- (BOOL)**dynamicRows** YES if rows are dynamic
- (BOOL)**dynamicColumns** YES if columns are dynamic
- (BOOL)**isRowHeadingVisible** YES if row heading view is visible
- (BOOL)**isColumnHeadingVisible** YES if column heading view is visible
- **setIntercell:**(const NXSize *)*aSize* Sets space between neighboring rows and columns
- **getIntercell:**(NXSize *)*theSize* Puts space between rows and columns into *theSize*
- **setGridVisible:**(BOOL)*flag* Makes grid lines between rows and columns visible or not
- (BOOL)**isGridVisible** YES if grid lines are visible
- **acceptArrowKeys:**(BOOL)*flag* Makes arrow keys acceptable for navigation
- (BOOL)**doesAcceptArrowKeys** YES if arrow keys are accepted for navigation
- **allowVectorReordering:**(BOOL)*flag* Lets/prevents user drag static row/column to new position
- (BOOL)**doesAllowVectorReordering** YES if user is permitted to reorder static row or column
- **allowVectorResizing:**(BOOL)*flag* Lets/prevents user drag the width of static row or column
- (BOOL)**doesAllowVectorResizing** YES if user can drag row or column to change width

Notifying the DBTableView of Change

- **reloadData:***sender* Redraw because data may have changed
- **layoutChanged:***sender* Redraw because row or column spacing changed
- **rowsChangedFrom:**(unsigned int)*startRow*
to:(unsigned int)*endRow* Redraw because data changed in a block of rows
- **columnsChangedFrom:**(unsigned int)*startColumn* Redraw because data changed in a block of columns
to:(unsigned int)*endColumn*

Handling Rows and Columns

- (unsigned int)**columnCount** Total number of columns
- (unsigned int)**rowCount** Total number of rows
- (id <DBTableVectors>)**rowAt:**(unsigned int)*aPosition*
Object specifying format of the static row at *aPosition*
- (id <DBTableVectors>)**columnAt:**(unsigned int)*aPosition*
Object specifying format of the static column at *aPosition*
- **addColumn:***identifier*
at:(unsigned int)*aPosition* Adds a static column at *aPosition*
- **addColumn:***identifier*
withTitle:(const char *)*title* Adds a static column with *title* at *aPosition*
- **addColumn:***identifier*
withFormatter:*formatter*
andTitle:(const char *)*title*
at:(unsigned int)*aPosition* Adds a static column with *title* and *formatter* at *aPosition*
- **removeColumnAt:**(unsigned int)*columnPosition* Deletes a static column
- (BOOL)**moveColumnFrom:**(unsigned int)*oldPos*
to:(unsigned int)*newPos* Changes a static column's position
- **addRow:***identifier*
at:(unsigned int)*aPosition* Adds a static row at *aPosition*
- **addRow:***identifier*
withTitle:(const char *)*title* Adds a static row with *title* at *aPosition*
- **addRow:***identifier*
withFormatter:*formatter*
andTitle:(const char *)*title*
at:(unsigned int)*aPosition* Adds a static row with *title* and *formatter* at *aPosition*
- **removeRowAt:**(unsigned int)*rowPosition* Deletes a static row
- (BOOL)**moveRowFrom:**(unsigned int)*oldPos*
to:(unsigned int)*newPos* Changes a static row's position
- (unsigned int)**indexOfColumnWithIdentifier:***anIdentifier*
Position in sequence of static column *anIdentifier*
- (unsigned int)**indexOfRowWithIdentifier:***anIdentifier*
Position in sequence of static row *anIdentifier*

Editing Support

- **editFieldAt:**(unsigned int)*row*
:(unsigned int)*column* Selects an item and invokes editor
- **setEditable:**(BOOL)*flag* Enables/disables editing.
- (BOOL)**isEditable** YES of the DBTableView is editable.

Handling the Selection

- **setMode:**(int)*newMode* Make selection list mode, radio mode, or none.
- (int)**mode** Returns DB_NOSELECT/RADIOMODE/LISTMODE
- **allowEmptySel:**(BOOL)*flag* Allow/prohibit user to leave nothing selected
- (BOOL)**doesAllowEmptySel** YES if user may leave nothing selected
- (unsigned int)**selectedRowCount** Number of rows currently selected
- (unsigned int)**selectedColumnCount** Number of columns currently selected
- (int)**selectedRow** The row number of the selected row
- (int)**selectedColumn** The *column* number of the selected column
- (BOOL)**isRowSelected:**(unsigned int)*row* YES if *row* is selected
- (BOOL)**isColumnSelected:**(unsigned int)*column* YES if column is selected
- **deselectAll:***sender* Makes nothing selected.
- **selectAll:***sender* Makes all rows and columns selected.
- **setRowSelectionOn:**(unsigned int)*start*
:(unsigned int)*end*
to:(BOOL)*flag* Sets block of rows to selected (YES) or deselected (NO)
- **setColumnSelectionOn:**(unsigned int)*start*
:(unsigned int)*end*
to:(BOOL)*flag* Sets block of columns to selected (YES) or deselected
- **selectRow:**(unsigned int)*row*
byExtension:(BOOL)*flag* Selects row, or extends selection if flag is YES
- **selectColumn:**(unsigned int)*column*
byExtension:(BOOL)*flag* Selects row, or extends selection if flag is YES
- **deselectRow:**(unsigned int)*row* Deselects the indicated row
- **deselectColumn:**(unsigned int)*column* Deselects the indicated column
- (unsigned int)**selectedRowAfter:**(unsigned int)*aRow*
Index of the first selected row after *aRow*
- (unsigned int)**selectedColumnAfter:**(unsigned int)*aColumn*
Index of the first selected column after *aColumn*
- **sendAction:**(SEL)*anAction*
to:*anObject*
forSelectedRows:(BOOL)*flag* Sends *anAction* to *anObject* for each selected row;
if YES, does it for each selected row

- **sendAction:**(SEL)*anAction*
to:*anObject*
forSelectedColumns:(BOOL)*flag*

Sends *anAction* to *anObject* for each selected column; if YES, does it for each selected column

Setting DBTableView Components

- **rowHeading**
- **setRowHeading:***newRowHeading*
- **setRowHeadingVisible:**(BOOL)*flag*
- **columnHeading**
- **setColumnHeading:***newColumnHeading*
- **setColumnHeadingVisible:**(BOOL)*flag*

Returns the row heading view
Makes *newRowHeading* the row heading view
Makes the row heading visible or not
Returns the column heading view
Makes *newColumnHeading* the column heading view
Makes the column heading visible or not

Adjusting the View

- **display**
- **scrollClip:***aClip*
to:(const NXPoint *)*newOrigin*
- (BOOL)**isHorizScrollerVisible**
- **setHorizScrollerVisible:**(BOOL)*flag*
- (BOOL)**isVertScrollerVisible**
- **setVertScrollerVisible:**(BOOL)*flag*
- **tile**
- **sizeTo:**(NXCoord)*width*
:(NXCoord)*height*
- **scrollRowToVisible:**(unsigned int)*row*
- **scrollColumnToVisible:**(unsigned int)*column*
- (BOOL)**acceptsFirstResponder**

Displays the DBTableView
Sets *aClip*'s origin to be *newOrigin* in the content view
YES if the content view's horizontal scroller is enabled
Makes the content view's horizontal scroller visible or not
YES if the content view's vertical scroller is visible
Makes the content view's vertical scroller visible
Recalculate positions of the component views and redraw
Adjust the overall size to *width* and *height*, and redraw
Scroll the content so that *row* is visible in the scroll clip
Scroll the content so that *column* is visible in the scroll clip
YES if the DBTableView will handle keyboard events

Transmitting Action

- **setAction:**(SEL)*aSelector*
- (SEL)**action**
- **setDoubleAction:**(SEL)*aSelector*
- (SEL)**doubleAction**
- **setTarget:***anObject*
- **target**

Makes *aSelector* the action in response to a click
The action to be sent on a click
Makes *aSelector* the action in response to a double click
The action in response to a double click
Makes anObject the target for an action message
The target for an action message

Archiving

- **read:**(NXTypedStream *)*stream* Creates an instance by reading from a typed stream
- **write:**(NXTypedStream *)*stream* Archives an instance by writing to a typed stream
- **finishUnarchiving** Automatically invoked final step in unarchiving

DBTextFormatter

Inherits From: DBFormatter : Object

Initializing

- **init** Initializes a new DBTextFormatter instance
- **free** Frees the space allocated to a DBTextFormatter.

Manipulating Font

- **font** Returns the formatter's font
- **setFont:***aFont* Makes *aFont* the formatter's font

Batching Format Requests

- **beginBatching:**(id <DBTableVectors>) *attrs* The format *attrs* applies to all following records
- **resetBatching:**(id <DBTableVectors>) *attrs* Begins batching if not already in effect
- **endBatching** Completes sequence of records in same format

Archiving

- **read:**(NXTypedStream *)*stream* Creates an instance by reading from a typed stream
- **write:**(NXTypedStream *)*stream* Archives an instance by writing to a typed stream

DBValue

Inherits From: Object

Conforms To: DBExpressionValues

Creating and Freeing

- + **initialize** Initialize the class (sent by a subclass)
- **init** Initialize a DBValue instance
- **free** Free space formerly used by a DBValue instance

Setting Values

- **setDoubleValue:(double)aDouble** Sets the object's value to *aDouble*
- **setFloatValue:(float)aFloat** Sets the object's value to *aFloat*
- **setIntValue:(int)anInt** Sets the object's value to *anInt*
- **setObjectValue:(id)anObject** Sets the object's value to *anObject*
- **setObjectValueNoCopy:(id)anObject** Sets the object's value so that it points to *anObject*
- **setStringValue:(const char *)aString** Sets the object's value to *aString*
- **setStringValueNoCopy:(const char *)aString** Sets the object's value so that it points to *aString*
- **setValueFrom:(DBValue *)aValue** Sets the object's to have the same value as *aValue*
- **setNull** Sets the object's value to NULL

Reporting Values

- (id <DBTypes>) **valueType** Returns the type of value the object contains
- (BOOL) **isEqual:(DBValue *)anotherValue** YEs if this object has same type and value as *anotherValue*
- (double) **doubleValue** Returns the object's value as a double
- (float) **floatValue** Returns the object's value as a float
- (int) **intValue** Returns the object's value as an int
- **objectValue** Returns the object's value as an object
- (const char *) **stringValue** Returns the object's value as a string
- (BOOL) **isNull** YES if the object's value is NULL

Archiving

- **read:(NXTypedStream *)stream** Creates an instance by reading from a typed stream
- **write:(NXTypedStream *)stream** Archives an instance by writing to a typed stream

Protocols

DBContainers

Adopted By: DBRecordList

Mandatory Methods

- **addObject:***anObject*
forBinder:(DBBinder *)*aBinder* Adds *anObject* to *aBinder*'s container
- (unsigned int)**count** The number of objects in the container
- **empty** Removes (but doesn't free) objects in the container
- **freeObjects** Frees space formerly allocated to objects in the container
- **objectAt:**(unsigned)*index*
forBinder:(DBBinder *)*aBinder* Returns the *index*'th object in *aBinder*'s container
- (unsigned int)**prepareForBinder:**(DBBinder *)*aBinder* Readies the container to move data; returns **count**

Optional Methods

- **binder:**(DBBinder *)*aBinder*
didAcceptObject:*anObject* Notification when *anObject* successfully stored in *aBinder*
- **binder:**(DBBinder *)*aBinder*
didRejectObject:*anObject* Notification when *anObject* could not be stored in *aBinder*

DBCursorPositioning

Adopted By: DBBinder
DBRecordList

Setting the Position

- **setFirst** Sets cursor to container's first record; returns the record

- **setLast** Sets cursor to container’s last record; returns the record
- **setNext** Sets cursor to container’s next record; returns the record
- **setPrevious** Sets cursor to container’s previous record; returns the record
- **setTo:(long)*index*** Sets cursor to *index*; returns the *index*’th record

Querying the Position

- (long)**currentPosition** Returns the current index (of objects in the container)

DBCUSTOMASSOCIATION (informal protocol)

Category Of: Object

Access to the Associated Value

- **association:association** Sets *value* into the associated data source
 setValue:(DBValue *)*value*
- **association:association** Puts the value of the associated data source into *value*
 getValue:(DBValue *)*value*;

Notifications to the Associated Display

- **associationContentsDidChange:association** Notification that the data source has changed
- **associationSelectionDidChange:association** Notification of a change in selection in the display
- **associationCurrentRecordDidDelete:association** Notification that current record has been deleted from the data source

DBEntities

Adopted By: none

Incorporates: DBTypes

Querying the Object

- (const char *)**name** Returns the name of the entity
- (DBDatabase *)**database** The DBDatabase object that created the entity
- **getProperties:**(List *)*aList* Puts the entity's properties into *aList*
- **propertyName:**(const char *)*aName* Returns the DBProperty named *aName*

Comparing the Object

- (BOOL)**matchesEntity:**(id <DBEntities>)*anEntity* YES if receiver is equivalent to *anEntity*

DBExpressionValues

Adopted By: DBExpression
DBQualifier
DBValue

Methods

- (const char *)**expressionValue** Returns the receiver's query-language expression
- (BOOL)**isDeferredExpression** YES if evaluation of this expression can be deferred

DBFormatConversion

(informal protocol)

Category Of: Object

- **writeBuffer:**(void*)*buffer*
 ofLength:(unsigned)*length*
 withFormat:(const char*)*aFormatString* If implemented, invoked automatically when custom object is written to the database

DBFormatInitialization

(informal protocol)

Category Of: Object

- **initFromBuffer:**(void*)*buffer*
 ofLength:(unsigned)*length*
 withFormat:(const char*)*aFormatString* If implemented, invoked automatically when custom object is written to the database

DBFormatterValidation

(informal protocol)

Category Of: Object

Notification Using the Changed Cell's Identifiers

- **formatterDidChangeValueFor:***rowIdentifier*
 :*columnIdentifier*
 sender:*sender* Notification when a formatter has changed a value in one of the fields of a display; returns **self**

- (BOOL)**formatterWillChangeValueFor:rowIdentifier**
 :*columnIdentifier* Notification when a formatter will change a value
 sender:sender
- (BOOL)**formatterWillChangeValueFor:rowIdentifier**
 :*columnIdentifier* Notification when a formatter will change a value,
 to:aValue with the proposed new value
 sender:sender

Notification Using the Changed Cell's Position

- **formatterDidChangeValueFor:identifier** Notification when a formatter has changed a value
 at:(unsigned int)position in one of the fields of a display; returns **self**
 sender:sender
- (BOOL)**formatterWillChangeValueFor:identifier**
 at:(unsigned int)position Notification when a formatter will change a value
 sender:sender
- (BOOL)**formatterWillChangeValueFor:identifier**
 at:(unsigned int)position Notification when a formatter will change a value,
 to:aValue with the proposed new value
 sender:sender

DBFormatterViewEditing

Category Of: Object

Accepting changes

- (BOOL)**formatterEndedEditing:sender** Invoked by editor when user ends editing;
 endChar:(unsigned short)whyEnd returns YES unless editing in progress

DBProperties

Adopted By: DBExpression

Identifying a Property

- (const char*)**name** Name of the property
- (BOOL)**setName:(const char *)aName** Set the name of a custom property; YES if successful
- (id <DBEntities>)**entity** The entity to which the property belongs
- (BOOL)**matchesProperty:(id <DBProperties>)aProperty**
YES if receiver identifies same thing as *aProperty*

Querying a Property

- (id <DBTypes>) **propertyType** The type, as an object that responds to DBTypes protocol
- (BOOL)**isSingular** YES if receive is an attribute or a one-to-one relationship
- (BOOL)**isReadOnly** YES if the property's data is read only
- (BOOL)**isKey** YES if the property is a key property in the entity
- (BOOL)**matchesProperty:(id <DBProperties>)aProperty**
YES if receiver has same name in same entity as *aProperty*

DBTableDataSources (informal protocol)

Category Of: Object

Reporting Table Size

- (unsigned int)**rowCount** Number of rows in the data source table
- (unsigned int)**columnCount** Number of columns in the data source table

Getting/Setting Data

- **getValueFor:rowIdentifier**
:columnIdentifier
into:aValue
Puts source value identified by properties *rowIdentifier* and *columnIdentifier* into *aValue*
 - **getValueFor:identifier**
at:(unsigned int) aPosition
into:aValue
Puts value for property *identifier* at record *aPosition* into *aValue*
 - **setValueFor:rowIdentifier**
:columnIdentifier
from:aValue
Sets source value identified by properties *rowIdentifier* and *columnIdentifier* to *aValue*
 - **setValueFor:identifier**
at:(unsigned int) aPosition
from:aValue
Sets value for property *identifier* at record *aPosition* to *aValue*
-

DBTableValues (informal protocol)

Category Of: Object

Getting a Value from the Table

- **objectValue**
The item's value as an object
- **(int)intValue**
The item's value as an int
- **(double)doubleValue**
The item's value as a double
- **(float)floatValue**
The item's value as a float
- **(const char *)stringValue**
The item's value as a string

Setting a Value in the Table

- **setObjectValue:(int)anObj**
Sets the item's value to *anObj*
- **setIntValue:(int)anInt**
Sets the item's value to *anInt*
- **setDoubleValue:(double)aDouble**
Sets the item's value to *aDouble*
- **setFloatValue:(float)aFloat**
Sets the item's value to *aFloat*
- **setStringValue:(const char *) aString**
Sets the item's value to *aString*

DBTableVectors

Category Of: Object

Controlling/Reporting Formatter

- formatter The formatter responsible for formatting the vector
- **setFormatter:***newFormatter* Makes *newFormatter* the vector's formatter

Controlling/Reporting Data Link

- identifier The identifier for the property that the vector displays
- **setIdentifier:***aDataAttribute* Makes *aDataAttribute* the vector's identifier

Controlling/Reporting Editing

- (BOOL)**isEditable** YES if the vector's display is editable
- **setEditable:**(BOOL)*flag* Permit/prohibit editing of the vector's display

Controlling/reporting Size

- (BOOL)**isResizable** YES if the vector's display can be resized
- **setResizable:**(BOOL)*flag* Permit/prohibit resizing the vector's display
- (BOOL)**isAutosizable** YES if automatically resized when the view is resized
- **setAutosizable:**(BOOL)*flag* Permit/prohibit automatic resizing
- (NXCoord)**size** Width and height of an item in the vector's display
- (NXCoord)**sizeTo:**(NXCoord)*newSize* Sets width and height of an item in the vector's display
- (NXCoord)**minSize** The minimum limit on resizing
- **setMinSize:**(NXCoord) *newMinSize* Sets minimum limit on resizing
- (NXCoord)**maxSize** The maximum limit on resizing
- **setMaxSize:**(NXCoord) *newMaxSize* Sets maximum limit on resizing

Controlling/Reporting Title

- (const char *)**title** The title (if row or column headings are shown)
- **setTitle:**(const char *) *title* Makes *title* the vector's title
- **titleFont** The font for displaying the title
- **setTitleFont:***fontObj* Makes *fontObj* the font for displaying the title

- (int)**titleAlignment** A constant: NX_LEFT/CENTERED/RIGHTALIGNED
- **setTitleAlignment:(int)align** Sets the title alignment to *align*

Controlling/Reporting Content Alignment

- (int)**contentAlignment** A constant: NX_LEFT/CENTERED/RIGHTALIGNED
- **setContentAlignment:(int) align** Sets the content alignment to *align*

DBTransactions

Adopted By: DBBasicAdaptor : Object

Basic Transaction Commands

- (BOOL)**beginTransaction** Begins a transaction; YES if successful
- (BOOL)**commitTransaction** Commits a transaction; YES if successful
- (BOOL)**rollbackTransaction** Rolls back a transaction; YES if successful

DBTypes

Adopted By: none

Querying for Type

- (const char *)**objcType** "id" / "char *" / "int" / "float" / "double" as appropriate
- (const char *)**databaseType** What the adaptor returns for the data's type, for example
"@" (for id) / "*" (for char *) / "i" / "f" / "d"
- (const char *)**objcClassName** Name of the stored data's class (if any)

Comparing Types

- (BOOL)**isEntity** YES if the object conforms to DBEntities protocol
- (BOOL)**matchesType:(id <DBTypes>)anObject** YES if *anObject* has the same data type as the receiver

Types and Constants

Defined Types

Access Exceptions

```
typedef enum _DBAccessErrors {  
    DB_UnimplementedException = DB_ERROR_BASE,  
    DB_CoercionException,  
    DB_FormatException,  
    DB_CursorException,  
    DB_CommitException  
} DBExceptions;
```

Failure Codes

```
typedef enum {  
    DB_ReasonUnknown = 0,  
    DB_RecordBusy,  
    DB_RecordStreamNotReady,  
    DB_RecordHasChanged,  
    DB_RecordLimitReached,  
    DB_NoRecordKey,  
    DB_RecordKeyNotUnique,  
    DB_NoAdaptor,  
    DB_AdaptorError,  
    DB_TransactionError  
} DBFailureCode;
```

Failure Responses

```
typedef enum {  
    DB_NotHandled = 0,  
    DB_Abort,  
    DB_Continue  
} DBFailureResponse;
```


Image Style

```
typedef enum {  
    DB_ImageNoFrame = 0,  
    DB_ImagePhoto,  
    DB_ImageGrayBezel,  
    DB_ImageGroove  
} DBImageStyle;
```

Order of Retrieved Records

```
typedef enum {  
    DB_NoOrder = 0  
    DB_AscendingOrder,  
    DB_DescendingOrder  
} DBRetrieveOrder;
```

Retrieval Mode of a Record List

```
typedef enum _DBRecordListMode {  
    DB_SynchronousStrategy,  
    DB_BackgroundStrategy,  
    DB_BackgroundNoBlockingStrategy,  
} DBRecordListRetrieveMode;
```

Selection Mode in a DBTableView

```
typedef enum {  
    DB_RADIOMODE = 0,  
    DB_LISTMODE = 2,  
    DB_NOSELECT = 5  
} DBSelectionMode
```

Status of Record Retrieval

```
typedef enum _DBRecordRetrievalStatus {  
    DB_NotReady,  
    DB_Ready,  
    DB_FetchLimitReached,  
    DB_FetchInProgress,  
    DB_FetchCompleted  
} DBRecordRetrieveStatus;
```

Symbolic Constants

Error Numbers' Base Value

DB_ERROR_BASE (6000)

Format Types

DBFormat_EPS "EPS"
DBFormat_RTF "RTF"
DBFormat_TIFF "TIFF"

No Index Indicator

DB_NoIndex 0xffffffff

Null Values

DB_NullDouble (NAN)
DB_NullFloat (NAN)
DB_NullInt ((int)0x7ffffffe)

Record Limit Default

DB_DEFAULT_RECORD_LIMIT 1000



5 *Display PostScript*

PostScript Operators

This section summarizes the PostScript® operators. The following notation is used:

Notation	Means
(Display)	Extensions that were made by Adobe Systems Incorporated and NeXTSTEP for the Display PostScript System.
(NeXTSTEP)	NeXTSTEP extensions to the Display PostScript System.
*	Use this operator only if you're bypassing the Application Kit. Your use of this operator within an application based on the Application Kit will conflict with the Kit's use.

This summary is organized into groups of related operators. The format used is the same as in the *PostScript Language Reference Manual*. Also as in that manual, the operand and result names suggest their types. Names representing numbers sometimes suggest their purpose (such as *angle* or *window*); in operators implemented by NeXTSTEP, these names represent integers except for the following, which are real:

- Coordinates, usually named as (or ending in) *x* and *y*
- Widths and heights, usually so named
- Coverage, a measure of transparency in “alpha” operators
- Time (named *secs*, and measured in seconds)

Several operators can optionally take an encoded number string as an operand, as indicated by *numstring* in the listing below. Use encoded number strings only when drawing to the screen. An application can use the global variable NXDrawingStatus to determine whether it's drawing to the screen.

Compositing and Transparency Operators (NeXTSTEP)

<i>src_x src_y width height srcgstate</i> <i>dest_x dest_y op</i>	composite -	composite rectangle in source graphics state with image in current window
<i>dest_x dest_y width height op</i>	compositerect -	composite rectangle of current color and coverage with image in current graphics state
<i>src_x src_y width height srcgstate</i> <i>dest_x dest_y delta</i>	dissolve -	dissolve between area of window referred to by <i>srcgstate</i> and equal area of window referred to by current graphics state
<i>window</i>	currentwindowalpha <i>state</i>	return information about how window's alpha values are stored
<i>x y width height proc₀ [... proc_{n-1}]</i> <i>string bool</i>	readimage -	read image's pixel values and pass to corresponding procedures
<i>x y width height matrix</i>	sizeimage <i>pixelswide pixelshigh bits/sample matrix multiproc ncolors</i>	get various parameters required for readimage to read the image
<i>pixelswide pixelshigh bits/sample</i> <i>matrix proc₀ [... proc_n] multiproc</i> <i>ncolors</i>	alphaimage -	render data and alpha information supplied by one or more procedures

Instance Drawing Operators (NeXTSTEP)

-	newinstance -	remove instance drawing from current window
<i>bool</i>	setinstance -	turn instance-drawing mode on or off
<i>x y width height</i>	hideinstance -	remove instance drawing from rectangle

Mouse and Cursor Operators (NeXTSTEP)

<i>eventnum</i>	stilldown <i>bool</i>	test whether left/only mouse button is still down from mouse-down <i>eventnum</i>
<i>eventnum</i>	rightstilldown <i>bool</i>	test whether right mouse button is still down from mouse-down <i>eventnum</i>
<i>window</i>	currentmouse <i>x y</i>	return mouse location in base coordinates *
-	buttondown <i>bool</i>	test whether left/only mouse button is down

	– rightbuttondown	<i>bool</i>	test whether right mouse button is down
<i>x y</i>	setmouse	–	set mouse and cursor location
<i>dx dy</i>	adjustcursor	–	adjust cursor location by (<i>dx</i> , <i>dy</i>)
<i>x y mx my</i>	setcursor	–	set cursor to image with upper left at (<i>x</i> , <i>y</i>) and hot spot at offset (<i>mx</i> , <i>my</i>) from (<i>x</i> , <i>y</i>).
	– hidecursor	–	remove cursor from screen
	– showcursor	–	restore cursor to screen
	– obscurecursor	–	remove cursor from screen until mouse moves
	– revealcursor	–	restore cursor if still obscured
<i>bool context</i>	setwaitcursorenabled	–	enable or disable wait cursor
<i>context</i>	currentwaitcursorenabled	<i>bool</i>	return whether wait cursor is enabled
<i>x y width height leftbool rightbool insidebool userdata trectnum gstate</i>	settrackingrect	–	set tracking rectangle in window referred to by <i>gstate</i>
<i>trectnum gstate</i>	cleartrackingrect	–	clear tracking rectangle in <i>gstate</i>

Event Operators (NeXTSTEP)

<i>mask window</i>	seteventmask	–	set window's Server-level event mask *
<i>window</i>	currenteventmask	<i>mask</i>	return window's current Server-level event mask *
<i>type x y time flags window subType misc0 misc1 context</i>	posteventbycontext	<i>bool</i>	post event to specified context
<i>bool window</i>	setsendexposed	–	set whether window-changed events are generated for exposed window areas *
<i>bool</i>	setflushexposures	–	set whether window-exposed and screen-changed subevents are flushed *
<i>bool context</i>	setwaitcursorenabled	–	enable or disable wait cursor operation
<i>context</i>	currentwaitcursorenabled	<i>bool</i>	return status of wait cursor in <i>context</i>
<i>context</i>	setactiveapp	–	establish application having <i>context</i> as the active application*
	– currentactiveapp	<i>context</i>	return context of active application *

Frame Buffer Operators (NeXTSTEP)

	– countframebuffers	<i>count</i>	return number of frame buffers in use
	<i>fbnum</i>	currentframebuffertransfer	<i>redproc greenproc blueproc grayproc</i> return current transfer function for <i>fbnum</i>
<i>index string</i>	framebuffer	<i>name slot unit romid x y width height maxdepth</i>	provide information on specific frame buffer
<i>redproc greenproc blueproc</i> <i>grayproc fbnum</i>	setframebuffertransfer	–	set the transfer function for <i>fbnum</i>

Window Management Operators (NeXTSTEP)

<i>x y width height type</i>	window	<i>window</i>	create window and return its number *
	<i>window</i>	termwindow	– remove window from screen list; cause eventual freeing *
<i>type window</i>	setwindowtype	–	set <i>window</i> 's type to <i>type</i>
<i>window</i>	windowdevice	–	set device of current graphics state to <i>window</i>
<i>window</i>	windowdeviceround	–	set device to <i>window</i> and round coordinate system to integer pixels
	– currentwindow	<i>window</i>	return window number of current device
	– setexposurecolor	–	set exposure color for nonretained window of current graphics state
	– flushgraphics	–	flush drawing in buffered window to screen
<i>place otherwindow window</i>	orderwindow	–	order Above or Below <i>otherwindow</i> (0 for all) or Out of screen list *
<i>level window</i>	setwindowlevel	–	set window tier for <i>window</i> to <i>level</i>
	<i>window</i>	currentwindowlevel	<i>level</i> return window tier for <i>window</i>
	– frontwindow	<i>window</i>	return frontmost window *
<i>x y width height window</i>	placewindow	–	reposition and resize with intersecting pixels unchanged *
<i>x y window</i>	movewindow	–	move lower left to screen coordinates (<i>x</i> , <i>y</i>) *
	<i>window</i>	currentwindowbounds	<i>x y width height</i> return window's location and size in screen coordinates
<i>x y place otherwindow</i>	findwindow	<i>x' y' window bool</i>	locate window under screen coordinates (<i>x</i> , <i>y</i>) and give base coordinates (or return <i>false</i>)

<i>bool window</i>	setautofill	–	set whether exposure color fills window's exposed areas automatically
<i>dict window</i>	setwindowdict	–	set window's dictionary *
<i>window</i>	currentwindowdict	<i>dict</i>	return window's dictionary *
<i>context</i>	countscreenlist	<i>count</i>	return number of windows in screen list that belong to <i>context</i>
<i>context</i>	countwindowlist	<i>count</i>	return number of windows that belong to <i>context</i>
<i>array context</i>	screenlist	<i>subarray</i>	return window numbers of all windows in screen list that belong to <i>context</i>
<i>array context</i>	windowlist	<i>subarray</i>	return window numbers of all windows that belong to <i>context</i>
<i>context window</i>	setowner	–	set owning PostScript context of <i>window</i> to <i>context</i>
<i>window</i>	currentowner	<i>context</i>	return PostScript context that owns <i>window</i>
<i>window</i>	currentdeviceinfo	<i>min max bool</i>	return window's sampling density and whether device is color
<i>depth</i>	setdefaultdepthlimit	–	set depth limit for new windows *
	– currentdefaultdepthlimit	<i>depth</i>	return depth limit for new windows *
<i>window</i>	currentwindowdepth	<i>depth</i>	return <i>window</i> 's depth *
<i>depth window</i>	setwindowdepthlimit	–	set <i>window</i> 's depth limit *
<i>window</i>	currentwindowdepthlimit	<i>depth</i>	return <i>window</i> 's depth limit *
<i>dumplevel window</i>	dumpwindow	–	report position and number of bytes of backing store for <i>window</i> *
<i>dumplevel context</i>	dumpwindows	–	report position and number of bytes of backing store for all windows owned by <i>context</i> *

statusdict Operators (NeXTSTEP)

– ostype	<i>int</i>	return category of operating system (1=standalone, 3=UNIX® variant)
– osname	<i>string</i>	return name of operating system

Operand Stack Manipulation Operators

<i>any</i>	pop	–	discard top element
<i>any₁ any₂</i>	exch	<i>any₂ any₁</i>	exchange top two elements
<i>any</i>	dup	<i>any any</i>	duplicate top element
<i>any₁ ... any_n n</i>	copy	<i>any₁ ... any_n any₁ ... any_n</i>	duplicate top <i>n</i> elements
<i>any_n ... any₀ n</i>	index	<i>any_n ... any₀ any_n</i>	duplicate arbitrary element
<i>a_{n-1} ... a₀ n j</i>	roll	<i>a_{(j-1) mod n} ... a₀ a_{n-1} ... a_{j mod n}</i>	roll <i>n</i> elements up <i>j</i> times
<i> - any₁ ... any_n</i>	clear	<i> -</i>	discard all elements
<i> - any₁ ... any_n</i>	count	<i> - any₁ ... any_n n</i>	count elements on stack
–	mark	<i>mark</i>	push mark on stack
<i>mark obj₁ ... obj_n</i>	cleartomark	–	discard elements down through mark
<i>mark obj₁ ... obj_n</i>	counttomark	<i>mark obj₁ ... obj_n n</i>	count elements down to mark

Arithmetic and Math Operators

<i>num₁ num₂</i>	add	<i>sum</i>	<i>num₁ plus num₂</i>
<i>num₁ num₂</i>	div	<i>quotient</i>	<i>num₁ divided by num₂</i>
<i>int₁ int₂</i>	idiv	<i>quotient</i>	integer divide
<i>int₁ int₂</i>	mod	<i>remainder</i>	<i>int₁ mod int₂</i>
<i>num₁ num₂</i>	mul	<i>product</i>	<i>num₁ times num₂</i>
<i>num₁ num₂</i>	sub	<i>difference</i>	<i>num₁ minus num₂</i>
<i>num₁</i>	abs	<i>num₂</i>	absolute value of <i>num₁</i>
<i>num₁</i>	neg	<i>num₂</i>	negative of <i>num₁</i>
<i>num₁</i>	ceiling	<i>num₂</i>	ceiling of <i>num₁</i>
<i>num₁</i>	floor	<i>num₂</i>	floor of <i>num₁</i>
<i>num₁</i>	round	<i>num₂</i>	round <i>num₁</i> to nearest integer
<i>num₁</i>	truncate	<i>num₂</i>	remove fractional part of <i>num₁</i>
<i>num</i>	sqrt	<i>real</i>	square root of <i>num</i>
<i>num den</i>	atan	<i>angle</i>	arctangent of <i>num/den</i> in degrees
<i>angle</i>	cos	<i>real</i>	cosine of <i>angle</i> (degrees)

<i>angle</i>	sin	<i>real</i>	sine of <i>angle</i> (degrees)
<i>base</i>	exp	<i>real</i>	raise <i>base</i> to <i>exponent</i> power
<i>num</i>	ln	<i>real</i>	natural logarithm (base e)
<i>num</i>	log	<i>real</i>	logarithm (base 10)
	rand	<i>int</i>	generate pseudo-random integer
<i>int</i>	srand	-	set random number seed
	rrand	<i>int</i>	return random number seed

Array Operators

<i>int</i>	array	<i>array</i>	create array of length <i>int</i>
	[<i>mark</i>	start array construction
<i>mark obj₀ ... obj_{n-1}</i>]	<i>array</i>	end array construction
<i>array</i>	length	<i>int</i>	number of elements in <i>array</i>
<i>array index</i>	get	<i>any</i>	get array element indexed by <i>index</i>
<i>array index any</i>	put	-	put <i>any</i> into array at <i>index</i>
<i>array index count</i>	getinterval	<i>subarray</i>	subarray of array starting at <i>index</i> for <i>count</i> elements
<i>array₁ index array₂</i>	putinterval	-	replace subarray of <i>array₁</i> starting at <i>index</i> by <i>array₂</i>
<i>array</i>	aload	<i>a₀ ... a_{n-1} array</i>	push all elements of array on stack
<i>any₀ ... any_{n-1} array</i>	astore	<i>array</i>	pop elements from stack into array
<i>array₁ array₂</i>	copy	<i>subarray₂</i>	copy elements of <i>array₁</i> to initial subarray of <i>array₂</i>
<i>array proc</i>	forall	-	execute <i>proc</i> for each element of array
<i>any₀ ... any_{n-1} n</i>	packedarray	<i>packedarray</i>	create packed array consisting of specified <i>n</i> elements
	currentpacking	<i>bool</i>	return array packing mode
<i>bool</i>	setpacking	-	set current array packing mode for '{...}' syntax (true = packedarray)
<i>packedarray</i>	length	<i>int</i>	number of elements in <i>packedarray</i>
<i>packedarray index</i>	get	<i>any</i>	get <i>packedarray</i> element indexed by <i>index</i>
<i>packedarray index count</i>	getinterval	<i>subarray</i>	subarray of <i>packedarray</i> starting at <i>index</i> for <i>count</i> elements

<i>packedarray</i>	aload	$a_0 \dots a_{n-1}$ <i>packedarray</i>	push all elements of <i>packedarray</i> on stack
<i>packedarray</i> ₁ <i>array</i> ₂	copy	<i>subarray</i> ₂	copy elements of <i>packedarray</i> ₁ to initial subarray of <i>array</i> ₂
<i>packedarray proc</i>	forall	–	execute <i>proc</i> for each element of <i>packedarray</i>

Dictionary Operators

<i>int</i>	dict	<i>dict</i>	create dictionary with capacity for <i>int</i> elements
<i>dict</i>	length	<i>int</i>	number of key-value pairs in <i>dict</i>
<i>dict</i>	maxlength	<i>int</i>	capacity of <i>dict</i>
<i>dict</i>	begin	–	push <i>dict</i> on <i>dict</i> stack
–	end	–	pop <i>dict</i> stack
<i>key value</i>	def	–	associate <i>key</i> and <i>value</i> in current <i>dict</i>
<i>key</i>	load	<i>value</i>	search <i>dict</i> stack for <i>key</i> and return associated <i>value</i>
<i>key value</i>	store	–	replace topmost definition of <i>key</i>
<i>dict key</i>	get	<i>any</i>	get <i>value</i> associated with <i>key</i> in <i>dict</i>
<i>dict key value</i>	put	–	associate <i>key</i> with <i>value</i> in <i>dict</i>
–	cleardictstack	–	return dictionary stack to initial state
<i>dict key</i>	known	<i>bool</i>	test whether <i>key</i> is in <i>dict</i>
<i>key</i>	where	<i>dict true</i> or <i>false</i>	find <i>dict</i> in which <i>key</i> is defined
<i>dict</i> ₁ <i>dict</i> ₂	copy	<i>dict</i> ₂	copy contents of <i>dict</i> ₁ to <i>dict</i> ₂
<i>dict proc</i>	forall	–	execute <i>proc</i> for each element of <i>dict</i>
–	errordict	<i>dict</i>	push errordict on operand stack
–	systemdict	<i>dict</i>	push systemdict on operand stack
–	userdict	<i>dict</i>	push userdict on operand stack
–	currentdict	<i>dict</i>	push current <i>dict</i> on operand stack
–	countdictstack	<i>int</i>	count elements on <i>dict</i> stack
<i>array</i>	dictstack	<i>subarray</i>	copy <i>dict</i> stack into <i>array</i>

String Operators

<i>int</i>	string	<i>string</i>		create string of length <i>int</i>
<i>string</i>	length	<i>int</i>		number of elements in string
<i>string index</i>	get	<i>int</i>		get string element indexed by <i>index</i>
<i>string index int</i>	put	-		put <i>int</i> into string at <i>index</i>
<i>string index count</i>	getinterval	<i>substring</i>		substring of string starting at <i>index</i> for <i>count</i> elements
<i>string₁ index string₂</i>	putinterval	-		replace substring of <i>string₁</i> starting at <i>index</i> by <i>string₂</i>
<i>string₁ string₂</i>	copy	<i>substring₂</i>		copy elements of <i>string₁</i> to initial substring of <i>string₂</i>
<i>string proc</i>	forall	-		execute <i>proc</i> for each element of string
<i>string seek</i>	anchorsearch	<i>post match true</i> or <i>string false</i>		determine if <i>seek</i> is initial substring of string
<i>string seek</i>	search	<i>post match pre true</i> or <i>string false</i>		search for <i>seek</i> in string
<i>string</i>	token	<i>post token true</i> or <i>false</i>		read token from start of <i>string</i>

Relational, Boolean, and Bitwise Operators

<i>any₁ any₂</i>	eq	<i>bool</i>		test equal
<i>any₁ any₂</i>	ne	<i>bool</i>		test not equal
<i>num₁ str₁ num₂ str₂</i>	ge	<i>bool</i>		test greater or equal
<i>num₁ str₁ num₂ str₂</i>	gt	<i>bool</i>		test greater than
<i>num₁ str₁ num₂ str₂</i>	le	<i>bool</i>		test less or equal
<i>num₁ str₁ num₂ str₂</i>	lt	<i>bool</i>		test less than
<i>bool₁ int₁ bool₂ int₂</i>	and	<i>bool₃ int₃</i>		logical bitwise and
<i>bool₁ int₁</i>	not	<i>bool₂ int₂</i>		logical bitwise not
<i>bool₁ int₁ bool₂ int₂</i>	or	<i>bool₃ int₃</i>		logical bitwise inclusive or
<i>bool₁ int₁ bool₂ int₂</i>	xor	<i>bool₃ int₃</i>		logical bitwise exclusive or
-	true	<i>true</i>		push boolean value <i>true</i>
-	false	<i>false</i>		push boolean value <i>false</i>
<i>int₁ shift</i>	bitshift	<i>int₂</i>		bitwise shift of <i>int₁</i> (positive is left)

Rectangle Operators (Display)

<i>x y width height</i>	rectfill	–	
<i>numarray</i>	rectfill	–	
<i>numstring</i>	rectfill	–	fill path consisting of one or more rectangles
<i>x y width height</i>	rectstroke	–	
<i>x y width height matrix</i>	rectstroke	–	
<i>numarray</i>	rectstroke	–	
<i>numarray matrix</i>	rectstroke	–	
<i>numstring</i>	rectstroke	–	
<i>numstring matrix</i>	rectstroke	–	stroke path consisting of one or more rectangles
<i>x y width height</i>	rectclip	–	
<i>numarray\ numstring</i>	rectclip	–	intersect inside of current clipping path with supplied path

Control Operators

<i>any</i>	exec	–	execute arbitrary object
<i>bool proc</i>	if	–	execute <i>proc</i> if <i>bool</i> is true
<i>bool proc₁ proc₂</i>	ifelse	–	execute <i>proc₁</i> if <i>bool</i> is true, <i>proc₂</i> if <i>bool</i> is false
<i>init incr limit proc</i>	for	–	execute <i>proc</i> with values from <i>init</i> by steps of <i>incr</i> to <i>limit</i>
<i>int proc</i>	repeat	–	execute <i>proc</i> <i>int</i> times
<i>proc</i>	loop	–	execute <i>proc</i> an indefinite number of times
	exit	–	exit innermost active loop
	stop	–	terminate stopped context
<i>any</i>	stopped	<i>bool</i>	establish context for catching stop
	countexecstack	<i>int</i>	count elements on exec stack
<i>array</i>	execstack	<i>subarray</i>	copy exec stack into <i>array</i>
	quit	–	terminate interpreter
	start	–	executed at interpreter startup

Type, Attribute, and Conversion Operators

<i>any</i>	type	<i>name</i>	return name identifying <i>any</i> 's type
<i>any</i>	cvlit	<i>any</i>	make object be literal

	<i>any</i>	cvx	<i>any</i>	make object be executable
	<i>any</i>	xcheck	<i>bool</i>	test executable attribute
	<i>array packedarray file string</i>	executeonly	<i>array packedarray file string</i>	reduce access to execute-only
	<i>array packedarray dict file string</i>	noaccess	<i>array packedarray dict file string</i>	disallow any access
	<i>array packedarray dict file string</i>	readonly	<i>array packedarray dict file string</i>	reduce access to read-only
	<i>array packedarray dict file string</i>	rcheck	<i>bool</i>	test read access
	<i>array packedarray dict file string</i>	wcheck	<i>bool</i>	test write access
	<i>numstring</i>	cvi	<i>int</i>	convert to integer
	<i>string</i>	cvn	<i>name</i>	convert to name
	<i>numstring</i>	cvr	<i>real</i>	convert to real
	<i>num radix string</i>	cvrs	<i>substring</i>	convert to string with <i>radix</i>
	<i>any string</i>	cvs	<i>substring</i>	convert to string

File Operators

<i>string₁ string₂</i>	file	<i>file</i>	open file identified by <i>string₁</i> with access <i>string₂</i>
<i>string</i>	deletefile	-	remove specified file from device
<i>string₁ string₂</i>	renamefile	-	change file name from <i>string₁</i> to <i>string₂</i>
<i>pattern proc scratch</i>	filenameforall	-	process each file whose name matches <i>pattern</i> with <i>proc</i>
<i>file</i>	closefile	-	close <i>file</i>
<i>file</i>	read	<i>int true</i> or <i>false</i>	read one character from <i>file</i>
<i>file int</i>	write	-	write one character to <i>file</i>
<i>file string</i>	readhexstring	<i>substring bool</i>	read hex from <i>file</i> into <i>string</i>
<i>file string</i>	writehexstring	-	write <i>string</i> to <i>file</i> as hex
<i>file string</i>	readstring	<i>substring bool</i>	read string from <i>file</i>
<i>file string</i>	writestring	-	write characters of <i>string</i> to <i>file</i>
<i>file string</i>	readline	<i>substring bool</i>	read line from <i>file</i> into <i>string</i>

<i>file</i>	token	<i>token true</i> or <i>false</i>		read token from <i>file</i>
<i>file int</i>	setfileposition	-		position next read or write in <i>file</i> to <i>int</i>
<i>file</i>	fileposition	<i>int</i>		return current position in already open file
<i>file</i>	bytesavailable	<i>int</i>		number of bytes available to read
	- flush	-		send buffered data to standard output file
<i>file</i>	flushfile	-		send buffered data or read to EOF
<i>file</i>	resetfile	-		discard buffered characters
<i>file</i>	status	<i>bool</i>		return status of file
<i>string</i>	status	<i>pages bytes referenced created true</i> or <i>false</i>		(Display) return status of file
<i>string</i>	run	-		execute contents of named file
	- currentfile	<i>file</i>		return file currently being executed
<i>string</i>	print	-		write characters of <i>string</i> to standard output file
	<i>any</i>	= -		write text representation of <i>any</i> to standard output file
	- <i>any</i> ₁ ... <i>any</i> _{<i>n</i>}	stack	- <i>any</i> ₁ ... <i>any</i> _{<i>n</i>}	print stack nondestructively using =
	<i>any</i>	== -		write syntactic representation of <i>any</i> to standard output file
	- <i>any</i> ₁ ... <i>any</i> _{<i>n</i>}	pstack	- <i>any</i> ₁ ... <i>any</i> _{<i>n</i>}	print stack nondestructively using ==
	- prompt	-		executed when ready for interactive input
<i>bool</i>	echo	-		turn on/off echoing

Structured Output Operators (Display)

<i>int</i>	setobjectformat	-		set format for object sequences written by printobject and writeobject
	- currentobjectformat	<i>int</i>		return current object format used by printobject and writeobject
<i>obj tag</i>	printobject	-		write binary object sequence to standard output file
<i>file obj tag</i>	writeobject	-		write binary object sequence to <i>file</i>

Virtual Memory Operators (Display)

	save	<i>save</i>		create VM snapshot
<i>save</i>	restore	–		restore VM snapshot
<i>dict key</i>	undef	–		remove <i>key</i> and associated value from dict
<i>bool</i>	setshared	–		set private or shared VM allocation mode
	–	currentshared	<i>bool</i>	return current value of VM allocation mode
<i>any</i>	scheck	<i>bool</i>		return whether <i>any</i> is sharable
<i>int</i>	vmreclaim	–		control garbage collection mode
	–	vmstatus	<i>level used maximum</i>	report VM status

Context Operators (Display)

<i>mark obj₁ ... obj_n proc</i>	fork	<i>context</i>		create new context within private VM of current context
<i>context</i>	join	<i>mark obj₁ ... obj_n</i>		when context ceases execution, push its operand stack onto current context's operand stack
<i>context</i>	detach	–		cause context to terminate when it's done executing
	–	currentcontext	<i>context</i>	return integer identifying current context
	–	lock	<i>lock</i>	create lock
	–	condition	<i>condition</i>	create condition object
<i>lock proc</i>	monitor	–		acquire lock, execute <i>proc</i> , and then release lock
<i>lock condition</i>	wait	–		release lock, wait for condition, and reacquire lock
<i>condition</i>	notify	–		resume execution of contexts waiting for condition
	–	yield	–	suspend current context until other contexts sharing same VM have executed

Miscellaneous Operators

<i>proc</i>	bind	<i>proc</i>		replace operator names in <i>proc</i> by operators
<i>index name</i>	defineusername	–		(Display) associate <i>index</i> with <i>name</i> in user name table

	- null <i>null</i>	push null on operand stack
	- usertime <i>int</i>	(Display) return PostScript interpreter execution time
	- realtime <i>int</i>	(Display) return value of clock that counts in real time
	- version <i>string</i>	interpreter version
	- nextrelease <i>string</i>	(NeXTSTEP) NeXTSTEP version information
<i>bool</i>	setwriteblock -	(NeXTSTEP) set whether Window Server blocks
	- currentwriteblock <i>bool</i>	(NeXTSTEP) return whether Server blocks
	- currentuser <i>uid gid</i>	(NeXTSTEP) return user id and group id of currently logged-in user
	- currentusage <i>ctime utime stime msgsend msgrcv nsignals nvcsw nivcsw</i>	(NeXTSTEP) report Window Server's resource usage
<i>soundname priority</i>	playsound -	(NeXTSTEP) play <i>soundname</i> at given priority level

Graphics State Operators

	- gsave -	save graphics state
	- grestore -	restore graphics state
	- grestoreall -	restore to bottommost graphics state
	- initgraphics -	(standard) reset graphics state parameters (NeXTSTEP) also reset alpha and instancing
<i>num</i>	setlinewidth -	set line width
	- currentlinewidth <i>num</i>	return current line width
<i>int</i>	setlinecap -	set shape of line ends for stroke (0=butt, 1=round, 2=square)
	- currentlinecap <i>int</i>	return current line cap
<i>int</i>	setlinejoin -	set shape of corners for stroke (0=miter, 1=round, 2=bevel)
	- currentlinejoin <i>int</i>	return current line join
<i>num</i>	setmiterlimit -	set miter length limit
	- currentmiterlimit <i>num</i>	return current miter limit

<i>array offset</i>	setdash	–	set dash pattern for stroking
	– currentdash	<i>array offset</i>	return current dash pattern
	<i>num</i>	setflat	– set flatness tolerance
	– currentflat	<i>num</i>	return current flatness
<i>patternname</i>	setpattern	–	(NeXTSTEP) set pattern for drawing
	<i>num</i>	setgray	– set color to gray value from 0 (black) to 1 (white)
	– currentgray	<i>num</i>	return current gray
<i>hue sat brt</i>	sethsbcolor	–	set color given hue, saturation, brightness
	– currenthsbcolor	<i>hue sat brt</i>	return current color hue, saturation, brightness
<i>red green blue</i>	setrgbcolor	–	set color given red, green, blue
	– currentrgbcolor	<i>red green blue</i>	return current color red, green, blue
<i>coverage</i>	setalpha	–	(NeXTSTEP) set current coverage
	– currentalpha	<i>coverage</i>	(NeXTSTEP) return current coverage setting
<i>cyan magenta yellow black</i>	setcmymkcolor	–	set current color parameter in graphics state
	– currentcmymkcolor	<i>cyan magenta yellow black</i>	return current color parameter in graphics state
<i>redproc greenproc blueproc grayproc</i>	setcolortransfer	–	set current transfer function parameters for specified colors
	– currentcolortransfer	<i>redproc greenproc blueproc grayproc</i>	return current transfer function parameters for specified colors
<i>proc</i>	setblackgeneration	–	set current black generation function parameter in graphics state
	– currentblackgeneration	<i>proc</i>	return current black generation function parameter in graphics state
<i>proc</i>	setundercolorremoval	–	set current undercolor removal function parameter in graphics state
	– currentundercolorremoval	<i>proc</i>	return current undercolor removal function parameter in graphics state

<i>redfrequency redangle redproc greenfrequency greenangle greenproc bluefrequency blueangle blueproc grayfrequency grayangle grayproc</i>	setcolorscreen -	set all twelve current halftone screen parameters in graphics state
	- currentcolorscreen <i>redfrequency redangle redproc greenfrequency greenangle greenproc bluefrequency blueangle blueproc grayfrequency grayangle grayproc</i>	return all twelve current halftone screen parameters in graphics state
<i>width height bits/sample matrix proc₀ [... proc_n] multiproc ncolors</i>	colorimage -	render sampled image with 1, 3, or 4 color values
<i>freq angle proc</i>	setscreen -	
<i>freq angle halftone</i>	setscreen -	set halftone screen
	- currentscreen <i>freq angle proc</i>	
	- currentscreen <i>60 0 halftone</i>	return current halftone screen or dictionary
<i>proc</i>	settransfer -	set gray transfer function
	- currenttransfer <i>proc</i>	return current transfer function

Graphics State Object Operators (Display)

	- gstate <i>gstate</i>	create graphics state object
<i>gstate</i>	setgstate -	replace current graphics state by value of <i>gstate</i>
<i>gstate</i>	currentgstate <i>gstate</i>	fill <i>gstate</i> with copy of current graphics state

Halftone Definition Operators (Display)

<i>dict</i>	sethalftone -	establish <i>dict</i> as current halftone dictionary
	- currenthalftone <i>dict</i>	return current halftone dictionary
<i>x y</i>	sethalftonephase -	(Display) set current halftone phase parameters
	- currenthalftonephase <i>x y</i>	(Display) return current halftone phase parameters

Coordinate System and Matrix Operators

$x\ y$	basetocurrent	$x'\ y'$	(NeXTSTEP) convert from base to current coordinate system
$x\ y$	basetoscreen	$x'\ y'$	(NeXTSTEP) convert from base to screen coordinate system
$x\ y$	currenttobase	$x'\ y'$	(NeXTSTEP) convert from current to base coordinate system
$x\ y$	currenttoscreen	$x'\ y'$	(NeXTSTEP) convert from current to screen coordinate system
$x\ y$	screeentobase	$x'\ y'$	(NeXTSTEP) convert from screen to base coordinate system
$x\ y$	screeentocurrent	$x'\ y'$	(NeXTSTEP) convert from screen to current coordinate system
	matrix	<i>matrix</i>	create identity matrix
	initmatrix	-	set CTM to device default
<i>matrix</i>	identmatrix	<i>matrix</i>	fill matrix with identity transform
<i>matrix</i>	defaultmatrix	<i>matrix</i>	fill matrix with device default matrix
<i>matrix</i>	currentmatrix	<i>matrix</i>	fill matrix with CTM
<i>matrix</i>	setmatrix	-	replace CTM by matrix
$t_x\ t_y$	translate	-	translate user space by (t_x, t_y)
$t_x\ t_y$	translate	<i>matrix</i>	define translation by (t_x, t_y)
$s_x\ s_y$	scale	-	scale user space by s_x and s_y
$s_x\ s_y$	scale	<i>matrix</i>	define scaling by s_x and s_y
<i>angle</i>	rotate	-	rotate user space by <i>angle</i> degrees
<i>angle</i>	rotate	<i>matrix</i>	define rotation by <i>angle</i> degrees
<i>matrix</i>	concat	-	replace CTM by matrix \times CTM
<i>matrix</i> ₁ <i>matrix</i> ₂ <i>matrix</i> ₃	concatmatrix	<i>matrix</i> ₃	fill <i>matrix</i> ₃ with <i>matrix</i> ₁ \times <i>matrix</i> ₂
$x\ y$	transform	$x'\ y'$	transform (x, y) by CTM
$x\ y$	transform	<i>matrix</i>	transform (x, y) by matrix
$dx\ dy$	dtransform	$dx'\ dy'$	transform distance (dx, dy) by CTM
$dx\ dy$	dtransform	<i>matrix</i>	transform distance (dx, dy) by matrix
$x'\ y'$	itransform	$x\ y$	inverse transform (x', y') by CTM
$x'\ y'$	itransform	<i>matrix</i>	inverse transform (x', y') by matrix
$dx'\ dy'$	idtransform	$dx\ dy$	inverse transform distance (dx', dy') by CTM
$dx'\ dy'$	idtransform	<i>matrix</i>	inverse transform distance (dx', dy') by matrix
<i>matrix</i> ₁ <i>matrix</i> ₂	invertmatrix	<i>matrix</i> ₂	fill <i>matrix</i> ₂ with inverse of <i>matrix</i> ₁

Path Construction Operators

	newpath	-	initialize current path to be empty
	currentpoint	$x\ y$	return current point coordinate
	moveto	$x\ y$	set current point to (x, y)
	rmoveto	$dx\ dy$	relative moveto
	lineto	$x\ y$	append straight line to (x, y)
	rlineto	$dx\ dy$	relative lineto
	arc	$x\ y\ r\ ang_1\ ang_2$	append counterclockwise arc
	arcn	$x\ y\ r\ ang_1\ ang_2$	append clockwise arc
	arcto	$x_1\ y_1\ x_2\ y_2\ r\ xt_1\ yt_1\ xt_2\ yt_2$	append tangent arc
	curveto	$x_1\ y_1\ x_2\ y_2\ x_3\ y_3$	append Bezier cubic section
	rcurveto	$dx_1\ dy_1\ dx_2\ dy_2\ dx_3\ dy_3$	relative curveto
	closepath	-	connect subpath back to its starting point
	flattenpath	-	convert curves to sequences of straight lines
	reversepath	-	reverse direction of current path
	strokepath	-	compute outline of stroked path
	charpath	$string\ bool$	append character outline to current path
	clippath	-	set current path to clipping path
	pathbbox	$ll_x\ ll_y\ ur_x\ ur_y$	return bounding box of current path
	pathforall	$move\ line\ curve\ close$	enumerate current path
	initclip	-	set clipping path to device default
	clip	-	establish new clipping path
	eoclip	-	clip using even-odd inside rule

User Path Operators (Display)

	setbbox	$ll_x\ ll_y\ ur_x\ ur_y$	establish bounding box for current path
	arct	$x_1\ y_1\ x_2\ y_2\ r$	append arc of circle to current path
	uappend	$userpath$	append <i>userpath</i> to current path
	upath	$bool\ userpath$	create <i>userpath</i> as copy of current path

<i>userpath</i>	ufill	–	fill <i>userpath</i> as if by using fill operator	
<i>userpath</i>	ueofill	–	fill <i>userpath</i> as if by using eofill operator	
<i>userpath</i>	ustroke	–		
<i>userpath matrix</i>	ustroke	–	stroke <i>userpath</i> as if by using stroke operator	
<i>userpath</i>	ustrokepath	–		
<i>userpath matrix</i>	ustrokepath	–	replace current path with <i>userpath</i> and stroke result	
	–	ucache	–	store enclosing user path if not already stored
	–	ucachestatus	<i>mark bsize bmax rsize rmax blimit</i>	report status of user path cache
<i>mark blimit</i>	setucacheparams	–	set user path cache parameters	

View Clip Operators (Display)

	–	viewclip	–	replace view clipping path with copy of current path
	–	eoviewclip	–	replace view clipping path with current path using even-odd inside rule
<i>x y width height</i>		rectviewclip	–	
<i>numarray numstring</i>		rectviewclip	–	replace view clipping path with specified path
	–	viewclippath	–	replace path with copy of current view clipping path
	–	initviewclip	–	replace view clipping path with one equal to imageable area

Painting Operators

	–	erasepage	–	(NeXTSTEP) erase entire window to opaque white (standard) paint current page white
	–	fill	–	fill current path with current color
	–	eofill	–	fill using even-odd rule
	–	stroke	–	draw line along current path
<i>width height bits/sample matrix proc</i>		image	–	render sampled image onto current page
<i>width height invert matrix proc</i>		imagemask	–	render mask onto current page

Window System Support Operators (Display)

<code>- wtranslation</code>	<code>x y</code>				return translation from window origin to device space origin
	<code>x y</code>	<code>infill</code>	<code>bool</code>		
	<code>userpath</code>	<code>infill</code>	<code>bool</code>		return <i>true</i> if pixel at (x, y) (or any pixels in <i>userpath</i>) would be painted by fill of current path
	<code>x y</code>	<code>ineofill</code>	<code>bool</code>		
	<code>userpath</code>	<code>ineofill</code>	<code>bool</code>		return <i>true</i> if pixel at (x, y) (or any pixels in <i>userpath</i>) would be painted by eofill of current path
	<code>x y userpath</code>	<code>inufill</code>	<code>bool</code>		
	<code>userpath₁ userpath₂</code>	<code>inufill</code>	<code>bool</code>		return <i>true</i> if pixel at (x, y) (or any pixels in <i>userpath₁</i>) would be painted by ufill of current path
	<code>x y userpath</code>	<code>inueofill</code>	<code>bool</code>		
	<code>userpath₁ userpath₂</code>	<code>inueofill</code>	<code>bool</code>		return <i>true</i> if pixel at (x, y) (or any pixels in <i>userpath₁</i>) would be painted by ueofill of current path
	<code>x y</code>	<code>instroke</code>	<code>bool</code>		
	<code>userpath</code>	<code>instroke</code>	<code>bool</code>		return <i>true</i> if pixel at (x, y) (or any pixels in <i>userpath</i>) would be painted by stroke of current path
	<code>x y userpath</code>	<code>inustroke</code>	<code>bool</code>		
	<code>x y userpath matrix</code>	<code>inustroke</code>	<code>bool</code>		
	<code>userpath₁ userpath₂</code>	<code>inustroke</code>	<code>bool</code>		
	<code>userpath₁ userpath₂ matrix</code>	<code>inustroke</code>	<code>bool</code>		return <i>true</i> if pixel at (x, y) (or any pixels in <i>userpath₁</i>) would be painted by ustroke of current path
<code>- deviceinfo</code>	<code>dict</code>				return <i>dict</i> containing static information about current device

Device Setup and Output Operators

<code>proc window</code>	<code>setshowpageprocedure</code>	<code>-</code>		set the procedure that's executed during showpage
<code>window</code>	<code>currentshowpageprocedure</code>	<code>proc</code>		return the procedure that's executed during showpage
<code>-</code>	<code>showpage</code>	<code>-</code>		output and reset current page

	- copypage -	output current page
	- nulldevice -	install no-output device
<i>width height bbox matrix</i> <i>hostname portname pixelencoding</i>	machportdevice -	(NeXTSTEP) set up PostScript device for generic rendering service

Scan Conversion Operators (Display)

<i>bool</i>	setstrokeadjust -	turn automatic stroke adjustment on or off
	- currentstrokeadjust <i>bool</i>	return current state of automatic stroke adjustment

Character and Font Operators

<i>key font</i>	definefont <i>font</i>	register <i>font</i> as font dictionary
<i>key</i>	undefinefont -	(Display) remove <i>key</i> from FontDirectory dictionary
<i>key</i>	findfont <i>font</i>	return font dict identified by <i>key</i>
<i>font scale</i>	scalefont <i>font'</i>	scale <i>font</i> by <i>scale</i> to produce new <i>font'</i>
<i>font matrix</i>	makefont <i>font'</i>	transform <i>font</i> by <i>matrix</i> to produce new <i>font'</i>
<i>font</i>	setfont -	set font dictionary
	- currentfont <i>font</i>	return current font dictionary
<i>string</i>	show -	print characters of <i>string</i> on page
<i>a_x a_y string</i>	ashow -	add (<i>a_x</i> , <i>a_y</i>) to width of each char while showing <i>string</i>
<i>text numarray numstring</i>	xyshow -	(Display) print characters according to x, y displacements in <i>numarray</i> or <i>numstring</i>
<i>text numarray numstring</i>	xshow -	(Display) print characters according to x displacements in <i>numarray</i> or <i>numstring</i>
<i>text numarray numstring</i>	yshow -	(Display) print characters according to y displacements in <i>numarray</i> or <i>numstring</i>
<i>c_x c_y char string</i>	widthshow -	add (<i>c_x</i> , <i>c_y</i>) to width of <i>char</i> while showing <i>string</i>
<i>c_x c_y char a_x a_y string</i>	awidthshow -	combine effects of ashow and widthshow
<i>proc string</i>	kshow -	execute <i>proc</i> between characters shown from <i>string</i>

<i>string</i>	stringwidth	<i>w_x w_y</i>		width of <i>string</i> in current font
	-	FontDirectory	<i>dict</i>	dictionary of font dictionaries
	-	SharedFontDirectory	<i>dict</i>	(Display) dictionary of font dictionaries
	-	NextStepEncoding	<i>array</i>	NeXTSTEP font encoding vector
	-	StandardEncoding	<i>array</i>	standard font encoding vector
<i>key scale\matrix</i>		selectfont	-	(Display) establish font specified by <i>key</i> as current font

Font Cache Operators

	-	cachestatus	<i>bsize bmax msize mmax csize cmax blimit</i>	return cache status and parameters
<i>w_x w_y ll_x ll_y ur_x ur_y</i>		setcachedevice	-	declare cached character metrics
<i>w_x w_y</i>		setcharwidth	-	declare uncached character metrics
<i>num</i>		setcachelimit	-	set max bytes in cached character
<i>mark size lower upper</i>		setcacheparams	-	set character cache parameters
	-	currentcacheparams	<i>mark lower upper</i>	return current font cache parameters

User Object Encoding Operators (Display)

<i>index any</i>		defineuserobject	-	associate <i>index</i> with an object in the user object array
<i>index</i>		undefineuserobject	-	remove associate between <i>index</i> and the object it referred to
<i>index</i>		execuserobject	-	execute object referred to by <i>index</i>

Errors

dictfull	no more room in dictionary
dictstackoverflow	too many begins
dictstackunderflow	too many ends
execstackoverflow	exec nesting too deep
handleerror	called to report error information
interrupt	external interrupt request (e.g., Control-C)

invalidaccess	attempt to violate access attribute
invalidcontext	invalid use of context synchronization facilities
invalidexit	exit not in loop
invalidfileaccess	unacceptable access string
invalidfont	invalid font name or dict
invalidrestore	improper restore
invalidid	(Display) invalid identifying number as operand
ioerror	input/output error occurred
limitcheck	implementation limit exceeded
nocurrentpoint	current point is undefined
rangecheck	operand out of bounds
stackoverflow	operand stack overflow
stackunderflow	operand stack underflow
syntaxerror	syntax error in PS program
timeout	time limit exceeded
typecheck	operand of wrong type
undefined	name not known
undefinedfilename	file not found
undefinedresult	over/underflow or meaningless result
unmatchedmark	expected mark not on stack
unregistered	internal error
VMerror	VM exhausted

Single-Operator Functions

The single-operator functions listed here begin with the prefix “PS.” For every single-operator function with a “PS” prefix, there’s a corresponding single-operator function with a “DPS” prefix. The PS and DPS functions are identical except that DPS functions take an additional (first) argument that represents the PostScript execution context. To conserve space, only the single-operator functions prefixed with “PS” are listed here. (See “Suggested Reading” for references to documentation about Display PostScript.)

Besides using the standard C types, these single-operator functions use **boolean** and **userobject**. A **boolean** variable is an **int** having either a zero or a nonzero value. The zero value is equivalent to the PostScript value *false*, and the nonzero value is equivalent to the PostScript value *true*. The **userobject** type is an **int** that refers to the value returned by **DPSDefineUserObject()**.

```
void    PSabs(void)
void    PSadd(void)
void    PSadjustcursor(float deltaX, float deltaY)†
void    PSaload(void)
void    PSalphaimage(void)†
void    PSanchorsearch(boolean *pflag)
void    PSand(void)
void    PSarc(float x, float y, float radius, float angle1, float angle2)
void    PSarcn(float x, float y, float radius, float angle1, float angle2)
void    PSarct(float x1, float y1, float x2, float y2, float radius)
void    PSarcto(float x, float y, float x2, float y2, float radius, float *pxt1, float *pyt1, float *pxt2,
               float *pyt2)
void    PSarray(int length)
void    PSashow(float x, float y, char *string)
void    PSastore(void)
void    PSatan(void)
void    PSawidthshow(float x, float y, int c, float ax, float ay, char *string)
void    PSbasetocurrent(float x, float y, float *px, float *py)†
void    PSbasetoscreen(float x, float y, float *px, float *py)†
void    PSbegin(void)
void    PSbind(void)
```

void **PSbitshift**(int *shift*)
 void **PSbuttondown**(boolean *pflag*)[†]
 void **PSbytesavailable**(int *pcount*)
 void **PScachestatus**(int *pysize*, int *pbmax*, int *pmsize*)
 void **PSceiling**(void)
 void **PScharpath**(char *string*, boolean *flag*)
 void **PSclear**(void)
 void **PScleardictstack**(void)
 void **PScleartomark**(void)
 void **PScleartrackingrect**(int *tRectNum*, userobject *gstate*)[†]
 void **PSclip**(void)
 void **PSclippath**(void)
 void **PSclosefile**(void)
 void **PSclosepath**(void)
 void **PScolorimage**(void)
 void **PScomposite**(float *x*, float *y*, float *width*, float *height*, userobject *srcGstate*, float *dest_x*,
 float *dest_y*, int *op*)[†]
 void **PScompositerect**(float *dest_x*, float *dest_y*, float *width*, float *height*, int *op*)[†]
 void **PSconcat**(float *m*[6])
 void **PSconcatmatrix**(void)
 void **PScondition**(void)
 void **PScopy**(int *n*)
 void **PScopypage**(void)
Warning: This function has no effect in NeXTSTEP.
 void **PScos**(void)
 void **PScount**(int *pn*)
 void **PScountdictstack**(int *plength*)
 void **PScountexecstack**(int *pcount*)
 void **PScountframebuffers**(int *pcount*)[†]
 void **PScountscreenlist**(int *context*, int *pcount*)[†]
 void **PScounttomark**(int *pn*)
 void **PScountwindowlist**(int *context*, int *pcount*)[†]
 void **PScshow**(char *string*)

void **PScurrentactiveapp**(int **pcontext*)
Warning: Don't use this function if you're using the Application Kit.

void **PScurrentalpha**(float **pcoverage*)[†]

void **PScurrentblackgeneration**(void)

void **PScurrentcacheparams**(void)

void **PScurrentcmykcolor**(float **pc*, float **pm*, float **py*, float **pk*)

void **PScurrentcolor**(void)

void **PScurrentcolorrendering**(void)

void **PScurrentcolorscreen**(void)

void **PScurrentcolorspace**(void)

void **PScurrentcolortransfer**(void)

void **PScurrentcontext**(int **pcontext*)

void **PScurrentdash**(void)

void **PScurrentdefaultdepthlimit**(int **plimit*)[†]
Warning: Don't use this function if you're using the Application Kit.

void **PScurrentdeviceinfo**(userobject *window*, int **pMinBPS*, int **pMaxBPS*, int **pColor*)[†]

void **PScurrentdevparams**(char **device*)

void **PScurrentdict**(void)

void **PScurrenteventmask**(userobject *window*, int **pmask*)
Warning: Don't use this function if you're using the Application Kit.

void **PScurrentfile**(void)

void **PScurrentflat**(float **pflatness*)

void **PScurrentfont**(void)

void **PScurrentframebuffertransfer**(void)

void **PScurrentglobal**(int **b*)

void **PScurrentgray**(float **pgray*)

void **PScurrentgstate**(userobject *gstate*)

void **PScurrenthalftone**(void)

void **PScurrenthalftonephase**(float **px*, float **py*)

void **PScurrenthsbcolor**(float **ph*, float **ps*, float **pb*)

void **PScurrentlinecap**(int **plinecap*)

void **PScurrentlinejoin**(int **plinejoin*)

void **PScurrentlinewidth**(float **pwidth*)

void **PScurrentmatrix**(void)

void **PScurrentmiterlimit**(float *plimit)

void **PScurrentmouse**(userobject *window*, float *px, float *py)
Warning: Don't use this function if you're using the Application Kit.

void **PScurrentobjectformat**(int *pcode)

void **PScurrentoverprint**(int *b)

void **PScurrentowner**(userobject *window*, int *pcontext)[†]

void **PScurrentpacking**(boolean *pflag)

void **PScurrentpagedevice**(void)

void **PScurrentpoint**(float *px, float *py)

void **PScurrentrgbcolor**(float *pr, float *pg, float *pb)

void **PScurrentrusage**(float *pnow, float *puTime, float *psTime, int *pmsgSend,
int *pmsgReceive, int *pnSignals, int *pnVCSw, int *pnIvCSw)[†]

void **PScurrentscreen**(void)

void **PScurrentshared**(boolean *pflag)

void **PScurrentshowpageprocedure**(void)

void **PScurrentstrokeadjust**(boolean *pflag)

void **PScurrentsystemparams**(void)

void **PScurrenttobase**(float x, float y, float *px, float *py)[†]

void **PScurrenttoscreen**(float x, float y, float *px, float *py)[†]

void **PScurrenttransfer**(void)

void **PScurrentundercolorremoval**(void)

void **PScurrentuser**(int *puid, int *pgid)[†]

void **PScurrentuserparams**(void)

void **PScurrentwaitcursorenabled**(boolean *pflag)[†]

void **PScurrentwindow**(int *pnum)[†]

void **PScurrentwindowalpha**(userobject *window*, int *palpha)[†]

void **PScurrentwindowbounds**(userobject *window*, float *px, float *py, float *pwidth,
float *pheight)[†]
Warning: Don't use this function if you're using the Application Kit.

void **PScurrentwindowdepth**(userobject *window*, int *pdepth)[†]
Warning: Don't use this function if you're using the Application Kit.

void **PScurrentwindowdepthlimit**(userobject *window*, int *plimit)[†]
Warning: Don't use this function if you're using the Application Kit.

void **PScurrentwindowdict**(userobject *window*)
Warning: Don't use this function if you're using the Application Kit.

void **PScurrentwindowlevel**(userobject *window*, int **plevel*)[†]
 void **PScurrentwriteblock**(int **pflag*)[†]
 void **PScurveto**(float *x₁*, float *y₁*, float *x₂*, float *y₂*, float *x₃*, float *y₃*)
 void **PScvi**(void)
 void **PScvlit**(void)
 void **PScvn**(void)
 void **PScvr**(void)
 void **PScvrs**(void)
 void **PScvs**(void)
 void **PScvx**(void)
 void **PSdef**(void)
 void **PSdefaultmatrix**(void)
 void **PSdefinefont**(void)
 void **PSdefineresource**(char **category*)
 void **PSdefineusername**(int *index*, char **name*)
 void **PSdefineuserobject**(void)
Warning: Use **DPSDefineUserObject()** instead.
 void **PSdeletefile**(char **filename*)
 void **PSdetach**(void)
 void **PSdeviceinfo**(void)
 void **PSdict**(int *length*)
 void **PSdictstack**(void)
 void **PSdissolve**(float *src_x*, float *src_y*, float *width*, float *height*, userobject *srcGstate*, float *dest_x*, float *dest_y*, float *delta*)[†]
 void **PSdiv**(void)
 void **PSdtransform**(float *x*, float *y*, float **px*, float **py*)
 void **PSdumpwindow**(int *level*, userobject *window*)
Warning: Don't use this function if you're using the Application Kit.
 void **PSdumpwindows**(int *level*, userobject *context*)
Warning: Don't use this function if you're using the Application Kit.
 void **PSdup**(void)
 void **PSecho**(boolean *flag*)
 void **PSend**(void)
 void **PSeoclip**(void)

void **PSeofill**(void)
 void **PSeoviewclip**(void)
 void **PSeq**(void)
 void **PSequals**(void)
 void **PSequalequals**(void)
 void **PSerasepage**(void)
Warning: This function is different in NeXTSTEP.
 void **PSerrordict**(void)
 void **PSexch**(void)
 void **PSexec**(void)
 void **PSexecform**(void)
 void **PSexecstack**(void)
 void **PSexecuteonly**(void)
 void **PSexecuserobject**(int *index*)
 void **PSexit**(void)
 void **PSexp**(void)
 void **PSfalse**(void)
 void **PSfile**(char **name*, char **access*)
 void **PSfilenameforall**(void)
 void **PSfileposition**(int **ppos*)
 void **PSfill**(void)
 void **PSfilter**(void)
 void **PSfindencoding**(char **key*)
 void **PSfindfont**(char **name*)
 void **PSfindresource**(char **key*, char **category*)
 void **PSfindwindow**(float *x*, float *y*, int *place*, userobject *otherWindow*, float **px*, float **py*,
 int **pwinFound*, boolean **pdidFind*)[†]
 void **PSflattenpath**(void)
 void **PSfloor**(void)
 void **PSflush**(void)
 void **PSflushfile**(void)
 void **PSflushgraphics**(void)[†]
Warning: Don't use this function if you're using the Application Kit.
 void **PSFontDirectory**(void)

void **PSfor**(void)
 void **PSforall**(void)
 void **PSfork**(void)
 void **PSframebuffer**(int *index*, int *nameLength*, char *name*[], int **pslot*, int **punit*,
 int **pROMid*, int **px*, int **py*, int **pwidth*, int **pheight*, int **pdepth*)[†]
 void **PSfrontwindow**(int **pnum*)
Warning: Don't use this function if you're using the Application Kit.
 void **PSgcheck**(int **b*)
 void **PSge**(void)
 void **PSget**(void)
 void **PSgetboolean**(boolean **pflag*)
 void **PSgetchararray**(int *size*, char *string*[])
 void **PSgetfloat**(float **pvalue*)
 void **PSgetfloatarray**(int *size*, float *array*[])
 void **PSgetint**(int **pvalue*)
 void **PSgetintarray**(int *size*, float *array*[])
 void **PSgetinterval**(void)
 void **PSgetstring**(char **string*)
 void **PSGlobalFontDirectory**(void)
 void **PSglobaldict**(void)
 void **PSglyphshow**(char **name*)
 void **PSgrestore**(void)
 void **PSgrestoreall**(void)
 void **PSgsave**(void)
 void **PSgstate**(void)
 void **PSgt**(void)
 void **PShidecursor**(void)[†]
 void **PShideinstance**(float *x*, float *y*, float *width*, float *height*)[†]
 void **PSidentmatrix**(void)
 void **PSidiv**(void)
 void **PSidtransform**(float *x*, float *y*, float **px*, float **py*)
 void **PSif**(void)
 void **PSifelse**(void)

void **PSimage**(void)
 void **PSimagemask**(void)
 void **PSindex**(int *n*)
 void **PSsineofill**(float *x*, float *y*, boolean **pflag*)
 void **PSsinfill**(float *x*, float *y*, boolean **pflag*)
 void **PSinitclip**(void)
 void **PSiniteventtimes**(void)
 void **PSinitgraphics**(void)
Warning: This function is different in NeXTSTEP.
 void **PSinitmatrix**(void)
 void **PSinitviewclip**(void)
 void **PSinstroke**(float *x*, float *y*, boolean **pflag*)
 void **PSinueofill**(float *x*, float *y*, char *nums*[*n*], int *n*, char *ops*[*l*], int *l*, boolean **pflag*)
 void **PSinufill**(float *x*, float *y*, char *nums*[*n*], int *n*, char *ops*[*l*], int *l*, boolean **pflag*)
 void **PSinustroke**(float *x*, float *y*, char *nums*[*n*], int *n*, char *ops*[*l*], int *l*, boolean **pflag*)
 void **PSinvertmatrix**(void)
 void **PSISOLatin1Encoding**(void)
 void **PSitransform**(float *x*, float *y*, float **px*, float **py*)
 void **PSjoin**(void)
 void **PSknown**(boolean **pflag*)
 void **PSkshow**(char **string*)
 void **PSlanguagelevel**(int **n*)
 void **PSle**(void)
 void **PSleftbracket**(void)
 void **PSleftleft**(void)
 void **PSlength**(int **pn*)
 void **PSlineto**(float *x*, float *y*)
 void **PSln**(void)
 void **PSload**(void)
 void **PSlock**(void)
 void **PSlog**(void)
 void **PSloop**(void)
 void **PSlt**(void)

void **PSmachportdevice**(int *w*, int *h*, int *bbox*[], int *bboxSize*, float *matrix*[], char **phost*,
 char **pport*, char **ppixelDict*)[†]

void **PSmakefont**(void)

void **PSmakepattern**(void)

void **PSmark**(void)

void **PSmatrix**(void)

void **PSmaxlength**(int **plength*)

void **PSmod**(void)

void **PSmonitor**(void)

void **PSmoveto**(float *x*, float *y*)

void **PSmovewindow**(float *x*, float *y*, userobject *window*)
Warning: Don't use this function if you're using the Application Kit.

void **PSmul**(void)

void **PSne**(void)

void **PSneg**(void)

void **PSnewinstance**(void)[†]

void **PSnewpath**(void)

void **PSnextrelease**(int *size*, char *string*[])[†]

void **PSnoaccess**(void)

void **PSnot**(void)

void **PSnotify**(void)

void **PSnull**(void)

void **PSnulldevice**(void)

void **PSobscurecursor**(void)[†]

void **PSor**(void)

void **PSorderwindow**(int *place*, userobject *otherWindow*, userobject *window*)
Warning: Don't use this function if you're using the Application Kit.

void **PSosname**(int *size*, char *string*[])[†]

void **PSostype**(int **ptype*)[†]

void **PSpackedarray**(void)

void **PSpathbbox**(float **pll_x*, float **pll_y*, float **pur_x*, float **pur_y*)

void **PSpathforall**(void)

void **PSplacewindow**(float *x*, float *y*, float *width*, float *height*, userobject *window*)
Warning: Don't use this function if you're using the Application Kit.

void **PSplaysound**(char *name, int priority)[†]
 void **PSpop**(void)
 void **PSposteventbycontext**(int type, float x, float y, int time, int flags, int window, int subtype,
 int data1, int data2, int context, boolean *psuccess)[†]
 void **PSprint**(void)
 void **PSprinteventtimes**(void)
 void **PSprintobject**(int code)
 void **PSproduct**(void)
 void **PSprompt**(void)
 void **PSpstack**(void)
 void **PSput**(void)
 void **PSputinterval**(void)
 void **PSquit**(void)
 void **PSrand**(void)
 void **PSrcheck**(boolean *pflag)
 void **PSrcurveto**(float x, float y, float x₂, float y₂, float x₃, float y₃)
 void **PSread**(boolean *pflag)
 void **PSreadhexstring**(boolean *pflag)
 void **PSreadimage**(void)[†]
 void **PSreadline**(boolean *pflag)
 void **PSreadonly**(void)
 void **PSreadstring**(boolean *pflag)
 void **PSrealtime**(int *pi)
 void **PSrectclip**(float x, float y, float width, float height)
 void **PSrectfill**(float x, float y, float width, float height)
 void **PSrectstroke**(float x, float y, float width, float height)
 void **PSrectviewclip**(float x, float y, float width, float height)
 void **PSrenamefile**(char *old, char *new)
 void **PSrepeat**(void)
 void **PSresetfile**(void)
 void **PSresourceforall**(char *category)
 void **PSresourcestatus**(char *key, char *category, int *b)
 void **PSrestore**(void)

void **PSrevealcursor**(void)[†]
 void **PSreversepath**(void)
 void **PSrevision**(int **n*)
 void **PSrightbracket**(void)
 void **PSrightbuttondown**(boolean **pflag*)[†]
 void **PSrightright**(void)
 void **PSrightstilldown**(int *eventNum*, boolean **pflag*)[†]
 void **PSrlineto**(float *x*, float *y*)
 void **PSrmoveto**(float *x*, float *y*)
 void **PSrootfont**(void)
 void **PSroll**(int *n*, int *j*)
 void **PSrotate**(float *angle*)
 void **PSround**(void)
 void **PSrrand**(void)
 void **PSrun**(char **name*)
 void **PSsave**(void)
 void **PSscale**(float *sx*, float *sy*)
 void **PSscalefont**(float *size*)
 void **PSscheck**(boolean **pflag*)
 void **PSscreenlist**(int *context*, int *count*, int *windows*[])[†]
 void **PSscreentobase**(float *x*, float *y*, float **px*, float **py*)[†]
 void **PSscreentocurrent**(float *x*, float *y*, float **px*, float **py*)[†]
 void **PSsearch**(boolean **pflag*)
 void **PSselectfont**(char **name*, float *scale*)
 void **PSsendboolean**(boolean *flag*)
 void **PSsendchararray**(char *string*[], int *size*)
 void **PSsendfloat**(float *value*)
 void **PSsendfloatarray**(float *array*[], int *size*)
 void **PSsendint**(int *value*)
 void **PSsendintarray**(int *array*[], int *size*)
 void **PSsendstring**(char **string*)
 void **PSserialnumber**(int **n*)

void **PSsetactiveapp**(int *context*)
Warning: Don't use this function if you're using the Application Kit.

void **PSsetalpha**(float *coverage*)[†]

void **PSsetautofill**(boolean *flag*, userobject *window*)[†]

void **PSsetbbox**(float *ll_x*, float *ll_y*, float *ur_x*, float *ur_y*)

void **PSsetblackgeneration**(void)

void **PSsetcachedevice**(float *w_x*, float *w_y*, float *ll_x*, float *ll_y*, float *ur_x*, float *ur_y*)

void **PSsetcachelimit**(float *num*)

void **PSsetcacheparams**(void)

void **PSsetcharwidth**(float *w_x*, float *w_y*)

void **PSsetcmkcolor**(float *c*, float *m*, float *y*, float *k*)

void **PSsetcolor**(void)

void **PSsetcolorrendering**(void)

void **PSsetcolorscreen**(void)

void **PSsetcolorspace**(void)

void **PSsetcolortransfer**(void)

void **PSsetcursor**(float *x*, float *y*, float *mx*, float *my*)[†]

void **PSsetdash**(float *pattern*[], int *size*, float *offset*)

void **PSsetdefaultdepthlimit**(int *limit*)[†]
Warning: Don't use this function if you're using the Application Kit.

void **PSsetdevparams**(void)

void **PSseteventmask**(int *mask*, userobject *window*)
Warning: Don't use this function if you're using the Application Kit.

void **PSsetexposurecolor**(void)[†]

void **PSsetfileposition**(int *pos*)

void **PSsetflat**(float *flatness*)

void **PSsetflushexposures**(boolean *flag*)[†]

void **PSsetfont**(userobject *font*)

void **PSsetframebuffertransfer**(void)

void **PSsetglobal**(int *b*)

void **PSsetgray**(float *num*)

void **PSsetgstate**(userobject *gstate*)

void **PSsethalftone**(void)

void **PSsethalftonephase**(float *x*, float *y*)

void **PSsethsbcolor**(float *hue*, float *sat*, float *brt*)
 void **PSsetinstance**(boolean *flag*)[†]
 void **PSsetlinecap**(int *linecap*)
 void **PSsetlinejoin**(int *linejoin*)
 void **PSsetlinewidth**(float *width*)
 void **PSsetmatrix**(void)
 void **PSsetmiterlimit**(float *limit*)
 void **PSsetmouse**(float *x*, float *y*)[†]
 void **PSsetobjectformat**(int *code*)
 void **PSsetoverprint**(int *b*)
 void **PSsetowner**(userobject *context*, userobject *window*)[†]
 void **PSsetpacking**(boolean *flag*)
 void **PSsetpagedevice**(void)
 void **PSsetpattern**(int *patternDict*)
 void **PSsetrgbcolor**(float *red*, float *green*, float *blue*)
 void **PSsetscreen**(void)
 void **PSsetsendexposed**(boolean *flag*, userobject *window*)[†]
Warning: Don't use this function if you're using the Application Kit.
 void **PSsetshared**(boolean *flag*)
 void **PSsetshowpageprocedure**(int *win*)[†]
Warning: Don't use this function if you're using the Application Kit.
 void **PSsetstrokeadjust**(boolean *flag*)
 void **PSsetsystemparams**(void)
 void **PSsettrackingrect**(float *x*, float *y*, float *width*, float *height*, boolean *leftFlag*,
 boolean *rightFlag*, boolean *inside*, int *userData*, int *trectNum*,
 userobject *gstate*)[†]
 void **PSsettransfer**(void)
 void **PSsetucacheparams**(void)
 void **PSsetundercolorremoval**(void)
 void **PSsetuserparams**(void)
 void **PSsetvmthreshold**(int *i*)
 void **PSsetwaitcursorenabled**(boolean *flag*)[†]
 void **PSsetwindowdepthlimit**(int *limit*, userobject *window*)[†]
Warning: Don't use this function if you're using the Application Kit.

void **PSsetwindowdict**(userobject *window*)
Warning: Don't use this function if you're using the Application Kit.

void **PSsetwindowlevel**(int *level*, userobject *window*)[†]

void **PSsetwindowtype**(int *type*, userobject *window*)[†]
Warning: Don't use this function if you're using the Application Kit.

void **PSsetwriteblock**(int *flag*)[†]

void **PSshreddict**(void)

void **PSSharedFontDirectory**(void)

void **PSshow**(char **string*)

void **PSshowcursor**(void)[†]

void **PSshowpage**(void)
Warning: This function is different in NeXTSTEP.

void **PSsin**(void)

void **PSsizeimage**(float *x*, float *y*, float *width*, float *height*, int **ppixelsWide*, int **ppixelsHigh*,
int **pbitsPerSample*, float *matrix*[], boolean **pmultiProc*,
int **pnColors*)[†]

void **PSsqrt**(void)

void **PSsrand**(void)

void **PSstack**(void)

void **PSStandardEncoding**(void)

void **PSstart**(void)

void **PSstartjob**(int *b*, char **password*)

void **PSstatus**(boolean **pflag*)

void **PSstatusdict**(void)

void **PSstilldown**(int *eventNum*, boolean **pflag*)[†]

void **PSstop**(void)

void **PSstopped**(void)

void **PSstore**(void)

void **PSstring**(int *length*)

void **PSstringwidth**(char **string*, float **px*, float **py*)

void **PSstroke**(void)

void **PSstrokepath**(void)

void **PSsub**(void)

void **PSsystemdict**(void)

void **PStermwindow**(userobject *window*)
Warning: Don't use this function if you're using the Application Kit.

void **PStoken**(boolean **pflag*)

void **PStransform**(float *x*, float *y*, float **px*, float **py*)

void **PStranslate**(float *x*, float *y*)

void **PStrue**(void)

void **PStruncate**(void)

void **PStype**(void)

void **PSuappend**(char *nums*[], int *n*, char *ops*[], int *l*)

void **PSucache**(void)

void **PSucachestatus**(void)

void **PSueofill**(char *nums*[], int *n*, char *ops*[], int *l*)

void **PSufill**(char *nums*[], int *n*, char *ops*[], int *l*)

void **PSundef**(char **name*)

void **PSundefinefont**(char **name*)

void **PSundefineresource**(char **key*, char **category*)

void **PSundefineuserobject**(int *index*)

void **PSupath**(boolean *flag*)

void **PSuserdict**(void)

void **PSuserobject**(void)

void **PSusertime**(int **pmillisecs*)

void **PSustroke**(char *nums*[], int *n*, char *ops*[], int *l*)

void **PSustrokepath**(char *nums*[], int *n*, char *ops*[], int *l*)

void **PSversion**(int *bufsize*, char *buf*[])

void **PSviewclip**(void)

void **PSviewclippath**(void)

void **PSvmreclaim**(int *code*)

void **PSvmstatus**(int **plevel*, int **pused*, int **pmax*)

void **PSwait**(void)

void **PSwcheck**(boolean **pflag*)

void **PSwhere**(boolean **pflag*)

void **PSwidthshow**(float *x*, float *y*, int *c*, char **string*)

void **PSwindow**(float *x*, float *y*, float *width*, float *height*, int *type*, int **pwindow*)
Warning: Don't use this function if you're using the Application Kit.

void **PSwindowdevice**(userobject *window*)[†]

void **PSwindowdeviceround**(userobject *window*)[†]

void **PSwindowlist**(int *context*, int *count*, int *windows*[])[†]

void **PSwrite**(void)

void **PSwritehexstring**(void)

void **PSwriteobject**(int *code*)

void **PSwritestring**(void)

void **PSwtranslation**(float **px*, float **py*)

void **PSxcheck**(boolean **pflag*)

void **PSxor**(void)

void **PSxshow**(char **string*, float *numArray*[], int *size*)

void **PSxyshow**(char **string*, float *numArray*[], int *size*)

void **PSyield**(void)

void **PSyshow**(char **string*, float *numArray*[], int *size*)

Client Library Functions

Controlling a PostScript Execution Context

Create a context

DPSContext	DPSCreateContext (const char * <i>hostName</i> , const char * <i>serverName</i> , DPSTextProc <i>textProc</i> , DPSErrorProc <i>errorProc</i>) [†]
DPSContext	DPSCreateContextWithTimeoutFromZone (const char * <i>hostName</i> , const char * <i>serverName</i> , DPSTextProc <i>textProc</i> , DPSErrorProc <i>errorProc</i> , int <i>timeout</i> , NXZone * <i>zone</i>) [†]
DPSContext	DPSCreateNonsecureContext (const char * <i>hostName</i> , const char * <i>serverName</i> , DPSTextProc <i>textProc</i> , DPSErrorProc <i>errorProc</i> , int <i>timeout</i> , NXZone * <i>zone</i>) [†]
DPSContext	DPSCreateStreamContext (NXStream * <i>stream</i> , int <i>debugging</i> , DPSPProgramEncoding <i>progEnc</i> , DPSNameEncoding <i>nameEnc</i> , DPSErrorProc <i>errorProc</i>) [†]
void	DPSDestroyContext (DPSContext <i>context</i>)

Create a child context

int	DPSChainContext (DPSContext <i>parent</i> , DPSContext <i>child</i>)
void	DPSUnchainContext (DPSContext <i>context</i>)

Access the current context

void	DPSSetContext (DPSContext <i>context</i>)
DPSContext	DPSGetCurrentContext (void)

Control a context

int	DPSSynchronizeContext (DPSContext <i>context</i> , int <i>enableFlag</i>) [†]
void	DPSWaitContext (DPSContext <i>context</i>)
void	DPSAsynchronousWaitContext (DPSContext <i>context</i> , DPSPingProc <i>handler</i> , void * <i>userData</i>)

Warning: The following two context-controlling functions aren't implemented in NeXTSTEP

void	DPSInterruptContext ()
void	DPSResetContext ()

Extract space from a context

DPSSpace **DPSSpaceFromContext**(DPSContext *context*)

Destroy a space and all contexts in it

void **DPSDestroySpace**(DPSSpace *space*)

Sending Data to the Window Server

Send PostScript code to the Window Server

void **DPSWritePostScript**(DPSContext *context*, const void **buf*, int *count*)
void **DPSWriteData**(DPSContext *context*, const void **buf*, unsigned int *count*)
void **DPSPrintf**(DPSContext *context*, const char **format*, ...)
void **DPSFlushContext**(DPSContext *context*)
void **DPSFlush**(void)[†]
void **DPSSendEOF**(DPSContext *context*)[†]

Send an encoded PostScript path to the Window Server

void **DPSDoUserPath**(void **coords*, int *numCoords*, DPSNumberFormat *numType*,
 unsigned char **ops*, int *numOps*, void **bbox*, int *action*)[†]
void **DPSDoUserPathWithMatrix**(void **coords*, int *numCoords*,
 DPSNumberFormat *numType*, unsigned char **ops*, int *numOps*,
 void **bbox*, int *action*, float *matrix*[6])[†]

User Objects and User Names

Create a user object

int **DPSDefineUserObject**(int *index*)[†]
void **DPSUndefineUserObject**(int *index*)[†]

Access the system and user name tables

void **DPSMapNames**(DPSContext *context*, unsigned int *numNames*,
 const char *const **nameArray*, long int *const **numPtrArray*)
const char * **DPSNameFromIndex**(int *index*)
const char * **DPSNameFromTypeAndIndex**(short *type*, int *index*)[†]

Event-Handling

Access events from the Window Server

int **DPSGetEvent**(DPSTContext *context*, NXEvent **anEvent*, int *mask*, double *timeout*, int *threshold*)[†]
int **DPSPeekEvent**(DPSTContext *context*, NXEvent **anEvent*, int *mask*, double *timeout*, int *threshold*)[†]
void **DPSDiscardEvents**(DPSTContext *context*, int *mask*)[†]

Coalesce events

int **DPSSetTracking**(int *flag*)[†]

Set the event-filter function

DPSEventFilterFunc **DPSSetEventFunc**(DPSTContext *context*, DPSEventFilterFunc *func*)[†]

Create an event

int **DPSPostEvent**(NXEvent **anEvent*, int *atStart*)[†]

Create a timed entry

DPSTimedEntry **DPSAddTimedEntry**(double *period*, DPSTimedEntryProc *handler*, void **userData*, int *priority*)[†]
void **DPSRemoveTimedEntry**(DPSTimedEntry *teNumber*)[†]

Initiate a count down for the wait cursor

void **DPSStartWaitCursorTimer**(void)[†]

Allow dead key processing

void **DPSSetDeadKeysEnabled**(DPSTContext *context*, int *flag*)[†]

Generate an event mask for an event type

int **NX_EVENTCODEMASK**(int *type*)

Handle errors

DPSErrorProc	DPSSetErrorProc (DPSErrorProc <i>context</i> , DPSErrorProc <i>ep</i>)
void	DPSDefaultErrorProc (DPSErrorProc <i>context</i> , DPSErrorCode <i>errorCode</i> , long unsigned int <i>arg1</i> , long unsigned int <i>arg2</i>)
void	DPSSetErrorBackstop (DPSErrorProc <i>errorProc</i>)
DPSErrorProc	DPSGetCurrentErrorBackstop (void)
void	DPSPrintError (FILE * <i>fp</i> , const DPSErrorProc <i>error</i>) [†]
void	DPSPrintErrorToStream (NXStream * <i>stream</i> , const DPSErrorProc <i>error</i>) [†]

Functions Used by pswrap

Wait for return values from the Window Server

void	DPSAwaitReturnValues (DPSErrorProc <i>context</i>)
------	--

Write strings in binary object sequence

void	DPSWriteStringChars (DPSErrorProc <i>context</i> , const char * <i>buf</i> , unsigned int <i>count</i>)
------	---

Write PostScript code in a specified format

void	DPSWriteTypedObjectArray (DPSErrorProc <i>context</i> , DPSErrorProc <i>type</i> , const void * <i>array</i> , unsigned int <i>length</i>)
------	---

Begin a new binary object sequence

void	DPSBinObjSeqWrite (DPSErrorProc <i>context</i> , const void * <i>buf</i> , unsigned int <i>count</i>)
------	---

Define information expected from the PostScript interpreter

void	DPSSetResultTable (DPSErrorProc <i>context</i> , DPSErrorProc <i>table</i> , unsigned int <i>length</i>)
------	--

Update a context's name map from the client library's name map

void	DPSUpdateNameMap (DPSErrorProc <i>context</i>)
------	--

Types and Constants

Defined Types

DPSTextRec

```
typedef struct _t_DPSTextRec {
    char *priv;
    DPSSpace space;
    DPSProgramEncoding programEncoding;
    DPSNameEncoding nameEncoding;
    struct _t_DPSProcsRec const * procs;
    void (*textProc)();
    void (*errorProc)();
    DPSResults resultTable;
    unsigned int resultTableLength;
    struct _t_DPSTextRec *chainParent, *chainChild;
    DPSTextType type;
} DPSTextRec, *DPSText;
```

DPSTextType

```
typedef enum {
    dps_machServer,
    dps_fdServer,
    dps_stream
} DPSTextType;
```

DPSErrorCode

```
typedef enum _DPSErrorCode {
    dps_err_ps = DPS_ERROR_BASE,
    dps_err_nameTooLong,
    dps_err_resultTagCheck,
    dps_err_resultTypeCheck,
    dps_err_invalidContext,
    dps_err_select = DPS_NEXT_ERROR_BASE,
    dps_err_connectionClosed,
    dps_err_read,
    dps_err_write,
```



```
    dps_err_invalidFD,  
    dps_err_invalidTE,  
    dps_err_invalidPort,  
    dps_err_outOfMemory,  
    dps_err_cantConnect  
} DPSErrorCode;
```

DPSEventFilterFunc

```
typedef int (*DPSEventFilterFunc)(NXEvent *ev);
```

DPSFDProc

```
typedef void (*DPSFDProc)( int fd, void *userData );
```

DPSNumberFormat

```
typedef enum _DPSNumberFormat {  
#ifdef __BIG_ENDIAN__  
    dps_float = 48,  
    dps_long = 0,  
    dps_short = 32  
#else  
    dps_float = 48+128,  
    dps_long = 0+128,  
    dps_short = 32+128  
} DPSNumberFormat;
```

DPSPingProc

```
typedef void (*DPSPingProc)  
    (DPSContext ctxt,  
    void *userData);
```

DPSPortProc

```
typedef void (*DPSPortProc)  
    ( msg_header_t *msg,  
    void *userData );
```

DPSTimedEntry

```
typedef struct __DPSTimedEntry *DPSTimedEntry;
```

DPSTimedEntryProc

```
typedef void (*DPSTimedEntryProc)
    (DPSTimedEntry timedEntry,
     double now,
     void *userData );
```

DPSUserPathAction

```
typedef enum _DPSUserPathAction {
    dps_uappend = 176,
    dps_ufill = 179,
    dps_ueofill = 178,
    dps_ustroke = 183,
    dps_ustrokepath = 364,
    dps_inufill = 93,
    dps_inueofill = 92,
    dps_inustroke = 312,
    dps_def = 51,
    dps_put = 120
} DPSUserPathAction;
```

DPSUserPathOp

```
typedef enum _DPSUserPathOp {
    dps_setbbox = 0,
    dps_moveto,
    dps_rmoveto,
    dps_lineto,
    dps_rlineto,
    dps_curveto,
    dps_rcurveto,
    dps_arc,
    dps_arcn,
    dps_arct,
    dps_closepath,
    dps_ucache
} DPSUserPathOp;
```

NXCoord

```
typedef float NXCoord
```

NXEvent

```
typedef struct _NXEvent {
    int type;
    NXPoint location;
    long time;
    int flags;
    unsigned int window;
    NXEventData data;
    DPSContext ctxt;
} NXEvent, *NXEventPtr;
```

NXEventData

```
typedef union {
    struct {
        short eventNum;
        int click;
        unsigned char pressure;
    } mouse;
    struct {
        short repeat;
        unsigned short charSet;
        unsigned short charCode;
        unsigned short keyCode;
        short keyData;
    } key;
    struct {
        short eventNum;
        int trackingNum;
        int userData;
    } tracking;
    struct {
        short reserved;
        short subtype;
        union {
            float F[2];
            long L[2];
            short S[4];
            char C[8];
        } misc;
    } compound;
} NXEventData;
```

NXPoint

```
typedef struct _NXPoint {  
    NXCoord x;  
    NXCoord y;  
} NXPoint;
```

NXSize

```
typedef struct _NXSize {  
    NXCoord width;  
    NXCoord height;  
} NXSize;
```

Symbolic Constants

All Contexts

DPS_ALLCONTEXTS

Alpha Constants

NX_DATA
NX_ONES

Character Set Values

NX_ASCIISET
NX_SYMBOLSET
NX_DINGBATSET

Compositing Operations

NX_CLEAR
NX_COPY
NX_SOVER
NX_SIN
NX_SOUT
NX_SATOP
NX_DOVER
NX_DIN
NX_DOUT
NX_DATOP
NX_XOR
NX_PLUSD
NX_HIGHLIGHT
NX_PLUSL

Error Code Bases

DPS_ERROR_BASE
DPS_NEXT_ERROR_BASE

Event Types

NX_NULLEVENT
NX_LMOUSEDOWN
NX_LMOUSEUP
NX_LMOUSEDRAGGED
NX_MOUSEDOWN
NX_MOUSEUP
NX_MOUSEDRAGGED
NX_RMOUSEDOWN
NX_RMOUSEUP
NX_RMOUSEDRAGGED
NX_MOUSEMOVED
NX_MOUSEENTERED
NX_MOUSEEXITED
NX_KEYDOWN
NX_KEYUP
NX_FLAGSCHANGED
NX_KITDEFINED
NX_SYSDEFINED
NX_APPDEFINED
NX_TIMER
NX_CURSORUPDATE
NX_JOURNALEVENT

Meaning

A non-event
Left mouse-down
Left mouse-up
left mouse-dragged
Same as NX_LMOUSEDOWN
Same as NX_LMOUSEUP
Same as NX_LMOUSEDRAGGED
Right mouse-down
Right mouse-up
Right mouse-dragged
Mouse-moved
Mouse-entered
Mouse-exited
Key-down
Key-up event
Flags-changed
Appkit-defined
System-defined
Application-defined
Timer used for tracking
Cursor tracking
Event used by journaling

NX_FIRSTEVENT
NX_LASTEVENT
NX_ALLEVENTS

The smallest-valued event constant
The greatest-valued event constant
A value that includes all event types

Event Type Masks

NX_NULLEVENTMASK
NX_LMOUSEDOWNMASK
NX_LMOUSEUPMASK
NX_RMOUSEDOWNMASK
NX_RMOUSEUPMASK
NX_MOUSEMOVEDMASK
NX_LMOUSEDRAGGEDMASK
NX_RMOUSEDRAGGEDMASK
NX_MOUSEENTEREDMASK
NX_MOUSEEXITEDMASK
NX_KEYDOWNMASK
NX_KEYUPMASK
NX_FLAGSCHANGEDMASK
NX_KITDEFINEDMASK
NX_APPDEFINEDMASK
NX_SYSDEFINEDMASK
NX_TIMERMASK
NX_CURSORUPDATESMASK
NX_MOUSEDOWNMASK
NX_MOUSEUPMASK
NX_MOUSEDRAGGEDMASK
NX_JOURNALEVENTMASK

Forever

NX_FOREVER

Keyboard State Flags Masks

<code>NX_ALPHASHIFTMASK</code>	Shift lock
<code>NX_SHIFTMASK</code>	Shift key
<code>NX_CONTROLMASK</code>	Control key
<code>NX_ALTERNATEMASK</code>	Alt key
<code>NX_COMMANDMASK</code>	Command key
<code>NX_NUMERICPADMASK</code>	Number pad key
<code>NX_HELPMASK</code>	Help key
<code>NX_NEXTCTRLKEYMASK</code>	Control key
<code>NX_NEXTLSHIFTKEYMASK</code>	Left shift key
<code>NX_NEXTRSHIFTKEYMASK</code>	Right shift key
<code>NX_NEXTLCMDKEYMASK</code>	Left command key
<code>NX_NEXTRCMDKEYMASK</code>	Right command key
<code>NX_NEXTLALTKEYMASK</code>	Left alt key
<code>NX_NEXTRALTKEYMASK</code>	Right alt key

Meaning

Event Flags Masks

<code>NX_STYLUSPROXIMITYMASK</code>	Stylus is in proximity (for tablets)
<code>NX_NONCOALSESCEDMASK</code>	Event coalescing disabled

Meaning

Window Backing Types

`NX_RETAINED`
`NX_NONRETAINED`
`NX_BUFFERED`

Window Screen List Placement

`NX_ABOVE`
`NX_BELOW`
`NX_OUT`

6 *Distributed Objects*

Classes

NXConnection

Inherits From: NXInvalidationNotifier : Object

Conforms To: NXSenderIsInvalid
NXReference (NXInvalidationNotifier)

Establishing a Connection

- | | |
|---|----------------------------------|
| + (NXProxy *) connectToName: (const char *) <i>n</i> | Connects to the named object |
| + (NXProxy *) connectToName: (const char *) <i>n</i>
fromZone: (NXZone *) <i>z</i> | Connects to the named object |
| + (NXProxy *) connectToName: (const char *) <i>n</i>
onHost: (const char *) <i>h</i> | Connects to the named object |
| + (NXProxy *) connectToName: (const char *) <i>n</i>
onHost: (const char *) <i>h</i>
fromZone: (NXZone *) <i>z</i> | Connects to the named object |
| + (NXProxy *) connectToPort: (NXPort *) <i>p</i> | Connects over the specified port |
| + (NXProxy *) connectToPort: (NXPort *) <i>p</i>
fromZone: (NXZone *) <i>z</i> | Connects over the specified port |

- + (NXProxy *)**connectToPort**:(NXPort *)*aPort*
 withInPort:(NXPort *)*inPort* Connects over the specified port
- + (NXProxy *)**connectToPort**:(NXPort *)*aPort*
 withInPort:(NXPort *)*inPort*
 fromZone:(NXZone *)*z* Connects over the specified port

Ascertaining Connections

- + **connections**:(List *)*l* Returns all connections

Registering an Object

- + **registerRoot**:*anObject* Establishes a root object
- + **registerRoot**:*anObject*
 fromZone:(NXZone *)*z* Establishes a root object
- + **registerRoot**:*anObject*
 withName:(const char *)*n* Establishes a named root object
- + **registerRoot**:*anObject*
 withName:(const char *)*n*
 fromZone:(NXZone *)*z* Establishes a named root object

Eliminating References

- + **removeObject**:*anObject* Removes an object from all connections

Invalidation

- + **unregisterForInvalidationNotification**:
 anObject Unregisters an object for notifications

Statistics

- + (int)**messagesReceived** Number of messages received over connection

Timeouts

- + **setDefaultTimeout**:(int)*t* Sets default timeout for all connections
- + (int)**defaultTimeout** Returns default timeout for all connections
- **setInTimeout**:(int)*t* Sets in timeout
- **setOutTimeout**:(int)*t* Sets out timeout
- (int)**inTimeout** Returns in timeout
- (int)**outTimeout** Returns out timeout

Zone Usage

- + **setDefaultZone:** (NXZone *)*zone* Sets default zone for all connections
- (NXZone *)**defaultZone** Returns default zone for all connections

Assigning a Delegate

- **setDelegate:***anObject* Sets the connection's delegate
- **delegate** Returns the connection's delegate

Returning Port Objects

- (NXPort *)**inPort** Returns the connection's in port
- (NXPort *)**outPort** Returns the connection's out port

Getting and Setting the Root Object

- **rootObject** Returns the connection's root object
- **setRoot:***anObject* Sets the connection's root object

Imported and Exported Objects

- (List *)**remoteObjects** Returns the connection's remote proxies
- (List *)**localObjects** Returns the connection's local proxies

Returning a Proxy

- **getLocal:***anId* Returns an object's local proxy
- **newRemote:**(unsigned)*anObject*
withProtocol:(Protocol *)*p* Creates a remote proxy for an object

Running a Connection

- **run** Runs the connection and blocks
- **runWithTimeout:**(int)*t* Runs the connection for a while
- **runInNewThread** Runs the connection asynchronously
- **runFromAppKit** Runs the connection from DPS client
- **runFromAppKitWithPriority:**(int)*priority* Runs the connection from DPS client

Freeing an NXConnection Instance

- **free** Frees the connection

NXProxy

Inherits From: none
Conforms To: NXReference
NXTransport

Counting References

- `addReference` Adds a reference
- `free` Eliminates a reference
- `references` Returns number of references

Returning the proxy's connection

- `connectionForProxy` Returns the proxy's connection

Freeing an NXProxy instance

- `freeProxy` Frees the proxy but not its real object

Determining if an object is a proxy

- `isProxy` Identifies the receiver as a proxy

Specifying a protocol

- `setProtocolForProxy:(Protocol *)proto` Sets the proxy's protocol for efficiency

Object Additions

Making Objects Distributable

- `encodeRemotelyFor:`
 (NXConnection *)*connection*
 `freeAfterEncoding:(BOOL *)flagp`
 `isBycopy:(BOOL)isBycopy`
 Transports an object using a proxy
- (BOOL)`isProxy` Identifies the receiver as an object

Protocols

NXDecoding

Adopted By: A private class that decodes data sent across a connection

Methods

- **decodeBytes:**(void *)*bytes*
 count:(int)*count* Decodes untyped data
- **decodeData:**(void *)*d*
 ofType:(const char *)*t* Decodes typed data
- **decodeMachPort:**(port_t *)*pp* Decodes a Mach port
- **decodeObject** Decodes an object
- **decodeVM:**(void **)*bytes*
 count:(int *)*count* Decodes virtual memory pages

NXEncoding

Adopted By: A private class that encodes data across a connection

Methods

- **encodeBytes:**(const void *)*bytes*
 count:(int)*count* Encodes untyped data
- **encodeData:**(void *)*data*
 ofType:(const char *)*type* Encodes typed data
- **encodeMachPort:**(port_t)*port* Encodes a Mach port
- **encodeObject:***anObject* Encodes an object as a proxy
- **encodeObjectBycopy:***anObject* Encodes a copy of an object
- **encodeVM:**(const void *)*bytes*
 count:(int)*count* Encodes virtual memory pages

NXTransport

Adopted By: List (common classes)
NXData (Mach Kit)
NXPort (Mach Kit)
NXProxy

Methods

- **encodeRemotelyFor:** Determines what to encode across a connection
(NXConnection *)*connection*
freeAfterEncoding:(BOOL *)*flag*
isBycopy:(BOOL)*isBycopy*
- **encodeUsing:**(id <NXEncoding>)*portal* Encodes an object across a connection
- **decodeUsing:**(id <NXDecoding>)*portal* Decodes an object over a connection

Types and Constants

Defined Types

NXRemoteException

```
typedef enum {  
    NX_REMOTE_EXCEPTION_BASE = 11000,  
    NX_couldntSendException = 11001,  
    NX_couldntReceiveException = 11002,  
    NX_couldntDecodeArgumentsException = 11003,  
    NX_unknownMethodException = 11004,  
    NX_objectInaccessibleException = 11005,  
    NX_objectNotAvailableException = 11007,  
    NX_remoteInternalException = 11008,  
    NX_multithreadedRecursionDeadlockException = 11009,  
    NX_destinationInvalid = 11010,  
    NX_originatorInvalid = 11011,  
    NX_sendTimedOut = 11012,  
    NX_receiveTimedOut = 11013,  
    NX_REMOTE_LAST_EXCEPTION = 11999  
} NXRemoteException;
```

Symbolic Constants

Timeout Constantst	Value
NX_CONNECTION_DEFAULT_TIMEOUT	15000



7 *Indexing Kit*

Classes

IXAttributeParser

Inherits From: Object

Initializing an Instance

– **init** Initializes and returns a new IXAttributeParser

Managing Readers

– **setAttributeReaders:(List *)aList** Sets the IXAttributeReaders to those in *aList*, freeing the previous IXAttributeReaders

– **getAttributeReaders:(List *)aList** Returns in *aList* the IXAttributeReaders

Managing Text Stream Types

– (BOOL)**understandsType:(const char *)aType** Returns YES if streams or files of *aType* are parsed

– **addSourceType:(const char *)aType** Sets *aType* as a type that will be parsed

– **removeSourceType:(const char *)aType** Unsets *aType* as a type that will be parsed

Managing Parse Options

- **setMinimumWeight:**(unsigned int)*anInt* Sets the minimum weight for a token to be included
- (unsigned int)**minimumWeight** Returns the minimum weight for a token to be included
- **setPercentPassed:**(unsigned int)*anInt* Sets the percent of tokens dropped
- (unsigned int)**percentPassed** Returns the percent of tokens dropped
- **setWeightingDomain:**(IXWeightingDomain *)*aDomain* Sets the weighting domain for calculating peculiarities
- (IXWeightingDomain *)**weightingDomain** Returns the weighting domain
- **setWeightingType:**(IXWeightingType)*anInt* Sets the type of weighting to calculate
- (IXWeightingType)**weightingType** Returns the type of weighting calculated

Parsing Text

- **parseFile:**(const char *)*filename* Parses the contents of *filename* if possible, adding the information to the attribute-value list
- **ofType:**(const char *)*aType*
- **parseStream:**(NXStream *)*stream* Parses the contents of *stream* if possible, adding the information to the attribute-value list
- **ofType:**(const char *)*aType*
- (NXStream *)**analyzeFile:**(const char *)*filename* Parses the contents of *filename* if possible, adding the information and returning the analyzed stream
- **ofType:**(const char *)*aType*
- (NXStream *)**analyzeStream:**(NXStream *)*stream* Parses the contents of *stream* if possible, adding the information and returning the analyzed stream
- **ofType:**(const char *)*aType*
- **reset** Clears all compiled information in the attribute-value list

IXAttributeQuery

Inherits From: Object

Initializing an IXAttributeQuery

- **initQueryString:**(const char *)*aString*
– **andAttributeParser:**(IXAttributeParser *)*aParser* Initializes a new IXAttributeQuery with *aString* as the query expression and *aParser* used to parse it

Accessing Attributes

- (char *)**attributeNames** Returns the names of attributes in the query string
- (IXAttributeParser *)**attributeParser** Returns the IXAttributeParser

Retrieving the Query Expression

- (const char *)**queryString** Returns the query string

Evaluating the Query

- (IXPostingList *)**evaluateFor:anObject** Evaluates the query string against *anObject*, returning the records that match in an IXPostingList

IXAttributeReader

Inherits From: Object

Conforms To: IXAttributeReading

Altering Words

- (unsigned int)**foldPlural:(char *)aString
inLength:(unsigned int)aLength** Reduces *aString* to its singular form
- (unsigned int)**reduceStem:(char *)aString
inLength:(unsigned int)aLength** Reduces *aString* to its base or root form

Setting Reader Options

- **setCaseFolded:(BOOL)flag** Sets whether uppercase letters are changed to lowercase
- (BOOL)**isCaseFolded** Returns whether uppercase letters are changed to lowercase
- **setPluralsFolded:(BOOL)flag** Sets whether plural words are reduced to singular
- (BOOL)**arePluralsFolded** Returns whether plural words are reduced to singular
- **setStemsReduced:(BOOL)flag** Sets whether words are reduced to base or root forms
- (BOOL)**areStemsReduced** Returns whether words are reduced to base or root forms
- **setPunctuation:(const char *)aString** Sets the set of characters used to delimit tokens to *aString*

- (char *)**punctuation** Returns the set of characters used to delimit tokens
- **setStopWords**:(const char *)*stopWords* Sets the words that are deleted from a stream
- (char *)**stopWords** Returns the words that are deleted from a stream

IXBTree

Inherits From: Object

Conforms To: IXBlockAndStoreAccess
IXNameAndFileAccess
IXComparatorSetting
IXComparisonSetting

Accessing IXBTree Information

- (unsigned int)**count** Returns the number of key-value pairs in the IXBTree
- (unsigned int)**keyLimit** Returns the maximum allowed length for a key

Affecting IXBTree Contents

- **empty** Removes the contents of the IXBTree
- **compact** Compacts the IXBTree's contents to consume less space

Optimizing Performance

- **optimizeForSpace** Keeps the IXBTree small, making insertions slower but seeks faster
- **optimizeForTime** Makes insertions faster, and seeks slower

IXBTreeCursor

Inherits From: Object

Conforms To: IXCursorPositioning

Initializing an IXBTreeCursor

– **initWithBTree:**(IXBTree *)*aBTree* Initializes a new IXBTreeCursor to move in *aBTree*

Accessing the IXBTree

– (IXBTree *)**btree** Returns the IXBTree that the cursor moves in

Positioning with Hints

– (BOOL)**setKey:**(void *)*aKey*
andLength:(unsigned int)*aLength*
withHint:(unsigned int)*aHint* Positions the IXBTreeCursor at *aKey* if possible, using *aHint* to speed search; returns YES if *aKey* is found

– (BOOL)**getKey:**(void **)*aKey*
andLength:(unsigned int *)*aLength*
withHint:(unsigned int *)*aHint* Returns the position of the cursor in *aKey* and *aLength*; also returning in *aHint* a hint that can be used to speed later search; returns NO if the cursor is past the end of the IXBTree’s key space

Accessing IXBTree Data

– (BOOL)**writeValue:**(void *)*aValue*
andLength:(unsigned int)*aLength* Writes or inserts *aValue* at the cursor’s position

– **writeRange:**(void *)*aRange*
atOffset:(unsigned int)*aOffset*
forLength:(unsigned int)*aLength* Writes *aRange* into the value at the cursor’s position; if the cursor isn’t exactly on a key, raises an exception

– (unsigned int)**readValue:**(void **)*aValue* Reads the value at the cursor’s position and returns its length; slides forward if needed to get a value

– (unsigned int)**readRange:**(void **)*aRange*
ofLength:(unsigned int)*aLength*
atOffset:(unsigned int)*aOffset* Reads a portion of the value at the cursor’s position and returns its length; if the cursor isn’t exactly on a key, raises an exception

– **removeValue** Removes the value at the cursor’s position; if the cursor isn’t exactly on a key, raises an exception

IXFileFinder

Inherits From: Object

Conforms To: IXBlockAndStoreAccess
IXNameAndFileAccess
IXFileFinderConfiguration
IXFileFinderQueryAndUpdate
NXReference

Initializing an IXFileFinder

- **initInStore:**(IXStore *)*aStore*
atPath:(const char *)*path* Initializes a new IXFileFinder in *aStore* to index files in *path*
- **initFromBlock:**(unsigned int)*aHandle*
inStore:(IXStore *)*aStore*
atPath:(const char *)*path* Reloads an IXFileFinder from block *aHandle* in *aStore* to index files in *path*
- **initWithName:**(const char *)*aName*
inFile:(const char *)*filename*
atPath:(const char *)*path* Initializes a new IXFileFinder named *aName* in *filename* to index files in *path*
- **initFromName:**(const char *)*aName*
inFile:(const char *)*filename*
forWriting:(BOOL)*flag*
atPath:(const char *)*path* Reloads an IXFileFinder stored under *aName* in *filename*, allowing writing back to the file according to *flag*, and set to index files in *path*

IXFileRecord

Inherits From: Object

Conforms To: NXTransport

Initializing a New Instance

- **initWithFileFinder:**(IXFileFinder *)*aFileFinder* Initializes a new IXFileRecord for *aFileFinder*

Getting the File Finder

- (IXFileFinder *)**fileFinder** Returns the IXFileFinder that the IXFileRecord belongs to

Accessing File Attributes

- **setFilename:**(const char *)*aName* Sets the filename that the IXFileRecord refers to *aName*
- (const char *)**filename** Returns the filename that the IXFileRecord refers to
- **setFiletype:**(const char *)*aType* Sets the recorded type for the associated file to *aType*
- (const char *)**filetype** Returns the recorded type for the associated file
- **setDescription:**(const char *)*aDescription* Sets the description for the associated file to *aName*
- (const char *)**description** Returns the description for the associated file
- **setFiledate:**(unsigned int)*aDate* Sets the recorded creation date for the associated file to *aDate*
- (unsigned int)**filedate** Returns the recorded creation date for the associated file

Accessing UNIX File Information

- (const struct stat *)**statBuffer** Returns the cached UNIX stat buffer for the associated file, or NULL if one hasn't been cached

IXLanguageReader

Inherits From: IXAttributeReader : Object

Conforms To: IXAttributeReading (IXAttributeReader)

Getting Language Information

- + (char *)**installedLanguages** Returns the languages for which readers are installed
- + (Class)**classForLanguage:**(const char *)*aLanguage* Returns the IXLanguageReader subclass for *aLanguage*

Getting Objects Associated with Languages

- + **readerForLanguage:**(const char *)*aLanguage* Returns an IXLanguageReader for *aLanguage*
- + **domainForLanguage:**(const char *)*aLanguage* Returns an IXWeightingDomain for *aLanguage*

Getting the Target Language

- + (NXAtom)**targetLanguage** Returns the target language of the subclass
- (NXAtom)**targetLanguage** Returns the target language of the reader

Disabling Dynamic Loading

+ `disableLoading`

Disables dynamic loading of external language readers

IXPostingCursor

Inherits From: IXBTreeCursor : Object

Conforms To: IXPostingExchange
IXPostingOperations
IXCursorPositioning (IXBTreeCursor)

Methods

This class declares no methods.

IXPostingList

Inherits From: List : Object

Conforms To: IXPostingExchange
NXTransport

Initializing an IXPostingList

- **initWithSource:**(id <IXRecordReading>)aSource Initializes a new IXPostingList to extract objects from *aSource*
- **initWithSource:**(id <IXRecordReading>)aSource
 andPostingsIn:(id <IXPostingExchange>)anObject Initializes a new IXPostingList to extract objects from *aSource* and initially contain the postings in *anObject*

Retrieving the Source

- (id <IXRecordReading>)source Returns the archive from which objects are extracted

Manipulating Objects by Handle

- **addHandle:**(unsigned int)*aHandle*
withWeight:(unsigned int)*aWeight* Adds a new posting to the end of the list
- **insertHandle:**(unsigned int)*aHandle*
withWeight:(unsigned int)*aWeight*
at:(unsigned int)*index* Inserts a new posting at *index*
- **replaceHandleAt:**(unsigned int)*index*
with:(unsigned int)*aHandle*
weight:(unsigned int)*aWeight* Replaces the posting at *index* with the information supplied

Manipulating Objects by id

- **addObject:***anObject*
withWeight:(unsigned int)*aWeight* Adds *anObject* to the end of the list with *aWeight*, but no posting handle
- **insertObject:***anObject*
withWeight:(unsigned int)*aWeight*
at:(unsigned int)*index* Inserts *anObject* with *aWeight* at *index*, but without a posting handle
- **replaceObjectAt:**(unsigned int)*index*
with:*anObject*
weight:(unsigned int)*aWeight* Replaces the object at *index* with the object and weight supplied, but without a posting handle

Manipulating Objects by Index

- (unsigned int)**indexForHandle:**(unsigned int)*handle* Returns the index of the posting with handle *handle*
- (unsigned int)**handleOfObjectAt:**(unsigned int)*index* Returns the handle for the posting at *index*
- (unsigned int)**weightOfObjectAt:**(unsigned int)*index* Returns the weight for the posting at *index*

Sorting the Contents

- **sortByWeightAscending:**(BOOL)*flag* Sorts the postings and objects by their weight
- **sortBySelector:**(SEL)*aSelector*
ascending:(BOOL)*flag* Sorts the postings and objects based on the return values of *aSelector* as sent to each object

IXPostingSet

Inherits From: Object

Conforms To: IXPostingExchange
IXPostingOperations

Initializing Instances

- **initWithCount:(unsigned int)count
andPostings:(const IXPosting *)postings** Initializes a new IXPostingSet to contain *count* postings
- **initWithPostingsIn:anObject** Initializes a new IXPostingSet to contain the postings in *anObject*

Setting the Postings

- **setCount:(unsigned int)count
andPostings:(const IXPosting *)postings
byCopy:(BOOL)flag** Sets the postings in the IXPostingSet to *count* postings, copying them if *flag* is YES

Accessing Postings by Position

- (unsigned int)**setPosition:(unsigned int)index** Sets the selected posting to the one at position *index*

Performing Set Operations

- **formUnionWithPostingsIn:(id <IXPostingExchange>)anObject** Performs a set union with the postings in *anObject*, altering the IXPostingSet's contents but not *anObject*'s
- **formIntersectionWithPostingsIn:(id <IXPostingExchange>)anObject** Performs a set intersection with the postings in *anObject*, altering the IXPostingSet's contents but not *anObject*'s
- **subtractPostingsIn:(id <IXPostingExchange>)anObject** Performs a set subtraction with the postings in *anObject*, altering the IXPostingSet's contents but not *anObject*'s

IXRecordManager

Inherits From: Object

Conforms To: IXBlockAndStoreAccess
IXNameAndFileAccess
IXBlobWriting
IXRecordDiscarding
IXRecordWriting
IXTransientAccess
IXTransientMessaging

Adding and Removing Attributes

- **addAttributeNamed:(const char *)aName
forSelector:(SEL)aSelector** Creates an attribute named *aName*, based on the values returned by objects sent *aSelector*
- **(BOOL)hasAttributeNamed:(const char *)aName** Returns whether an attribute named *aName* exists
- **removeAttributeNamed:(const char *)aName** Removes the attribute named *aName*

Key Comparison

- **setComparisonFormat:(const char *)aFormat
forAttributeNamed:(const char *)aName** Sets the data format used for values in the attribute named *aName*
- **(const char *)comparisonFormatForAttributeNamed:(const char *)aName** Returns the data format used for values in the attribute named *aName*
- **setComparator:(IXComparator *)aComparator
andContext:(const void *)aContext
forAttributeNamed:(const char *)aName** Sets the comparator function and context used for values in the attribute named *aName*
- **getComparator:(IXComparator **)aComparator
andContext:(const void **)aContext
forAttributeNamed:(const char *)aName** Returns the comparator function and context used for values in the attribute named *aName*

Setting Attribute Targets

- **setTargetClass:aClass
forAttributeNamed:(const char *)aName** Restricts the attribute named *aName* to contain references only to objects of class *aClass* (or a subclass)
- **getTargetName:(const char **)aName
andVersion:(unsigned int *)targetVersion
forAttributeNamed:(const char *)aName** Returns the name and class version of the target class for the attribute named *aName*

Accessing Attributes

- (IXPostingCursor *)**cursorForAttributeNamed:(const char *)*aName***
Returns an IXPostingCursor that may be used to search for values for the attribute named *aName*

Getting Attribute Information

- (SEL)**selectorForAttributeNamed:(const char *)*aName***
Returns the selector used to build the attribute named *aName*
- (char *)**attributeNames**
Returns the names of all attributes in the IXRecordManager

Accessing Classes

- (char *)**classNames**
Returns the names of all classes that have instances stored in the IXRecordManager
- (char *)**attributeNamesForClass:*aClass***
Returns the names of the attributes in the IXRecordManager that contain references to instances of *aClass* (or a subclass)
- (IXPostingList *)**recordsForClass:*aClass***
Returns in an IXPostingList all instances in the IXRecordManager of *aClass* (or a subclass)

Retrieving Discarded Records

- (IXPostingList *)**discards**
Returns in an IXPostingList all discarded records

Setting Attribute Descriptions

- **setDescription:(const char *)*aDescription* forAttributeNamed:(const char *)*aName***
Sets the description for the attribute named *aName* to *aDescription*
- **getDescription:(char **)*aDescription* forAttributeNamed:(const char *)*aName***
Returns the description for the attribute named *aName*

Setting Parsers

- **setParser:(IXAttributeParser *)*aParser* forAttributeNamed:(const char *)*aName***
Sets the IXAttributeParser used for the attribute named *aName*
- (IXAttributeParser *)**parserForAttributeNamed:(const char *)*aName***
Returns the IXAttributeParser used for the attribute named *aName*

IXStore

Inherits From: Object

Initializing, Copying, and Freeing Instances

- **init** Initializes a new IXStore
- **copy** Returns a copy of the IXStore that references the same storage
- **free** Deallocates the IXStore, and all the storage if it's not shared

Creating, Copying, and Freeing Blocks

- **createBlock:(unsigned int *)aHandle ofSize:(unsigned int)size** Creates a block of size *size* and returns its identifier in *aHandle*
- **(unsigned int)copyBlock:(unsigned int)aHandle atOffset:(unsigned int)anOffset forLength:(unsigned int)aLength** Copies a portion of the block identified by *aHandle* and returns the copies identifier
- **freeBlock:(unsigned int)aHandle** Frees the block identified by *aHandle*

Opening and Closing Blocks

- **(void *)openBlock:(unsigned int)aHandle atOffset:(unsigned int)anOffset forLength:(unsigned int)aLength** Opens a portion of the block identified by *aHandle* for writing if possible, otherwise raises an exception
- **(void *)readBlock:(unsigned int)aHandle atOffset:(unsigned int)anOffset forLength:(unsigned int)aLength** Opens a portion of the block identified by *aHandle* for reading if possible, otherwise raises an exception
- **closeBlock:(unsigned int)aHandle** Closes the open block identified by *aHandle*

Managing Block Sizes

- **resizeBlock:(unsigned int)aHandle toSize:(unsigned int)aSize** Resizes the block identified by *aHandle* to *aSize*
- **(unsigned int)sizeOfBlock:(unsigned int)aHandle** Returns the size of the block identified by *aHandle*

Using Transactions

- **(unsigned int)startTransaction** Starts a new transaction and returns the nested level of transactions pending

- **abortTransaction** Aborts the current transaction
- **commitTransaction** Commits the changes made in the current transaction
- (BOOL)**areTransactionsEnabled** Returns YES if a transaction has ever been started
- (unsigned int)**nestingLevel** Returns the nesting level of transactions pending
- (unsigned int)**changeCount** Returns the number of **commitTransaction** and **abortTransaction** messages received by the instantiation

Accessing the Contents

- **setContentts:(vm_address_t)someContents andLength:(vm_size_t)aLength** Replaces the IXStore’s storage with that in *someContents*
- **getContentts:(vm_address_t *)theContents andLength:(vm_size_t *)aLength** Returns in *theContents* and *aLength* the IXStore’s storage and its length

Reducing Memory Consumption

- **compact** Compacts the blocks of storage to take as little space as possible; delayed until all transactions are finished

IXStoreBlock

Inherits From: Object

Conforms To: IXBlockAndStoreAccess
NXReference

Accessing the Block’s Contents

- (void *)**openAtOffset:(unsigned int)anOffset forLength:(unsigned int)aLength** Opens a portion of the block for writing if possible; raises an exception otherwise
- (void *)**readAtOffset:(unsigned int)anOffset forLength:(unsigned int)aLength** Opens a portion of the block for reading if possible; raises an exception otherwise
- (unsigned int)**copyAtOffset:(unsigned int)anOffset forLength:(unsigned int)aLength** Copies a portion of the block, returning the copy’s identifier
- **close** Closes the block if it’s been opened

Managing the Block Size

- **resizeTo:**(unsigned int)*size* Resizes the block to *size*
- (unsigned int)*size* Returns the size of the block

Archiving an Object in an IXStoreBlock

- **readObject** Unarchives the object previously archived into the block
- **writeObject:**(unsigned int)*anObject* Archives *anObject* into the block

IXStoreDirectory

Inherits From: Object

Conforms To: IXBlockAndStoreAccess
IXNameAndFileAccess

Adding Entries or Objects

- **addEntryNamed:**(const char *)*aName*
ofClass:*aName* Creates an instance of the store client class named *aName*, and stores it under *aName*
- **addEntryNamed:**(const char *)*aName*
ofClass:*aName*
atBlock:(IXBlockHandle)*aHandle* Creates an instance of the store client class named *aName*, and stores it under *aName* in the block of the IXStoreDirectory's IXStore identified by *aHandle*
- **addEntryNamed:**(const char *)*aName*
forObject:*anObject* Stores the store client *anObject* under *aName*

Removing Entries

- **freeEntryNamed:**(const char *)*aName* Removes from the IXStore the object stored under *aName*
- **removeName:**(const char *)*aName* Removes the reference to the object stored under *aName*, but doesn't remove the object itself
- **empty** Empties all stored objects and names
- **reset** Removes all references to stored objects, but not the objects themselves

Getting Entries

- (BOOL)**hasEntryNamed:**(const char *)*aName* Returns whether there's an object stored under *aName*
- **getBlock:**(unsigned int *)*aHandle* Gets the block identifier for the object stored under *aName*
 ofEntryNamed:(const char *)*aName*
- **getClass:**(id *)*aClass* Gets the class object for the object stored under *aName*
 ofEntryNamed:(const char *)*aName*
- **openEntryNamed:**(const char *)*aName* Activates and returns the object stored under *aName*
- (const char **)**entries** Returns the stored names

IXStoreFile

Inherits From: IXStore : Object

Initializing and Freeing Instances

- **init** Initializes a new IXStoreFile in a temporary file
- **initWithFile:**(const char *)*filename* Initializes a new IXStoreFile with a file named *filename*
- **initWithFile:**(const char *)*filename* Initializes an IXStoreFile from existing storage in *filename*,
 forWriting:(BOOL)*flag* allowing writing back to the file if *flag* is YES
- **free** Frees the IXStoreFile, but doesn't affect the storage file

Limiting the File Mapping Size

- **setSizeLimit:**(vm_size_t)*aLimit* Sets the caching limit for the IXStoreFile to *aLimit*
- (vm_size_t)**sizeLimit** Returns the caching limit for the IXStoreFile

Getting File Information

- (int)**descriptor** Returns the file descriptor for the storage file
- (const char *)**filename** Returns the name of the storage file

IXWeightingDomain

Inherits From: Object

Initializing Instances

- **initFromDomain:**(NXStream *)*stream* Initializes a new IXWeightingDomain from domain format data in *stream*
- **initFromHistogram:**(NXStream *)*stream* Initializes a new IXWeightingDomain from histogram format data in *stream*
- **initFromWFTable:**(NXStream *)*stream* Initializes a new IXWeightingDomain from NeXTSTEP 2.0 WFTable format data in *stream*

Saving Domain Information

- **writeDomain:**(NXStream *)*stream* Write domain format data to *stream*
- **writeHistogram:**(NXStream *)*stream* Writes histogram format data to *stream*
- **writeWFTable:**(NXStream *)*stream* Writes NeXTSTEP 2.0 WFTable format data to *stream*

Counting Tokens

- (unsigned int)**totalTokens** Returns the total number of tokens in the weighting domain
- (unsigned int)**uniqueTokens** Returns the total number of unque tokens

Retrieving Information about Tokens

- (unsigned int)**countForToken:**(void *)*aToken*
 ofLength:(unsigned int)*aLength* Returns the number of occurrences of *aToken* in the domain
- (unsigned int)**rankForToken:**(void *)*aToken*
 ofLength:(unsigned int)*aLength* Returns the ordinal rank of occurrences of *aToken* in the domain
- (float)**frequencyOfToken:**(void *)*aToken*
 ofLength:(unsigned int)*aLength* Returns the frequency of *aToken* in the domain
- (float)**peculiarityOfToken:**(void *)*aToken*
 ofLength:(unsigned int)*aLength*
 andFrequency:(float)*aFrequency* Returns the peculiarity of *aToken* in the domain

Protocols

IXAttributeReading

Adopted By: IXAttributeReader

Methods

- (NXStream *)**analyzeStream:(NXStream *)stream**
Analyzes the text in stream, returning a stream of lexed text in Attribute Reader Format

IXBlobWriting

Adopted By: IXRecordManager

Methods

- (BOOL)**setValue:(const void *)aValue
andLength:(unsigned int)aLength
ofBlob:(const char *)blobName
forRecord:(unsigned int)aHandle**
Stores *aValue* under *blobName* on behalf of the record identified by *aHandle*; returns YES on success, NO otherwise
- (BOOL)**getValue:(void **)aValue
andLength:(unsigned int *)aLength
ofBlob:(const char *)blobName
forRecord:(unsigned int)aHandle**
Retrieves the value and length for *blobName* on behalf of the record identified by *aHandle*; returns YES on success, NO otherwise

IXBlockAndStoreAccess

Adopted By: IXBTree,
IXFileFinder,
IXRecordManager,
IXStoreBlock,
IXStoreDirectory

Initializing and Freeing a Client

- **initInStore:**(IXStore *)*aStore* Initializes a new store client in *aStore*
- **initFromBlock:**(unsigned int)*aHandle*
 inStore:(IXStore *)*aStore* Initializes a store client from data previously stored in the
 block identified by *aHandle* in *aStore*
- **freeFromStore** Removes the store client’s storage from the IXStore and
 frees the run-time object
- + **freeFromBlock:**(unsigned int)*aHandle*
 andStore:(IXStore *)*aStore* Frees the data for the store client identified by *aHandle* in
 aStore, without necessarily creating an instance

Retrieving the Block and Store

- **getBlock:**(unsigned int *)*aHandle*
 andStore:(IXStore **)*aStore* Gets the identifier of the block owned by the store client,
 and the IXStore that the block exists in

IXComparatorSetting

Adopted By: IXBTree

Methods

- **setComparator:**(IXComparator *)*aComparator*
 andContext:(const void *)*aContext* Sets the function used to compare items; also
 provides *aContext* as arbitrary data to use in comparison
- **getComparator:**(IXComparator **)*aComparator*
 andContext:(const void **)*aContext* Gets the comparator function and context

IXComparisonSetting

Adopted By: IXBTree

Methods

- **setComparisonFormat:(const char *)format** Sets the data format of items compared by the receiver
- **(const char *)comparisonFormat** Returns the data format of items compared

IXCursorPositioning

Adopted By: IXBTreeCursor

Absolute Positioning

- **(BOOL)setKey:(void *)aKey
andLength:(unsigned int)aLength** Sets the position of the receiver to *aKey*, if it exists, otherwise to where *aKey* would logically be; returns YES if *aKey* exists
- **(BOOL)getKey:(void **)aKey
andLength:(unsigned int *)aLength** Gets the key for the receiver's position and its length, sliding the receiver forward in the key space if needed and if possible; returns YES if the receiver ends up on a key

Relative Positioning

- **(BOOL)setFirst** Positions the receiver at the first key if there is one; returns YES if there is one, NO if not
- **(BOOL)setNext** Moves the receiver forward one key value and returns YES if there's a key there
- **(BOOL)setLast** Positions the receiver at the last key if there is one; returns YES if there is one, NO if not
- **(BOOL)setPrevious** Moves the receiver back one key value and returns YES if there's a key there

Checking Positioning Success

- (BOOL)isMatch Returns YES if the receiver is on a key, NO if it's between two keys or off either end of the key space

IXFileFinderConfiguration

Adopted By: IXFileFinder

Managing Attribute Parsers

- setAttributeParsers:(List *)aList Sets the IXAttributeParsers used to parse files
- getAttributeParsers:(List *)aList Returns in aList the IXAttributeParsers used to parse files

Generating Descriptions

- setGeneratesDescriptions:(BOOL)flag Sets whether descriptions are generated automatically for files indexed
- (BOOL)generatesDescriptions Returns whether descriptions are generated automatically for files indexed

Enabling Automatic Updating

- setUpdatesAutomatically:(BOOL)flag Sets whether the file finder automatically updates its indexes upon finding out of date references
- (BOOL)updatesAutomatically Returns whether the file finder automatically updates its indexes

Setting File System Options

- setCrossesDeviceChanges:(BOOL)flag Sets whether the file finder indexes or searches files on a different device from its root directory
- (BOOL)crossesDeviceChanges Returns whether the file finder indexes or searches files on a different device from its root directory
- setFollowsSymbolicLinks:(BOOL)flag Sets whether the file finder follows symbolic links when building indexes
- (BOOL)followsSymbolicLinks Returns whether the file finder follows symbolic links when building indexes

- **setScansForModifiedFiles:(BOOL)*flag*** Sets whether the file finder scans for files whose modification times have changed
- (BOOL)**scansForModifiedFiles** Returns whether the file finder scans for modified files

Ignoring Files

- **setIgnoredTypes:(const char *)*types*** Sets to *types* the types of files that won't be indexed
- (char *)**ignoredTypes** Returns the types of files that aren't indexed
- **setIgnoredNames:(const char *)*names*** Sets to *names* the literal, base names of files that won't be indexed
- (char *)**ignoredNames** Returns the names of files that aren't indexed

IXFileFinderQueryAndUpdate

Adopted By: IXFileFinder

Getting the Target Directory

- (const char *)**rootPath** Returns the base path for the file finder's index

Getting the Record Manager

- **recordManager** Returns the object that stores the file finder's IXFileRecords

Performing Queries

- (IXPostingList *)**performQuery:(const char *)*aQuery***
 - atPath:(const char *)*path*** Evaluates *aQuery* for *sender* returning in an IXPostingList the IXFileRecords that match
 - forSender:*sender***
- **stopQueryForSender:*sender*** Stops the query requested by *sender*

Updating Indexes

- **updateIndexPath:(const char *)path
forSender:sender** Updates the indexes for files within *path* relative to the file finder's root path
- **(BOOL)isUpdating** Returns whether the file finder is updating its indexes
- **suspendUpdating** Suspends updating of indexes
- **resumeUpdating** Resumes updating of indexes
- **clean** Removes inaccurate or out of data information from indexes
- **reset** Completely empties indexes

Methods Implemented by the Sender of a Query or Update

- **fileFinder:(IXFileFinder *)aFinder
didFindFile:(IXFileRecord *)aRecord** Asynchronously notifies the sender of a **performQuery:atPath:forSender:** message that *aRecord* matches the query
- **fileFinder:(IXFileFinder *)aFinder
didFindList:(IXPostingList *)aList** Asynchronously notifies the sender of an **performQuery:atPath:forSender:** message that the IXFileRecords in *aList* match the query
- **fileFinder:(IXFileFinder *)aFinder
willAddFile:(IXFileRecord *)aRecord** Asynchronously notifies the sender of an **updateIndexPath:forSender:** message that *aRecord* is about to be added to the index

IXLexemeExtraction

Adopted By: No NeXTSTEP classes.

Lexing a Stream

- **(unsigned int)getLexeme:(char *)aString
inLength:(unsigned int)aLength
fromStream:(NXStream *)stream** Puts the next lexeme from *stream* into *aString*

Manipulating a Word/Lexeme

- **(unsigned int)foldCase:(char *)aString
inLength:(unsigned int)aLength** Reduces *aString* to lowercase letters

IXNameAndFileAccess

Adopted By: IXBTree,
IXFileFinder,
IXRecordManager,
IXStoreDirectory

Incorporates: IXBlockAndStoreAccess

Initializing and Freeing a Client

- **initWithName:**(const char *)*aName*
inFile:(const char *)*filename* Initializes a new store client under *aName* in *filename*
- **initWithName:**(const char *)*aName*
inFile:(const char *)*filename*
forWriting:(BOOL)*flag* Initializes a store client from data previously stored under *aName* in *filename*; if *flag* is YES, changes can be written back to the file
- **freeFromStore** Removes the store client's storage from the IXStoreFile and frees the run-time object
- + **freeFromName:**(const char *)*aName*
andFile:(const char *)*filename* Frees the data for the store file client identified by *aName* in *filename*, without necessarily creating an instance

Retrieving the Name and File

- **getName:**(const char **)*aName*
andFile:(const char **)*filename* Gets the name of the store client, and the name of the file that the data exists in

IXPostingExchange

Adopted By: IXPostingCursor
IXPostingList
IXPostingSet

Methods

- **setCount:**(unsigned int)*count*
andPostings:(IXPosting *)*postings* Sets the receiver's posting set to *count postings*

– **getCount**:(unsigned int *)*count*
 andPostings:(IXPosting **)*thePostings*

Gets the receiver’s postings and their amount

IXPostingOperations

Adopted By: IXPostingCursor
IXPostingSet

Manipulating Postings by Handle

– (unsigned int)**addHandle**:(unsigned int)*aHandle* Adds a postings to the set of postings
 withWeight:(unsigned int)*aWeight*
– **removeHandle**:(unsigned int)*aHandle* Removes a postings from the set

Getting the Number of Postings

– (unsigned int)**count** Returns the number of postings in the set

Emptying a Posting Set

– **empty** Empties all postings from the set

Traversing a Posting Set

– (unsigned int)**setHandle**:(unsigned int)*aHandle* Sets the selected posting to the one with *aHandle* and
 returns that handle, or 0 if *aHandle* isn’t in the set
– (unsigned int)**getHandle**:(unsigned int *)*aHandle* Gets the handle and weight of the selected posting
 andWeight:(unsigned int *)*aWeight*
– (unsigned int)**setFirstHandle** Sets the selected posting to the first in the set and returns its
 handle, or 0 if there are no postings
– (unsigned int)**setNextHandle** Sets the selected posting to the next in the set and returns
 its handle, or 0 if there are no more postings

IXRecordDiscarding

Adopted By: IXRecordManager

Discarding Records

- **discardRecord:(unsigned int)aHandle** Discards the records identified by *aHandle*
- **reclaimRecord:(unsigned int)aHandle** Reclaims the discarded record identified by *aHandle*

Removing Discarded Records

- **clean** Permanently deletes all discarded records

IXRecordReading

Adopted By: No NeXTSTEP classes

Methods

- **(unsigned int)count** Returns the number of records in the archive
- **readRecord:(unsigned int)aHandle
fromZone:(NXZone *)zone** Reads the record identified by *aHandle* and returns the corresponding object allocated from *zone*

IXRecordTranscription

Adopted By: No NeXTSTEP classes.

Methods

- **source:***aTranscriber*
didReadRecord:(unsigned int)aHandle Notifies the record identified by *aHandle* that it's been read
- **source:***aTranscriber*
willWriteRecord:(unsigned int)aHandle Notifies the record identified by *aHandle* that it's going to be written
- **finishReading** Allows a record just read to reinitialize itself or provide a replacement

IXRecordWriting

Adopted By: IXRecordManager

Incorporates: IXRecordReading

Manipulating Records by Handle

- (unsigned int)**addRecord:anObject** Adds *anObject* to the receiver's archive
- **replaceRecord:(unsigned int)aHandle**
with:anObject Replaces the record identified by *aHandle* with *anObject*
- **removeRecord:(unsigned int)aHandle** Removes from the archive the record identified by *aHandle*

Emptying a Record Storer

- **empty** Empties the receiver's archive of all records

IXTransientAccess

Adopted By: IXRecordManager

Methods

- (BOOL)**getDoubleValue**:(double *)*aValue*
ofIvar:(const char *)*ivarName*
forRecord:(unsigned int)*aHandle* Retrieves as a **double** the value of *ivarName* for the object whose record is identified by *aHandle*
- (BOOL)**getFloatValue**:(float *)*aValue*
ofIvar:(const char *)*ivarName*
forRecord:(unsigned int)*aHandle* Retrieves as a **float** the value of *ivarName* for the object whose record is identified by *aHandle*
- (BOOL)**getIntValue**:(int *)*aValue*
ofIvar:(const char *)*ivarName*
forRecord:(unsigned int)*aHandle* Retrieves as a **int** the value of *ivarName* for the object whose record is identified by *aHandle*
- (BOOL)**getObjectValue**:(Object **)*aValue*
ofIvar:(const char *)*ivarName*
forRecord:(unsigned int)*aHandle* Retrieves as a object the value of *ivarName* for the object whose record is identified by *aHandle*
- (BOOL)**getOpaqueValue**:(NXData **)*aValue*
ofIvar:(const char *)*ivarName*
forRecord:(unsigned int)*aHandle* Retrieves as untyped data the value of *ivarName* for the object whose record is identified by *aHandle*
- (BOOL)**getStringValue**:(char **)*aValue*
ofIvar:(const char *)*ivarName*
forRecord:(unsigned int)*aHandle* Retrieves as a string the value of *ivarName* for the object whose record is identified by *aHandle*
- (BOOL)**getStringValue**:(char **)*aValue*
inLength:(unsigned int)*aLength*
ofIvar:(const char *)*ivarName*
forRecord:(unsigned int)*aHandle* Retrieves as a string no longer than *aLength* the value of *ivarName* for the object whose record is identified by *aHandle*

IXTransientMessaging

Adopted By: IXRecordManager

Methods

- (BOOL) **getDoubleValue:**(double *)*aValue*
 ofMessage:(SEL)*aSelector*
 forRecord:(unsigned int)*aHandle*
Retrieves as a **double** the value obtained by sending *aSelector* to the object whose record is identified by *aHandle*
- (BOOL) **getFloatValue:**(float *)*aValue*
 ofMessage:(SEL)*aSelector*
 forRecord:(unsigned int)*aHandle*
Retrieves as a **float** the value obtained by sending *aSelector* to the object whose record is identified by *aHandle*
- (BOOL) **getIntValue:**(int *)*aValue*
 ofMessage:(SEL)*aSelector*
 forRecord:(unsigned int)*aHandle*
Retrieves as a **int** the value obtained by sending *aSelector* to the object whose record is identified by *aHandle*
- (BOOL) **getObjectValue:**(Object **)*aValue*
 ofMessage:(SEL)*aSelector*
 forRecord:(unsigned int)*aHandle*
Retrieves as an object the value obtained by sending *aSelector* to the object whose record is identified by *aHandle*
- (BOOL) **getOpaqueValue:**(NXData **)*aValue*
 ofMessage:(SEL)*aSelector*
 forRecord:(unsigned int)*aHandle*
Retrieves as untyped data the value obtained by sending *aSelector* to the object whose record is identified by *aHandle*
- (BOOL) **getStringValue:**(char **)*aValue*
 ofMessage:(SEL)*aSelector*
 forRecord:(unsigned int)*aHandle*
Retrieves as a string the value obtained by sending *aSelector* to the object whose record is identified by *aHandle*
- (BOOL) **getStringValue:**(char **)*aValue*
 inLength:(unsigned int)*aLength*
 ofMessage:(SEL)*aSelector*
 forRecord:(unsigned int)*aHandle*
Retrieves as a string no longer than *aLength* the value obtained by sending *aSelector* to the object whose record is identified by *aHandle*

Functions

Comparator Functions

Compare two sets of data as arrays of various types and return their ordering.

int	IXCompareBytes (const void * <i>data1</i> , unsigned short <i>length1</i> , const void * <i>data2</i> , unsigned short <i>length2</i> , const void * <i>context</i>)
int	IXCompareUnsignedBytes (const void * <i>data1</i> , unsigned short <i>length1</i> , const void * <i>data2</i> , unsigned short <i>length2</i> , const void * <i>context</i>)
int	IXCompareShorts (const void * <i>data1</i> , unsigned short <i>length1</i> , const void * <i>data2</i> , unsigned short <i>length2</i> , const void * <i>context</i>)
int	IXCompareUnsignedShorts (const void * <i>data1</i> , unsigned short <i>length1</i> , const void * <i>data2</i> , unsigned short <i>length2</i> , const void * <i>context</i>)
int	IXCompareLongs (const void * <i>data1</i> , unsigned short <i>length1</i> , const void * <i>data2</i> , unsigned short <i>length2</i> , const void * <i>context</i>)
int	IXCompareUnsignedLongs (const void * <i>data1</i> , unsigned short <i>length1</i> , const void * <i>data2</i> , unsigned short <i>length2</i> , const void * <i>context</i>)
int	IXCompareFloats (const void * <i>data1</i> , unsigned short <i>length1</i> , const void * <i>data2</i> , unsigned short <i>length2</i> , const void * <i>context</i>)
int	IXCompareDoubles (const void * <i>data1</i> , unsigned short <i>length1</i> , const void * <i>data2</i> , unsigned short <i>length2</i> , const void * <i>context</i>)
int	IXCompareShort (const void * <i>data1</i> , unsigned short <i>length1</i> , const void * <i>data2</i> , unsigned short <i>length2</i> , const void * <i>context</i>)
int	IXCompareUnsignedShort (const void * <i>data1</i> , unsigned short <i>length1</i> , const void * <i>data2</i> , unsigned short <i>length2</i> , const void * <i>context</i>)
int	IXCompareLong (const void * <i>data1</i> , unsigned short <i>length1</i> , const void * <i>data2</i> , unsigned short <i>length2</i> , const void * <i>context</i>)
int	IXCompareUnsignedLong (const void * <i>data1</i> , unsigned short <i>length1</i> , const void * <i>data2</i> , unsigned short <i>length2</i> , const void * <i>context</i>)
int	IXCompareFloat (const void * <i>data1</i> , unsigned short <i>length1</i> , const void * <i>data2</i> , unsigned short <i>length2</i> , const void * <i>context</i>)
int	IXCompareDouble (const void * <i>data1</i> , unsigned short <i>length1</i> , const void * <i>data2</i> , unsigned short <i>length2</i> , const void * <i>context</i>)
int	IXCompareStringAndUnsigneds (const void * <i>data1</i> , unsigned short <i>length1</i> , const void * <i>data2</i> , unsigned short <i>length2</i> , const void * <i>context</i>)
int	IXCompareUnsignedAndStrings (const void * <i>data1</i> , unsigned short <i>length1</i> , const void * <i>data2</i> , unsigned short <i>length2</i> , const void * <i>context</i>)
int	IXCompareStrings (const void * <i>data1</i> , unsigned short <i>length1</i> , const void * <i>data2</i> , unsigned short <i>length2</i> , const void * <i>context</i>)
int	IXCompareMonocaseStrings (const void * <i>data1</i> , unsigned short <i>length1</i> , const void * <i>data2</i> , unsigned short <i>length2</i> , const void * <i>context</i>)

Comparison Format Functions

Compare two arrays of data based on a type encoding and return their ordering.

```
int          IXFormatComparator(const void *data1, unsigned short length1,  
                                const void *data2, unsigned short length2, void *format)
```

IXBTree Locking Macros

Lock and unlock an IXBTree for thread-safe access

```
void          IXLockBTreeMutex(IXBTree *aBTree)  
void          IXUnlockBTreeMutex(IXBTree *aBTree)
```

IXStoreBlock Archiving Functions

Archive or unarchive an object to or from an IXStore

```
unsigned int  IXWriteRootObjectToStore(IXStore *aStore, unsigned int aHandle, id anObject)  
id           IXReadObjectFromStore(IXStore *aStore, unsigned int aHandle,  
                                   NXZone *aZone)
```

Types and Constants

Defined Types

IXComparator

```
typedef int IXComparator  
    (const void *data1,  
     unsigned short length1,  
     const void *data2,  
     unsigned short length2,  
     const void *context);
```

IXPosting

```
typedef struct IXPosting {  
    unsigned handle;  
    unsigned weight;  
} IXPosting;
```

IXStoreErrorType

```
typedef enum IXStoreErrorType {  
    IX_NoError = IX_STOREUSERERRORBASE,  
    IX_InternalError,  
    IX_ArgumentError,  
    IX_QueryEvalError,  
    IX_QueryTypeError,  
    IX_QueryAttrError,  
    IX_QueryImplError,  
    IX_QueryYaccError,  
    IX_MemoryError,  
    IX_LockedError,  
    IX_MachineError,  
    IX_VersionError,  
    IX_DamagedError,  
    IX_DuplicateError,  
    IX_NotFoundError,  
    IX_TooLargeError,  
    IX_UnixErrorBase = IX_STOREUNIXERRBASE,  
    IX_MachErrorBase = IX_STOREMACHERRBASE,  
} IXStoreErrorType;
```

Weighting Types

```
typedef enum {  
    IX_NoWeighting = 0,  
    IX_AbsoluteWeighting,  
    IX_FrequencyWeighting,  
    IX_PeculiarityWeighting  
} IXWeightingType;
```

Symbolic Constants

IXStore Constants

IX_ALLBLOCKS ((unsigned int)-1L)

Indexing Kit Error Base Constants

IX_STOREUSERERRBASE	(9000)
IX_STOREMACHERRBASE	IX_STOREUSERERRBASE + (100)
IX_STOREUNIXERRBASE	IX_STOREUSERERRBASE + (300)

Global Variables

IXStore Pasteboard Type

NXAtom IXStorePboardType;

Indexing Pasteboard Types

NXAtom IXAttributeReaderPboardType;
NXAtom IXFileDescriptionPboardType;

Query Language Symbols and Operators

Literals

self	Refers to the object the expression is being evaluated for
yes or true	Boolean values for TRUE
no or false	Boolean values for FALSE
"aString" or 'aString'	Declares <i>aString</i> a string literal
<i>number</i>	Any legal integer or floating point number
<i>AttributeName</i>	An attribute reference

Transform Operators

quote (<i>some text</i>)	Results in a string value equal to <i>some text</i>
regex (<i>a BSD regular expression</i>)	Results in a regular expression
shell (<i>an expression with shell wildcards</i>)	Results in a regular expression
parse (<i>some text</i>)	Parses the text argument into an attribute-value list

Projection Operators

project (<i>AttributeName entity_or_object</i>)	Projects the attribute named <i>AttributeName</i> from the set of attributes represented by <i>entity_or_object</i>
--	---

Boolean Operators

or (<i>a b</i>)	Results in true if either <i>a</i> or <i>b</i> is true , false if both are false
and (<i>a b</i>)	Results in true if both <i>a</i> and <i>b</i> are true , false if either is false
not (<i>a b</i>)	Equivalent to and (<i>a not(b)</i>)
not (<i>a</i>)	The logical negation of <i>a</i>

Set Operators

or (<i>s t</i>)	Results in the union of sets <i>s</i> and <i>t</i>
or (<i>s</i>)	"Any one or more of the items in set <i>s</i> " (for use with search operators only)
and (<i>s t</i>)	Results in the intersection of sets <i>s</i> and <i>t</i>
and (<i>s</i>)	"All of the items in set <i>s</i> " (for use with search operators only)
not (<i>s t</i>)	Results in a set containing those items in set <i>s</i> that aren't in set <i>t</i>

Relational Operators

gt(*a b*)

Results in **true** if *a* is greater than *b*, **false** otherwise

ge(*a b*)

Results in **true** if *a* is greater than or equal to *b*, **false** otherwise

eq(*a b*)

Results in **true** if *a* is equal to *b*, **false** otherwise

ne(*a b*)

Results in **true** if *a* is not equal to *b*, **false** otherwise

lt(*a b*)

Results in **true** if *a* is less than *b*, **false** otherwise

le(*a b*)

Results in **true** if *a* is less than or equal *b*, **false** otherwise

Arithmetic Operators

add(*a b*)

Results in the sum of *a* and *b*

sub(*a b*)

Results in difference of *a* and *b*

mul(*a b*)

Results in product of *a* and *b*

div(*a b*)

Results in quotient of *a* and *b*

neg(*a*)

Results in the arithmetic negative of *a*

Search Operators

whole(*value*)

Searches for *value* as an exact match in the Default attribute

whole(*AttributeName value*)

Searches for *value* as an exact match in the attribute named *AttributeName*

whole(*entity value*)

Searches for *value* as an exact match in *entity*; results in **true** if value is found, **false** if not

whole(*string value*)

Searches for *value* as an exact match to *string*; results in **true** if value is found, **false** if not (this is equivalent to **eq**(*string value*))

prefix()

Searches for a value as a prefix match; arguments may be as for **whole**()

within()

Searches for a value as a match anywhere within an attribute or string; arguments may be as for **whole**()

Pre-defined Attributes

Default

The default attribute defined by the query language

Content

For a file or file record, the literal text of the file

IXFileRecord Attributes

FileName	The name of a file relative to the root path of the IXFileFinder that created the IXFileRecord (string).
FileType	The file's type (string); for example, "rtf".
FileDevice	The device number for the device the file is on, as returned by stat() (number).
FileInode	The inode number of the file (number).
FileMode	The file's permissions (number).
FileCount	The number of hard links to the file (number).
FileOwner	The file's owner (string).
FileGroup	The file's group (string).
FileSize	The file's size, in bytes (number).
AccessTime	The time the file was last accessed, as returned by stat() (number).
ModifyTime	The time the file's content was last modified, as returned by stat() (number).
ChangeTime	The time the file's status information was last changed, as returned by stat() (number).
UnixType	The inode type for the file, encoded as a number.

Attribute Reader Format

Declaring Attributes

`{\zd AttributeName}`

`{\zd AttributeName\zttypeEncoding}`

Declare an attribute named *AttributeName*

Declares the preceding attribute to be of type *typeEncoding*

Marking Lexemes

`{\z lexeme}`

`{\zrn}`

Mark *lexeme* as a lexeme associated with the Default attribute

Short reference equivalent to the n^{th} lexeme encountered

Adding Information to Lexeme Markings

`{\z lexeme\zan}`

`{\z lexeme\zwweight}`

`{\z lexeme\zccookie}`

Associate *lexeme* with the n^{th} attribute declared

Marks *lexeme* as though it occurred *weight* times in the text

Declare *cookie* as an opaque identifier associated with the *lexeme*



Interface Builder

Classes

IBInspector

Inherits From: Object

Conforms To: IBInspectors

Accessing Objects

- **object** Returns the object that's being inspected
- **window** Returns the window that contains the user interface for the inspector

Managing Changes

- **touch:sender** Changes the image in the Inspector panel's close box to a broken "X"
- **textDidChange:sender** Sends the IBInspector a touch: message

IBPalette

Inherits From: Object

Associating Views and Objects

– **associateObject:***anObject*
 type:(NXAtom)*type*
 with:*aView*

Establishes an association between *aView* and the object that should be instantiated when the user drags the *aView* from the palette.

Initializing the Palette

– **finishInstantiate**

Implement to complete the initialization of your IBPalette object

Accessing Related Objects

– **paletteDocument**

Returns an object representing the dynamically loaded palette

– **originalWindow**

Returns the Window that contains the objects to be loaded into Palette window

– **findImageNamed:**(const char *)*name*

Returns the NXImage instance associated with *name*

Object Additions

Interface Builder declares these methods as additions to the Object class.

Identifying Inspectors, Editors, and Images

- (const char *)**getConnectionInspectorClassName** Implement to return class name of Connection inspector
- (const char *)**getEditorInspectorClassName** Implement to return class name of the object's editor
- (const char *)**getHelpInspectorClassName** Implement to return class name of the Help inspector
- (NXImage *)**getIBImage** Implement to return an image to represent the object in the File window.
- (const char *)**getInspectorClassName** Implement to return class name of the Attributes inspector
- (const char *)**getSizeInspectorClassName** Implement to return class name of the Size inspector

View Additions

Interface Builder declares these methods as additions to the Application Kit's View class.

Controlling Size

- **getMinSize:**(NXSize *)*minSize*
maxSize:(NXSize *)*maxSize*
from:(int)*where* Implement this method to control the dimensions of a View
- **placeView:**(NXRect *)*frameRect* Notifies a View of a change in its frame size

Protocols

IB

Adopted By: Interface Builder's subclass of the Application class

Accessing the Document

– **activeDocument** Returns the active document

Accessing the Selection Owner

– **selectionOwner** Returns the editor of the currently selected object

Managing Connections

– **connectSource** Returns the object that's the source of the connection
– **connectDestination** Returns the object that's the destination of the connection
– **(BOOL)isConnecting** Returns YES if connection lines are being displayed
– **stopConnecting** Removes any connection lines from the screen
– **displayConnectionBetween:source
and:destination** Causes Interface Builder to draw connection lines between
source and *destination*

Querying the Mode

– **(BOOL)isTestingInterface** Returns YES if Interface Builder is in Test mode

Registering Controllers

– **registerDocumentController:aController** Adds *aController* to the list of objects to be notified when documents are opened or saved
– **unregisterDocumentController:** Removes *aController* from the list of document controllers

IBConnectors

Adopted By: Connector objects

Connector Methods

- **destination** Implement to return the object that's the destination of the connection
- **establishConnection** Implement to connect the source and destination objects
- **free** Implement to release the storage for the connector object
- **nibInstantiate** Implement to verify the identities of the connector's source and destination objects
- **read:(NXTypedStream *)stream** Implement to unarchive the connector object from *stream*
- **renewObject:old
to:new** Implement to update a connector by replacing its old source or destination object with a new object
- **source** Implement to return the object that's the source of the connection
- **write:(NXTypedStream *)stream** Implement to archive the connector object to *stream*

IBDocuments

Adopted By: Interface Builder's document objects

Managing the Document

- **touch** Marks the document as edited
- **getDocumentPathIn:(char *)buffer** Places the document's path in *buffer*

Managing the Object Hierarchy

- **attachObject:anObject
to:parent** Adds *anObject* to the hierarchy by attaching it to *parent*
- **attachObjects:(List *)objectList
to:parent** Adds the objects in *objectList* to the hierarchy by attaching them to *parent*
- **deleteObject:anObject** Removes *anObject* from the object hierarchy

- **deleteObjects:**(List *)*objectList* Removes objects in *objectList* from the object hierarchy
- **copyObject:***anObject* Copies *anObject* to the specified pasteboard
 type:(NXAtom)*type*
 inPasteboard:(Pasteboard *)*aPasteboard*
- **copyObjects:**(List *)*objectList* Copies the objects in *objectList* to the specified pasteboard
 type:(NXAtom)*type*
 inPasteboard:(Pasteboard *)*aPasteboard*
- (List *)**pasteType:**(NXAtom)*type* Alerts the document object that objects were pasted
 fromPasteboard:(Pasteboard *)*pboard*
 parent:*theParent*
- (BOOL)**objectIsMember:***anObject* Returns YES if *anObject* is a part of the object hierarchy
- **getObjects:**(List *)*objectList* Places the objects from the object hierarchy into *objectList*
- **getParentForObject:***anObject* Returns the object above *anObject* in the object hierarchy

Setting Object Names

- (BOOL)**setName:**(const char *)*name* Sets the name associated with the *anObject*
 for:*anObject*
- **getNameIn:**(char *)*name* Places the name associated with *anObject* in the buffer
 for:*anObject*
 name

Managing Connectors

- **addConnector:***aConnector* Adds a connector object to Interface Builder’s list
- **removeConnector:***aConnector* Removes *aConnector* from the list of connectors.
- **listConnectors:**(List *)*aList* Places in *aList* connector objects whose sources
 forSource:*aSource* are *aSource*
- **listConnectors:**(List *)*aList* Places in *aList* connector objects whose destinations
 forDestination:*aDestination* are *aDestination*
- **listConnectors:**(List *)*aList* Places in *aList* the connector objects of class
 forSource:*aSource* *filterClass* whose sources are *aSource*
 filterClass:*filterClass*
- **listConnectors:**(List *)*aList* Places in *aList* the connector objects of class
 forDestination:*aDestination* *filterClass* whose destinations are *aDestination*
 filterClass:*filterClass*

Managing Editors

- **setSelectionFrom:***anEditor* Registers *anEditor* as the editor that owns the selection
- **editorDidClose:***anEditor* Informs the document object that *anEditor* is no longer
 for:*anObject* active

- **getEditor:(BOOL)createIt
for:anObject** Returns the editor object for *anObject*
- **(BOOL)openEditorFor:anObject** Opens the editor object for *anObject*

Updating the Display

- **redrawObject:anObject** Redraws the selected object by opening required editors

IBDocumentControllers

Adopted By: Document controller objects

Notification Methods

- **didOpenDocument:theDocument** Notifies the controller that *theDocument* has been opened
- **didSaveDocument:theDocument** Notifies the controller that *theDocument* has been saved
- **willSaveDocument:theDocument** Notifies the controller that the user is attempting to save *theDocument*

IBEditors

Adopted By: Editor objects in Interface Builder

Initializing

- **initWith:anObject
inDocument:aDocument** Implement to initialize a newly allocated editor

Identifying Objects

- **document** Implement this method to return the active document
- **editedObject** Implement to return the object that's being edited
- **window** Implement to return the editor window

Displaying Objects

- `resetObject:anObject` Implement to redraw *anObject*

Managing the Selection

- `(BOOL)wantsSelection` Implement to return YES if the editor is willing to become the selection owner
- `selectObjects:(List *)objectList` Implement to draw the objects in *objectList* as selected
- `makeSelectionVisible:(BOOL)showIt` Implement to make the current selection visible

Copying and Pasting Objects

- `(BOOL)copySelection` Implement to copy the selected object(s) to the pasteboard
- `(BOOL)deleteSelection` Implement to delete the selected object(s)
- `(BOOL)pasteInSelection` Implement to paste object(s) from pasteboard into selection
- `(NXAtom)acceptsTypeFrom:(const NXAtom *)typeList` Implement to return the pasteboard type your editor accepts

Opening and Closing Editors

- `close` Implement to close the editor and free its resources
- `openSubeditorFor:anObject` Implement to open the subeditor for *anObject*
- `closeSubeditors` Implement to close all subeditors

Activating the Editor

- `orderFront` Implement to bring the editor's window to the front
- `(BOOL)activate` Implement to activate the editor

IBInspectors

Adopted By: IBInspector

Required Inspector Methods

- `ok:sender` Implement in subclass to commit changes made in the Inspector panel

- **revert:sender**
- (BOOL)**wantsButtons**

Implement in subclass to load data into inspector's display
Returns whether the inspector requires Interface Builder to display OK and Revert buttons in the Inspector panel

IBObject (informal protocol)

Adopted By: Custom palette objects

Identifying Inspectors, Editors, and Images

- (const char *)**getConnectionInspectorClassName** Returns the class name of the receiver's connection inspector
- (const char *)**getEditorClassName** Returns the class name of the receiver's editor
- (const char *)**getHelpInspectorClassName** Returns the class name of the receiver's help inspector
- (NXImage *)**getIBImage** Returns the image that's displayed in the File window when an instance of this class is created
- (const char *)**getInspectorClassName** Returns the class name of the receiver's attributes inspector
- (const char *)**getSizeInspectorClassName** Returns the class name of the receiver's size inspector

IBSelectionOwners

Adopted By: Editor objects

Selection Methods

- **getSelectionInto:(List *)objectList** Implement to place the selected objects into *objectList*
- **redrawSelection** Implement to redraw the objects in the selection
- (unsigned)**selectionCount** Implement to return the number of objects in the editor's selection

Types and Constants

Symbolic Constants

Control Point Constants

IB_BOTTOMLEFT
IB_MIDDLELEFT
IB_TOPLEFT
IB_MIDDLETOP
IB_TOPRIGHT
IB_MIDDLERIGHT
IB_BOTTOMRIGHT
IB_MIDDLEBOTTOM

Global Variables

Pasteboard Types

NXAtom **IBObjectPboardType**;
NXAtom **IBCellPboardType**;
NXAtom **IBMenuPboardType**;
NXAtom **IBMenuCellPboardType**;
NXAtom **IBViewPboardType**;
NXAtom **IBWindowPboardType**;

9 *Mach Kit*

Classes

NXConditionLock

Inherits From: Object

Conforms To: NXLock

Initializing an Instance

- **init** Initialize a condition lock
- **initWith:(int)condition** Initialize and set condition

Get the Condition of the Lock

- **condition** Return the condition

Acquire or Release the Lock

- **lock** Grab the lock
- **lockWhen:(int)condition** Wait for the condition
- **unlock** Release the lock
- **unlockWith:(int)condition** Release and set the condition

NXData

Inherits From: Object

Conforms To: NXTransport

Transporting Data

- **encodeRemotelyFor:** (NXConnection *)*connection*
freeAfterEncoding:(BOOL *)*flagp*
isBycopy:(BOOL)*isBycopy* Has the data copied across a connection
- **encodeUsing:**(id <NXEncoding>)*portal* Encodes the data across a connection
- **decodeUsing:**(id <NXDecoding>)*portal* Encodes the data across a connection

Initializing and Freeing Instances

- **initWithSize:**(unsigned int) *size* Allocates memory for data
- **initWithData:**(void *)*data*
size:(unsigned) *size*
dealloc:(BOOL) *flag* Wraps preexisting data
- **free** Frees the object

Getting the Object's Data

- **data** Returns the data

Getting the Data's Size

- **size** Returns the size of the data

Copying the Object

- **copyFromZone:**(NXZone *)*zone* Copies the object

NXInvalidationNotifier

Inherits From: Object

Conforms To: NXReference

Counting References

- **addReference** Adds a reference
- **free** Removes a reference, but doesn't free object
- **references** Returns number of references

Initializing a New Object

- **init** Initializes an instance

Really Freeing an Object

- **deallocate** Frees an instance

Getting and Setting Validity

- **invalidate** Marks the object as invalid
- **isValid** Returns whether object is valid

Invalidation Notification

- **registerForInvalidationNotification:**
(id <NXSenderIsInvalid>)anObject Ensures object will be notified when receiver dies
- **unregisterForInvalidationNotification:**
(id <NXSenderIsInvalid>)anObject Removes object for notification list

NXLock

Inherits From: Object

Conforms To: NXLock

Acquire or Release a Lock

- lock Grab the lock
- unlock Release the lock

NXNetNameServer

Inherits From: Object

Making a Port Available

- + **checkInPort:**(NXPort *)*nxport*
withName:(const char *)*aName* Advertises a port

Removing a Port

- + **checkOutPortWithName:**
(const char *)*name* Withdraws a port from public display

Getting Ports

- + (NXPort *) **lookUpPortWithName:**
(const char *)*name* Finds a port on the local host
- + (NXPort *) **lookUpPortWithName:**
(const char *)*name*
onHost:(const char *)*hostname* Finds a port on a named host

NXPort

Inherits From: NXInvalidationNotifier : Object

Conforms To: NXReference (NXInvalidationNotifier)
NXTransport

Transporting Ports

- **encodeRemotelyFor:** Has the port copied across a connection
(NXConnection *)*connection*
freeAfterEncoding:(BOOL *)*flagp*
isBycopy:(BOOL)*isBycopy*
- **encodeUsing:**(id <NXEncoding>)*portal* Encodes the port across a connection
- **decodeUsing:**(id <NXDecoding>)*portal* Encodes the port across a connection

Creating an NXPort

- + **new** Creates a new NXPort object
- + **newFromMachPort:**(port_t) *p* Wraps an existing port
- + **newFromMachPort:**(port_t) *p* Wraps an existing port
dealloc:(BOOL) *flag*

Freeing an NXPort

- **free** Frees an NXPort object

Listening for Port Deaths

- + **worryAboutPortInvalidation** Forks a thread to listen for port deaths

Identifying the Mach Port

- **machPort** Returns the embedded mach port
- **hash** Returns a hashtable value for the object

NXSpinLock

Inherits From: Object

Conforms To: NXLock

Acquire or Release a Lock

- lock Grab the lock
- unlock Release the lock

Protocols

NXLock

Adopted By: NXConditionLock
NXLock
NXSpinLock
NXRecursiveLock

Declared In: machkit/NXLock.h

Mandatory methods

- lock Acquires the lock
- unlock Releases the lock

NXReference

Adopted By: IXFileFinder
IXStoreBlock
NXConnection
NXInvalidationNotifier
NXProxy

Declared In: machkit/reference.h

Mandatory methods

- addReference Adds a reference
- free Removes a reference, possibly freeing object
- references Returns number of references

NXSenderIsInvalid

Adopted By: NXConnection
NXDataLinkManager

Declared In: machkit/senderIsInvalid.h

Mandatory methods

– **senderIsInvalid:***sender* Sent when *sender* becomes dysfunctional

Types and Constants

Defined Types

NXMachKitException

```
typedef enum {  
    NX_MACH_KIT_EXCEPTION_BASE = 10000,  
    NX_portInvalidException = 10001,  
    NX_restrictionEnforcedException = 10010,  
    NX_referenceAlreadyFreeException = 10020,  
    NX_MACH_KIT_LAST_EXCEPTION = 10999  
} NXMachKitException
```

10 *MIDI Driver API*

Driver Functions

Clock Functions

Functions to set clock behavior:

```
kern_return_t    MIDISetClockMode(port_t driverPort, port_t ownerPort, short synchUnit,  
                                  int mode)  
kern_return_t    MIDISetClockQuantum(port_t driverPort, port_t ownerPort, int interval)
```

Functions to set and get clock time:

```
kern_return_t    MIDISetClockTime(port_t driverPort, port_t ownerPort, int time)  
kern_return_t    MIDIGetClockTime(port_t driverPort, port_t ownerPort, int *time)  
kern_return_t    MIDIGetMTCTime(port_t driverPort, port_t ownerPort, short *format,  
                                  short *hours, short *minutes, short *seconds, short *frames)
```

Functions to start and stop the clock:

```
kern_return_t    MIDIStartClock(port_t driverPort, port_t ownerPort)  
kern_return_t    MIDIStopClock(port_t driverPort, port_t ownerPort)
```

Data Sending Function

Send data via the MIDI driver:

```
kern_return_t    MIDISendData(port_t driverPort, port_t ownerPort, short unit,  
                             MIDIRawEvent *data, unsigned int count)
```

Driver Ownership Functions

Acquire and release ownership of the MIDI driver:

```
kern_return_t    MIDIBecomeOwner(port_t driverPort, port_t ownerPort)  
kern_return_t    MIDIReleaseOwnership(port_t driverPort, port_t ownerPort)
```

Ignore MIDI Codes Function

Request that the driver ignore certain MIDI codes:

```
kern_return_t    MIDISetSystemIgnores(port_t driverPort, port_t ownerPort, short unit,  
                                       unsigned int ignoreBits)
```

Queue Management Functions

Query about and manage data flow in the queue:

```
kern_return_t    MIDIClearQueue(port_t driverPort, port_t ownerPort, short unit)  
kern_return_t    MIDIFlushQueue(port_t device_port, port_name_t ownerPort_port, short unit)  
kern_return_t    MIDIGetAvailableQueueSize(port_t driverPort, port_t ownerPort, short unit,  
                                           int *theSize)
```

Reply Handling Functions

Handle replies from the MIDI driver to a client:

```
kern_return_t    MIDIAwaitReply(port_t reply_port, MIDIReplyFunctions *handlers, int timeout)
kern_return_t    MIDIHandleReply(msg_header_t *msg, MIDIReplyFunctions *handlers)
```

Request Functions

Request services from the MIDI driver:

```
kern_return_t    MIDIRequestData(port_t driverPort, port_t ownerPort, short unit,
                                port_t replyPort)
kern_return_t    MIDIRequestAlarm(port_t driverPort, port_t ownerPort, port_t replyPort,
                                int time)
kern_return_t    MIDIRequestExceptions(port_t driverPort, port_t ownerPort, port_t replyPort)
kern_return_t    MIDIRequestQueueNotification(port_t driverPort, port_t ownerPort, short unit,
                                port_t replyPort, int size)
```

Serial Port Ownership Functions

Acquire and release ownership of the serial ports:

```
kern_return_t    MIDIClaimUnit(port_t driverPort, port_t ownerPort, short unit)
kern_return_t    MIDIReleaseUnit(port_t driverPort, port_t ownerPort, short unit)
```

Types and Constants

Defined Types

MIDIAlarmReplyFunction

```
typedef void (*MIDIAlarmReplyFunction)(port_t replyPort, int requestedTime,  
int actualTime)
```

MIDIDataReplyFunction

```
typedef void (*MIDIDataReplyFunction)(port_t replyPort, short unit, MIDIRawEvent *events,  
unsigned int count)
```

MIDIExceptionReplyFunction

```
typedef void (*MIDIExceptionReplyFunction)(port_t replyPort, int exception)
```

MIDIQueueReplyFunction

```
typedef void (*MIDIQueueReplyFunction)(port_t replyPort, short unit)
```

MIDIRawEvent

```
typedef struct {  
    int time;  
    unsigned char byte;  
} MIDIRawEvent
```

MIDIReplyFunctions

```
typedef struct _MIDIReplyFunctions {  
    MIDIDataReplyFunction dataReply;  
    MIDIAlarmReplyFunction alarmReply;  
    MIDIExceptionReplyFunction exceptionReply;  
    MIDIQueueReplyFunction queueReply;  
} MIDIReplyFunctions;
```

Symbolic Constants

Clock Modes

MIDI_CLOCK_MODE_INTERNAL
MIDI_CLOCK_MODE_MTC_SYNC

Controller Definitions

MIDI_EXTERNALEFFECTSDEPTH
MIDI_TREMELODEPTH
MIDI_CHORUSDEPTH
MIDI_DETUNEDEPTH
MIDI_PHASERDEPTH
(from original 1.0 MIDI spec)

MIDI_EFFECTS1
MIDI_EFFECTS2
MIDI_EFFECTS3
MIDI_EFFECTS4
MIDI_EFFECTS5
MIDI_DATAINCREMENT
MIDI_DATADECREMENT
(From June 1990 spec)

Error Codes

MIDI_ERROR_BUSY
MIDI_ERROR_NOT_OWNER
MIDI_ERROR_QUEUE_FULL
MIDI_ERROR_BAD_MODE
MIDI_ERROR_UNIT_UNAVAILABLE
MIDI_ERROR_ILLEGAL_OPERATION
MIDI_ERROR_UNKNOWN_ERROR

Event Count	Value
MIDI_MAX_EVENT	100

Event Size	Value
MIDI_MAX_MSG_SIZE	1024

Exception Codes

MIDI_EXCEPTION_MTC_STOPPED
MIDI_EXCEPTION_MTC_STARTED_FORWARD
MIDI_EXCEPTION_MTC_STARTED_REVERSE

General MIDI Constants

MIDI_RESETCONTROLLERS
MIDI_LOCALCONTROL
MIDI_ALLNOTESOFF
MIDI_OMNIOFF
MIDI_OMNION
MIDI_MONO
MIDI_POLY
MIDI_NOTEOFF
MIDI_NOTEON
MIDI_POLYPRES
MIDI_CONTROL
MIDI_PROGRAM
MIDI_CHANPRES
MIDI_PITCH
MIDI_CHANMODE
MIDI_CONTROL
MIDI_SYSTEM
MIDI_SYSEXCL
MIDI_TIMECODEQUARTER
MIDI_SONGPOS
MIDI_SONGSEL
MIDI_TUNEREQ
MIDI_EOX
MIDI_CLOCK
MIDI_START
MIDI_CONTINUE
MIDI_STOP
MIDI_ACTIVE
MIDI_RESET
MIDI_MAXDATA
MIDI_MAXCHAN
MIDI_NUMCHANS
MIDI_NUMKEYS
MIDI_ZEROBEND
MIDI_DEFAULTVELOCITY

Ignores

MIDI_IGNORE_CLOCK
MIDI_IGNORE_START
MIDI_IGNORE_CONTINUE
MIDI_IGNORE_STOP
MIDI_IGNORE_ACTIVE
MIDI_IGNORE_RESET
MIDI_IGNORE_REAL_TIME

Least Significant Bit for Controller Numbers

MIDI_MODWHEELLSB
MIDI_BREATHLSB
MIDI_FOOTLSB
MIDI_PORTAMENTOTIMELSB
MIDI_DATAENTRYLSB
MIDI_MAINVOLUMELSB
MIDI_BALANCELSB
MIDI_PANLSB
MIDI_EXPRESSIONLSB
MIDI_DAMPER
MIDI_PORTAMENTO
MIDI_SOSTENUTO
MIDI_SOFTPEDAL
MIDI_HOLD2

Masks for MIDI Status Bytes

MIDI_STATUSBIT
MIDI_STATUSMASK
MIDI_SYSRTBIT

Miscellaneous

MIDI_NO_TIMEOUT

MIDI Controller Numbers

MIDI_MODWHEEL
MIDI_BREATH
MIDI_FOOT
MIDI_PORTAMENTOTIME
MIDI_DATAENTRY
MIDI_MAINVOLUME
MIDI_BALANCE
MIDI_PAN
MIDI_EXPRESSION
MIDI_EFFECTCONTROL1
MIDI_EFFECTCONTROL2

Port Constants

MIDI_PORT_A_UNIT
MIDI_PORT_B_UNIT

11 *NetInfo Kit*

Classes

NIDomain

Inherits From: Object

Allocating and Initializing an NIDomain Object

- | | |
|--------------------------------|--|
| + alloc | Allocates an NIDomain object |
| + allocFromZone:(NXZone *)zone | Allocates an NIDomain object from the specified zone |
| - init | Initializes a new NIDomain object |

Freeing an NIDomain Object

- | | |
|--------|--------------------------------|
| - free | Deallocates an NIDomain object |
|--------|--------------------------------|

Making or Freeing a Connection to a Domain

- | | |
|--|--|
| -(ni_status)setConnection:(const char *)domain | Makes a connection to a domain |
| -(ni_status)setConnection:(const char *)domain
readTimeout:(int)rtime
writeTimeout:(int)wtime
canAbort:(BOOL)abort
mustWrite:(BOOL)write | Makes a connection to a domain, as specified |

- (ni_status)setTaggedConnection:(const char *)tag
to:(char *)hostName Makes a connection to a domain, using host name and tag
- (ni_status)setTaggedConnection:(const char *)tag
to:(char *)hostName
readTimeout:(int)rtime Makes a connection to a domain, as specified
writeTimeout:(int)wtime
canAbort:(BOOL)abort
- disconnectFromCurrent Terminates a connection to a domain

Getting Data about or from the Current Domain

- (const char *)getFullPath Gets the path name of the current domain
- (const char *)getMasterServer Gets the host name of the current domain's master server
- (const char *)getCurrentServer Gets the host name of the current domain's current server
- (const char *)getTag Gets the tag of the current domain
- (const struct sockaddr_in *)getServerIPAddress Gets the socket address of the current domain's current server
- (void *)getDomainHandle Gets the NetInfo™ handle of the current domain
- (ni_entrylist *)findDirectory:(const char *)parentDirectory
withProperty:(const char *)property Gets the values associated with a property

Checking the Error Status

- (ni_status)lastError Returns the status code from the most recent NetInfo call

Assigning a Delegate

- setDelegate:anObject Sets the delegate of the NIDomain object

NIDomainPanel

Inherits From: Object

Allocating and Initializing an NIDomainPanel Object

- + new Returns an NIDomainPanel object
- + allocWithoutPanelFromZone:(NXZone *)zone Allocates an NIDomainPanel object that has no panel
- init Initializes a new NIDomainPanel object

Displaying the Panel

- (int)**runModal** Brings up the panel
- **resizePanelBeforeShowing:(const char *)panelDefaultName** Resizes the panel
- **panel** Returns the **id** of the panel
- **windowDidResize:sender** Detects that the window has changed size

Getting Data

- (int)**exitFlags** Returns the exit flags from the panel
- (const char *)**domain** Returns the name of the selected domain
- (const char *)**panelSizeDefaultName** Returns a string indicating the panel's default size

Filling the Browser

- **loadDomainBrowser** Loads current domain information into the browser
- **loadDomainBrowserFrom:(const char *)whereFrom** Loads information from the domain into the browser
- (int)**browser:sender** Fills the indicated column with data
 fillMatrix:matrix
 inColumn:(int)column
- **browser:sender** Fills the indicated cell with data
 loadCell:cell
 atRow:(int)row
 inColumn:(int)column
- **freeLastColumn** Clears the data in the rightmost column of the browser
- **fillNextColumn** Fills the next column of the browser

Text-Related Methods

- **completeDomain** Completes the text in the text field
- **runOk:sender** Calls the OK button's action method
- **text:textObject** Detects empty text field
 isEmpty:(BOOL)flag
- **textWillChange:textObject** Detects that text is about to change
- (BOOL)**textWillEnd:textObject** Detects that the user has finished editing the text field

Target and Action Methods

- **cellWasHitInBrowser:(id)sender** Method invoked by the browser
- **cancel:sender** Method invoked by the Cancel button
- **ok:sender** Method invoked by the OK button

NILoginPanel

Inherits From: Panel : Window : Responder : Object

Creating a Panel

+ **new** Returns an NILoginPanel object

Running the Panel

- (BOOL)**runModal:***sender*
 inDomain:(void *)*domainID* Brings up the panel
- (BOOL)**runModal:***sender*
 inDomain:(void *)*domainID*
 withUser:(const char *)*userName*
 withInstruction:(const char *)*whatWarning*
 allowChange:(BOOL)*disableUser* Brings up the panel, as specified
- (BOOL)**runModalWithValidation:***sender*
 inDomain:(void *)*domainID*
 withUser:(const char *)*userName*
 withInstruction:(const char *)*whatWarning*
 allowChange:(BOOL)*enableUser* Brings up the panel and specifies validation by the delegate

Target and Action Methods

- **ok:***sender* Method invoked by the OK button
- **cancel:***sender* Method invoked by the Cancel button

Getting Data

- (BOOL)**isValidLogin:***sender* Returns whether the login was successful
- (const char *)**getPassword:***sender* Returns the text in the password field
- (const char *)**getUser:***sender* Returns the text in the account name field

NIOpenPanel

Inherits From: NIDomainPanel : Object

Initializing and Running a Panel

- + **new** Returns an NIOpenPanel object
- (int)**runModal** Brings up the panel

Getting Data from the Panel

- (const char *)**directory** Returns the directory that's selected in the lower browser
- (const char *)**panelSizeDefaultName** Returns a string indicating the panel's default size

Manipulating the Panel

- **setDirectoryPath**:(const char *)*path* Sets the initial directory path in the lower browser
- **setListTitle**:(const char *)*title* Sets the title of the lower half of the panel
- **setPanelTitle**:(const char *)*title* Sets the title of the panel
- **refreshLowerData**:*sender* Rereads and redraws the lower browser

Searching

- **searchItemList**:*textThing* Keeps lower browser and text field in sync
- **searchTextField** Keeps lower browser and text field in sync

Filling the Browser

- (int)**browser**:*sender* Fills the indicated column with data
 fillMatrix:*matrix*
 inColumn:(int)*column*
- **browser**:*sender* Fills the indicated cell with data
 loadCell:*cell*
 atRow:(int)*row*
 inColumn:(int)*column*

Text-Related Methods

- **text**:*textObj* Detects empty text field
 isEmpty:(BOOL)*flag*
- (BOOL)**textWillChange**: *textObject* Detects that text is about to change

- **completeItemName** Reserved for future use
- **completeDomain** Completes the text in the upper text field

Target and Action Methods

- **cellWasHitInBrowser:(id)sender** Method invoked by the upper browser
- **cellWasHitInItemList:sender** Method invoked by the lower browser

NISavePanel

Inherits From: NIOpenPanel : NIDomainPanel : Object

Creating a New NISavePanel Object

- + **new** Returns an NISavePanel object

Displaying the Panel

- (int)**runModal** Brings up the panel
- (int)**runModalWithString:(char *)initialValue** Brings up the panel, with the string in the lower text field
- (int)**runModalWithUneditableString:(char *)initialValue** Brings up the panel, with an uneditable string in the lower text field

Getting Data from the Panel

- (const char *)**panelSizeDefaultName** Returns a string indicating the panel's default size
- (const char *)**directory** Returns the directory that's selected in the lower browser

Target and Action Methods

- **cellWasHitInItemList:sender** Method invoked by the lower browser

Manipulating the Panel

- **setStartingDomainPath:(const char *)directory** Sets the initial path in the upper browser
- **refreshLowerData:sender** Rereads and redraws the lower browser

Functions

This section summarizes the single NetInfo Kit function.

Fill a column of a domain structure:

```
ni_status      NIFillDomainHierarchy(struct NIIierarchyOfDomains *domains, int level, const
                                         char *toMatch, int selectedLevel);
```


Types and Constants

Symbolic Constants

Connection Status

NL_ALREADYCONNECTED
NL_NOTCONNECTED

Test Mode	Value
NL_USERTESTMODE	0
NL_NETINFOTESTMODE	1

Structures

NIDomainCellData

```
struct NIDomainCellData {  
    char *name;  
    BOOL isaLeaf;  
}
```

NIHierarchyOfDomains

```
struct NIHierarchyOfDomains {  
    int numberOfLevels;  
    struct NIMultiDomainList *domainListAtLevel;  
}
```

NIMultiDomainList

```
struct NIMultiDomainList{  
    int numberOfDomains;  
    int activeDomain;  
    id activeDomainObject;  
    struct NIDomainCellData *topDomain;  
}
```

12 *Networks: Novell NetWare*

Please contact Novell, Inc. for documentation on the Novell® NetWare® programming interface.

13 *Phone Kit*

Classes

NXPhone

Inherits From: Object

Initializing an NXPhone Object

- **init** Initializes the NXPhone object to be type NX_ISDNDevice
- **initWith:(NXPhoneDeviceType)deviceType** Initializes the NXPhone object to *deviceType*

Running the Connection to the Phone Server

- **run** Runs the connection to the Phone Server
- **runFromAppKit** Makes the main event loop monitor the Phone Server
- **addPort:(port_t)aPort** Adds *aPort* as a source for remote input
 - receiver:anObject**
 - method:(SEL)aSelector**

Testing the Phone Line

- **(BOOL)isActive** Returns whether the phone line is working

Managing Channels

- (void)**addChannel:***aChannel* Registers *aChannel* as one that the application can use
- (void)**removeChannel:***aChannel* Disassociates *aChannel* from the NXPhone object
- (BOOL)**acquireChannel:***aChannel* Locks down *aChannel* for the application's exclusive use
- (void)**releaseChannel:***aChannel* Relinquishes exclusive rights over *aChannel*

NXPhoneCall

Inherits From: Object

Initializing an NXPhoneCall Object

- **init** Initializes the NXPhoneCall to be type NX_VoiceCall
- **initWithType:**(NXPhoneCallType)*callType* Initializes the NXPhoneCall object to *callType*
- (void)**setType:**(NXPhoneCallType)*callType* Sets the NXPhoneCall object to be *callType*
- (NXPhoneCallType)**type** Returns the call type, NX_VoiceCall or NX_DataCall

Initiating a Call

- (void)**pickUp** Takes the phone off-hook to initiate an outgoing call
- (void)**dialToneReceived** Implemented by subclasses to respond to a dial tone
- (void)**dialDigits:**(char *)*digits* Dials the phone number recorded in *digits*
- (void)**dialingComplete** Implemented by subclasses to react when number is dialed
- (void)**remoteBusy** Implemented by subclasses to respond to busy signal
- (void)**remoteRing** Implemented by subclasses to respond to remote ring
- (void)**remotePickup** Implemented by subclasses to respond to remote pickup

Getting a Call

- (void)**ring** Implemented by subclasses to respond to ringing phone
- (void)**pickUp** Takes the phone off-hook to answer an incoming call

Sending and Receiving Data

- (void)**callConnected**
- (void)**transmitData:(void *)data
length:(int)numBytes**
- (void)**dataReceived:(void *)data
length:(int)numBytes**
- (void)**useHDLC:(BOOL)flag**

Implemented by subclasses to respond when call is set up

Implemented by subclasses to transmit data during a call

Implemented by subclasses to accept data during a call

Determines whether transmitted data is HDLC encoded

Putting a Call on Hold

- (void)**hold**
- (void)**resume**

Not implemented for Release 3

Not implemented for Release 3

Detecting a Touch-Tone

- (void)**toneReceived:(int)key**

Implemented by subclasses to respond to a remote tone

Terminating a Call

- (void)**hangUp**
- (void)**remoteHangup**
- (void)**callReleased**

Puts the phone back on-hook and terminates the call

Implemented by subclasses to respond to a remote hangup

Implemented by subclasses to respond when call is ended

Finding the Current State of the Call

- (NXPhoneCallState)**state**

Returns the current status of the phone call

Responding to Errors

- (void)**error:(SEL)lastMessage
reason:(NXPhoneError)cause**

Implemented by subclasses to respond to error notifications

NXPhoneChannel

Inherits From: Object

Initializing an NXPhoneChannel Object

- **init** Initializes the NXPhoneChannel to NX_AnyISDNChannel
- **initWithType:(NXPhoneChannelType)channelType** Initializes the NXPhoneChannel object to *channelType*
- (void)**setType:(NXPhoneChannelType)channelType** Sets the type of the NXPhoneChannel to *channelType*
- (NXPhoneChannelType)**type** Returns the channel type

Tracking Calls

- (void)**addCall:aCall** Adds *aCall* to the channel
- (void)**removeCall:aCall** Removes *aCall* from the channel
- **activeCall** Returns the currently active NXPhoneCall for the channel

Setting Up an Incoming Call

- **allocateIncomingCallOfType:(NXPhoneCallType)callType** Implemented by subclasses to provide object to handle call
- (BOOL)**acceptCall:newCall** Implemented by subclasses to accept or refuse a call

Responding to Errors

- (void)**channelError:(NXPhoneError)cause** Implemented by subclasses to handle errors due to *cause*

Functions

Error Function

Get a string matching an error constant:

```
const char *NXPhoneErrorString(NXPhoneError errval)
```


Types and Constants

Defined Types

NXPhoneCallState

```
typedef enum {  
    NX_PhoneNullState = -1,  
    NX_PhoneIdle = 0,  
    NX_PhoneOriginating = 1,  
    NX_PhoneDialing = 2,  
    NX_PhoneConversation = 3,  
    NX_PhoneAlerting = 4,  
    NX_PhoneReleasing = 5  
} NXPhoneCallState;
```

NXPhoneCallType

```
typedef enum {  
    NX_DataCall = 4,  
    NX_VoiceCall = 5  
} NXPhoneCallType;
```

NXPhoneChannelType

```
typedef enum {  
    NX_B1Channel,  
    NX_B2Channel,  
    NX_DChannel,  
    NX_POTSChannel,  
    NX_AnyISDNChannel  
} NXPhoneChannelType;
```

Not implemented for Release 3

NXPhoneDeviceType

```
typedef enum {  
    NX_ISDNDevice,  
    NX_POTSDevice  
} NXPhoneDeviceType;
```

NXPhoneError

```
typedef enum {  
    NX_NotEndToEndISDN,  
    NX_BufferOverflow,  
    NX_TransmitFailure,  
    NX_NoHardwareAttached,  
    NX_HardwareFailure,  
    NX_TemporaryNetworkFailure,  
    NX_FacilityNotSubscribed  
} NXPhoneError;
```

14 Preferences

Classes

Application Additions

Preference implements these methods as additions to the Application class of the Application Kit.

Loading the Interface

- **loadNibForLayout:(const char *)name** Loads the nib file named “name.nib” and makes *anOwner*
 owner:anOwner its owner

Controlling Menu Items

- **enableEdit:(int)aMask** Enables and disables menu items in the Edit menu
- **enableWindow:** Enables and disables menu items in the Window menu

Accessing the Preferences Window

- **appWindow** Returns the **id** of the Preferences window

Layout

Inherits From: Object

Accessing the Root View

– view

Returns the View that's loaded into the Preferences window

Notification of State Change

– didHide:*sender*

Received when the application hides itself

– didUnhide:*sender*

Received when the application unhides itself

– willSelect:*sender*

Received before the module is displayed

– didSelect:*sender*

Received after the module is displayed

– willUnselect:*sender*

Received before the module is removed from display

– didUnselect:*sender*

Received after the module has been removed from display

Types and Constants

Symbolic Constants

Window Menu Constants

MINIATURIZE_ITEM
CLOSE_ITEM
WINDOW_ALL_ITEMS

Edit Menu Constants

CUT_ITEM
COPY_ITEM
PASTE_ITEM
SELECTALL_ITEM
EDIT_ALL_ITEMS

Functions

Class Functions

Create a new instance of a class:

id	<code>class_createInstance(Class <i>aClass</i>, unsigned int <i>indexedIvarBytes</i>)</code>
id	<code>class_createInstanceFromZone(Class <i>aClass</i>, unsigned int <i>indexedIvarBytes</i>, NXZone <i>*zone</i>)</code>

Get the class template for an instance variable:

Ivar	<code>class_getInstanceVariable(Class <i>aClass</i>, const char <i>*variableName</i>)</code>
------	--

Get, add, and remove methods:

Method	<code>class_getInstanceMethod(Class <i>aClass</i>, SEL <i>aSelector</i>)</code>
Method	<code>class_getClassMethod(Class <i>aClass</i>, SEL <i>aSelector</i>)</code>
void	<code>class_addMethods(Class <i>aClass</i>, struct objc_method_list <i>*methodList</i>)</code>
void	<code>class_removeMethods(Class <i>aClass</i>, struct objc_method_list <i>*methodList</i>)</code>

Pose as the superclass:

Class	<code>class_poseAs(Class <i>theImposter</i>, Class <i>theSuperclass</i>)</code>
-------	---

Set and get the class version:

void	<code>class_setVersion(Class <i>aClass</i>, int <i>versionNumber</i>)</code>
int	<code>class_getVersion(Class <i>aClass</i>)</code>

System Functions

Manage run-time structures:

id	<code>objc_getClass(const char *aClassName)</code>
id	<code>objc_lookUpClass(const char *aClassName)</code>
id	<code>objc_getMetaClass(const char *aClassName)</code>
NXHashTable *	<code>objc_getClasses(void)</code>
void	<code>objc_addClass(Class aClass)</code>
Module *	<code>objc_getModules(void)</code>

Dynamically load and unload classes:

long	<code>objc_loadModules(char *files[], NXStream *stream, void (*callback)(Class, Category), struct mach_header **header, char *debugFilename)</code>
long	<code>objc_unloadModules(NXStream *stream, void (*callback)(Class, Category))</code>

Send messages at run time:

id	<code>objc_msgSend(id theReceiver, SEL theSelector, ...)</code>
id	<code>objc_msgSendSuper(struct objc_super *superContext, SEL theSelector, ...)</code>
id	<code>objc_msgSendv(id theReceiver, SEL theSelector, unsigned int argSize, marg_list argFrame)</code>

Make the run-time system thread safe:

void	<code>objc_setMultithreaded(BOOL flag)</code>
------	---

Object Functions

Manage object memory:

```
id          object_dispose(Object *anObject)
id          object_copy(Object *anObject, unsigned int indexedIvarBytes)
id          object_copyFromZone(Object *anObject, unsigned int indexedIvarBytes,
                                NXZone *zone)
id          object_realloc(Object *anObject, unsigned int numBytes)
id          object_reallocFromZone(Object *anObject, unsigned int numBytes,
                                    NXZone *zone)
```

Return the class name:

```
const char * object_getClassName(id anObject)
```

Set and get instance variables:

```
Ivar        object_setInstanceVariable(id anObject, const char *variableName, void *value)
Ivar        object_getInstanceVariable(id anObject, const char *variableName, void **value)
```

Return a pointer to an object's extra memory:

```
void *      object_getIndexedIvars(id anObject)
```

Method Functions and Macros

Get information about a method:

```
unsigned int method_getNumberOfArguments(Method aMethod)
unsigned int method_getSizeOfArguments(Method aMethod)
unsigned int method_getArgumentInfo(Method aMethod, int index, const char **type,
                                     int *offset)
```

Examine and alter method argument values:

```
type-name    marg_getValue(marg_list argFrame, int offset, type-name)
type-name *  marg_getRef(marg_list argFrame, int offset, type-name)
void         marg_setValue(marg_list argFrame, int offset, type-name, type-name value)
```

Selector Functions

Match method names with method selectors:

SEL `sel_getUid(const char *aName)`
const char * `sel_getName(SEL aSelector)`

Determine whether a selector is valid:

BOOL `sel_isMapped(SEL aSelector)`

Register a method name:

SEL `sel_registerName(const char *aName)`

Types and Constants

Defined Types

Cache

```
typedef struct objc_cache *Cache;
```

Category

```
typedef struct objc_category *Category;
```

Ivar

```
typedef struct objc_ivar *Ivar;
```

marg_list

```
typedef void *marg_list;
```

Method

```
typedef struct objc_method *Method;
```

Module

```
typedef struct objc_module *Module;
```

Symbolic Constants

Type Constants	Meaning	Defined As
<code>_C_ID</code>	id	<code>'@'</code>
<code>_C_CLASS</code>	Class	<code>'#'</code>
<code>_C_SEL</code>	SEL	<code>'.'</code>
<code>_C_VOID</code>	void	<code>'v'</code>
<code>_C_CHR</code>	char	<code>'c'</code>
<code>_C_UCHR</code>	unsigned char	<code>'C'</code>
<code>_C_SHT</code>	short	<code>'s'</code>
<code>_C_USHT</code>	unsigned short	<code>'S'</code>
<code>_C_INT</code>	int	<code>'i'</code>
<code>_C_UINT</code>	unsigned int	<code>'I'</code>
<code>_C_LNG</code>	long	<code>'l'</code>
<code>_C_ULNG</code>	unsigned long	<code>'L'</code>
<code>_C_FLT</code>	float	<code>'f'</code>
<code>_C_DBL</code>	double	<code>'d'</code>
<code>_C_UNDEF</code>	an undefined type	<code>'?'</code>
<code>_C_PTR</code>	a pointer	<code>'^'</code>
<code>_C_CHARPTR</code>	char *	<code>'*'</code>
<code>_C_BFLD</code>	a bitfield	<code>'b'</code>
<code>_C_ARY_B</code>	begin an array	<code>'['</code>
<code>_C_ARY_E</code>	end an array	<code>']'</code>
<code>_C_UNION_B</code>	begin a union	<code>'('</code>
<code>_C_UNION_E</code>	end a union	<code>)'</code>
<code>_C_STRUCT_B</code>	begin a structure	<code>'{'</code>
<code>_C_STRUCT_E</code>	end a structure	<code>'}'</code>

Structures

```
objc_cache  
struct objc_cache {  
    unsigned int mask;  
    unsigned int occupied;  
    Method buckets[1];  
};
```

objc_category

```
struct objc_category {
    char *category_name;
    char *class_name;
    struct objc_method_list *instance_methods;
    struct objc_method_list *class_methods;
    struct objc_protocol_list *protocols;
};
```

objc_class

```
struct objc_class {
    struct objc_class *isa;
    struct objc_class *super_class;
    const char *name;
    long version;
    long info;
    long instance_size;
    struct objc_ivar_list *ivars;
    struct objc_method_list *methods;
    struct objc_cache *cache;
    struct objc_protocol_list *protocols;
};
```

objc_ivar

```
struct objc_ivar {
    char *ivar_name;
    char *ivar_type;
    int ivar_offset;
};
```

objc_ivar_list

```
struct objc_ivar_list {
    int ivar_count;
    struct objc_ivar ivar_list[1];
};
```

objc_method

```
struct objc_method {
    SEL method_name;
    char *method_types;
    IMP method_imp;
};
```

objc_method_description

```
struct objc_method_description {
    SEL name;
    char *types;
};
```

objc_method_description_list

```
struct objc_method_description_list {
    int count;
    struct objc_method_description list[1];
};
```

objc_method_list

```
struct objc_method_list {
    struct objc_method_list *method_next;
    int method_count;
    struct objc_method method_list[1];
};
```

objc_module

```
struct objc_module {
    unsigned long version;
    unsigned long size;
    const char *name;
    Symtab symtab;
};
```


objc_protocol_list

```
struct objc_protocol_list {
    struct objc_protocol_list *next
    int count;
    Protocol *list[1];
};
```

objc_super

```
struct objc_super {
    id receiver;
    Class class;
};
```

Global Variables

Function Pointers

```
id (*_alloc)      (Class aClass, unsigned int indexedIvarBytes)
id (*_dealloc)   (Object *anObject)
id (*_realloc)   (Object *anObject, unsigned int numBytes)
id (*_copy)      (Object *anObject, unsigned int indexedIvarBytes)
id (*_zoneAlloc) (Class aClass, unsigned int indexedIvarBytes, NXZone *zone)
id (*_zoneRealloc) (Object *anObject, unsigned int numBytes, NXZone *zone)
id (*_zoneCopy)  (Object *anObject, unsigned int indexedIvarBytes, NXZone *zone)
id (*_error)     (Object *anObject, char *format, va_list ap)
```

16 *Sound*

Classes

NXPlayStream

Inherits From: NXSoundStream : Object

Initializing an NXPlayStream

– **initWithDevice:***anObject* Initializes a newly allocated NXPlayStream

Activating and Playing

– (NXSoundDeviceError)**activate** Prepares the NXPlayStream for playback
– (NXSoundDeviceError)**playBuffer:**(void *)*data* Plays a buffer of sound
 size:(unsigned int)*bytes*
 tag:(int)*aTag*
 channelCount:(unsigned int)*channels*
 samplingRate:(float)*rate*

- (NXSoundDeviceError)**playBuffer**:(void *)*data* Plays a buffer of sound
 size:(unsigned int)*bytes*
 tag:(int)*aTag*
 channelCount:(unsigned int)*channels*
 samplingRate:(float)*rate*
 bufferGainLeft:(float)*leftAmp*
 right:(float)*rightAmp*
 lowWaterMark:(unsigned int)*lowWater*
 highWaterMark:(unsigned int)*highWater*

Setting Gain and Peak Detection

- (NXSoundDeviceError)**setGainLeft**:(float)*leftAmp* Sets the NXPlayStream’s stereo gain
 right:(float)*rightAmp*
- **getGainLeft**:(float *)*leftScale* Returns the NXPlayStream’s gain by reference
 right:(float *)*rightScale*
- (NXSoundDeviceError)**getPeakLeft**:(float *)*leftAmp* Returns the most recent peak amplitudes by reference
 right:(float *)*rightAmp*
- (NXSoundDeviceError)**setDetectPeaks**:(BOOL)*flag* Sets whether the NXPlayStream will detect peaks
- (BOOL)**isDetectingPeaks** Returns whether the NXPlayStream is detecting peaks
- (NXSoundDeviceError)**setPeakHistory**:(unsigned int)*bufferCount* Sets the number of buffers over which peaks are detected
- (unsigned int)**peakHistory** Returns the number of buffers over which peaks are detected

Delegate Methods

- **soundStreamDidUnderrun**:*sender* Invoked when the sound driver underruns

NXRecordStream

Inherits From: NXSoundStream : Object

Enqueueing Buffers

- (NXSoundDeviceError)**recordSize**:(unsigned int)*bytes*
 tag:(int)*anInt* Enqueues a recording buffer

- (NXSoundDeviceError)recordSize:(unsigned int)bytes
 tag:(int)aTag Enqueues a recording buffer
- lowWaterMark:(unsigned int)lowWater
- highWaterMark:(unsigned int)highWater

Requesting Data

- (NXSoundDeviceError)sendRecordedDataToDelegate
 Sends the current buffer to the delegate

Delegate Methods

- soundStreamDidOverrun:sender Invoked when the sound driver overruns
- soundStreamDidRecordData:(void *)data Delivers a buffer of recorded data
 size:(unsigned int)numBytes
- forBuffer:(int)tag

NXSoundDevice

Inherits From: Object

Initializing and Freeing an NXSoundDevice

- init Initializes a newly allocated NXSoundDevice
- initWithHost:(const char *)hostName Initializes a new NXSoundDevice on the given host
- free Frees the NXSoundDevice

Using a Separate Thread

- + (pthread_t)replyThread Returns the thread in which driver messages are sent
- + setUseSeparateThread:(BOOL)flag Sets whether a separate thread is used for driver messages
- + (BOOL)isUsingSeparateThread Returns whether a separate thread is used for driver messages
- + setThreadThreshold:(int)threshold Sets the message-reception threshold
- + (int)threadThreshold Returns the message-reception threshold

Examining Ports

- (port_t)**devicePort** Returns the port used to communicate with the sound driver
- + (port_t)**replyPort** Returns the port to which the sound driver sends messages
- (port_t)**streamOwnerPort** Returns the port used to connect to the sound driver

Identifying the Host Computer

- (const char *)**host** Returns the name of the computer on which the NXSoundDevice was initialized

Configuring the Object

- (NXSoundDeviceError)**setBufferCount:(unsigned int)count** Sets the number of DMA buffers for the sound device
- (unsigned int)**bufferCount** Returns the number of DMA buffers for the sound device
- (unsigned int)**bufferSize** Returns the size in bytes of each DMA buffer
- (NXSoundDeviceError)**setReserved:(BOOL)flag** Sets whether the underlying device is reserved
- (BOOL)**isReserved** Returns whether the underlying device is reserved
- (NXSoundDeviceError)**setBufferSize:(unsigned int)bytes** Sets the size in bytes of each DMA transfer buffer
- + **setTimeout:(unsigned int)milliseconds** Sets the timeout period
- + (unsigned int)**timeout** Returns the timeout period

Finding Peak Amplitudes

- (NXSoundDeviceError)**getPeakLeft:(float *)leftAmp right:(float *)rightAmp** Returns the most recent peak amplitudes by reference
- (NXSoundDeviceError)**setDetectPeaks:(BOOL)flag** Sets whether the NXSoundDevice will detect peaks
- (BOOL)**isDetectingPeaks** Returns whether the NXSoundDevice is detecting peaks
- (NXSoundDeviceError)**setPeakHistory:(unsigned int)bufferCount** Sets the number of buffers over which peaks are detected
- (unsigned int)**peakHistory** Returns the number of buffers over which peaks are detected

Controlling Streams

- **abortStreams:sender** Aborts all streams connected to this NXSoundDevice
- **pauseStreams:sender** Pauses all streams connected to this NXSoundDevice
- **resumeStreams:sender** Resumes all streams connected to this NXSoundDevice

Handling errors

- (NXSoundDeviceError)**lastError** Returns the most recent error
- + (const char *)**textForError:(NXSoundDeviceError)errorCode** Returns a string that describes the given error

NXSoundIn

Inherits From: NXSoundDevice : Object

Finding the Device Port

- + (port_t)**lookUpDevicePortOnHost:(const char *)hostName** Returns the sound-in port on *hostName*

NXSoundOut

Inherits From: NXSoundDevice : Object

Setting Attributes for Sound Output

- (NXSoundDeviceError)**setAttenuationLeft:(float)leftDB**
right:(float)rightDB Sets the attenuation level for playback
- (NXSoundDeviceError)**setDeemphasis:(BOOL)flag** Sets the state of the de-emphasis filter
- (NXSoundDeviceError)**setInsertsZeros:(BOOL)flag** Sets the way in which the driver converts low to high sampling rate
- (NXSoundDeviceError)**setRampsDown:(BOOL)flag** Sets whether the end of sounds are ramped
- (NXSoundDeviceError)**setRampsUp:(BOOL)flag** Sets whether the beginning of sounds are ramped
- (NXSoundDeviceError)**setSpeakerMute:(BOOL)flag** Mutes or unmutes the internal speaker

Querying Sound Output Settings

- (BOOL)**doesDeemphasize** Returns whether the de-emphasis filter is turned on
- (BOOL)**doesInsertZeros** Returns whether the sound driver inserts zeros when it converts low to high sampling rate.
- (BOOL)**doesRampDown** Returns whether the end of a sound stream is ramped down
- (BOOL)**doesRampUp** Returns whether the start of a sound stream is ramped down
- (NXSoundDeviceError)**getAttenuationLeft:(float *)leftDB right:(float *)rightDB** Returns the attenuation settings by reference
- (BOOL)**isSpeakerMute** Returns whether the internal speaker is muted
- (unsigned int)**clipCount** Returns the number of sample frames that were clipped

Finding the Device Port

- + (port_t)**lookupDevicePortOnHost:(const char *)hostName** Returns the sound-out port on hostName

NXSoundStream

Inherits From: Object

Initializing and Freeing an NXSoundStream

- **init** Initializes the NXSoundStream
- **initOnDevice:aDevice** Initializes the NXSoundStream and connects it to *aDevice*
- **free** Deactivates and frees the NXSoundStream.

Setting the Device

- (NXSoundDeviceError)**setDevice:aDevice** Connects the NXSoundStream to *aDevice*
- **device** Returns the NXSoundDevice that the NXSoundStream is connected to.

Activating and Deactivating

- (NXSoundDeviceError)**activate** Adds the NXSoundStream to the list of active streams
- (NXSoundDeviceError)**deactivate** Aborts the NXSoundStream's current activity

Controlling the stream

- **abort:***sender* Stops the NXSoundStream’s playback or recording
- (NXSoundDeviceError)**abortAtTime:**(NXSoundStreamTime *)*time*
Schedules the NXSoundStream to be aborted
- **pause:***sender* Pauses the NXSoundStream’s recording or playback
- (NXSoundDeviceError)**pauseAtTime:**(NXSoundStreamTime *)*time*
Schedules the NXSoundStream to be paused
- **resume:***sender* Resumes the NXSoundStream’s recording or playback
- (NXSoundDeviceError)**resumeAtTime:**(NXSoundStreamTime *)*time*
Schedules the NXSoundStream to be resumed

Querying the Object

- (unsigned int)**bytesProcessed** Returns the number of bytes of sound that the NXSoundStream has recorded or played
- (BOOL)**isActive** Returns whether the NXSoundStream is currently activate
- (BOOL)**isPaused** Returns whether the NXSoundStream is currently paused
- (port_t)**streamPort** Returns the port used to connect to the sound driver
- (NXSoundDeviceError)**lastError** Returns the most recent sound device error

Assigning a Delegate

- **setDelegate:***anObject* Sets the NXSoundStream’s delegate.
- **delegate** Returns the NXSoundStream’s delegate.

Delegate Methods

- **soundStream:***sender* **didCompleteBuffer:**(int)*tag* Invoked when the driver finishes playing or recording
- **soundStream:***sender* **didStartBuffer:**(int)*tag* Invoked when the driver starts playing or recording
- **soundStreamDidAbort:***sender*
deviceReserved:(BOOL)*flag* Invoked when the driver aborts the stream
- **soundStreamDidPause:***sender* Invoked when the NXSoundStream is paused
- **soundStreamDidResume:***sender* Invoked when the NXSoundStream is resumed

Sound

Inherits From: Object

Conforms To: soundkit/Sound.h

Creating and Freeing a Sound Object

- | | |
|---|--|
| + addName: (const char *) <i>name</i>
fromBundle: (NXBundle *) <i>aBundle</i> | Creates a Sound object from the sound resource named <i>name</i> in the NXBundle <i>aBundle</i> |
| + addName: (const char *) <i>name</i>
fromSection: (const char *) <i>sectionName</i> | Creates a Sound object from the <i>sectionName</i> section of the sound segment in the application executable file |
| + addName: (const char *) <i>name</i>
fromSoundfile: (const char *) <i>filename</i> | Creates a Sound object from <i>filename</i> |
| – initWithSection: (const char *) <i>sectionName</i> | Creates a Sound object from the <i>sectionName</i> section of the sound segment in the application executable file |
| – initWithPasteboard: (Pasteboard *) <i>thePboard</i> | Creates a Sound object from the named pasteboard |
| – initWithSoundfile: (const char *) <i>filename</i> | Creates a Sound object from <i>filename</i> |
| – free | Frees the Sound object |

Accessing the Sound Name Table

- | | |
|--|---|
| + addName: (const char *) <i>name</i> sound: <i>aSound</i> | Assigns the name <i>name</i> to the Sound <i>aSound</i> and adds it to the name table |
| + findSoundFor: (const char *) <i>aName</i> | Finds and returns the named Sound object |
| + removeSoundForName: (const char *) <i>name</i> | Removes the named Sound from the name table |

Accessing the Sound's Name

- | | |
|---|----------------------------------|
| – setName: (const char *) <i>aName</i> | Set's the Sound object's name |
| – (const char *) <i>name</i> | Return's the Sound object's name |

Reading and Writing Sound Data

- | | |
|--|---|
| – (int) readSoundfile: (const char *) <i>filename</i> | Replaces the Sound's data with that in <i>filename</i> |
| – readSoundFromStream: (NXStream *) <i>stream</i> | Replaces the Sound's data with that read from <i>stream</i> |
| – (int) writeSoundfile: (const char *) <i>filename</i> | Writes the Sound's data to <i>filename</i> |
| – writeSoundToStream: (NXStream *) <i>stream</i> | Writes the Sound's data to <i>stream</i> |
| – (int) writeToPasteboard: (Pasteboard *) <i>pboard</i> | Writes the Sound's data to the named pasteboard |

Modifying Sound Data

- (int)**convertToFormat**:(int)*newFormat*
samplingRate:(double)*newRate*
channelCount:(int)*newChannelCount*
Converts the Sound's data to the specified format, sampling rate, and channel count
- (int)**convertToFormat**:(int)*newFormat*
Convert's the Sound's data to the specified format
- (int)**setDataSize**:(int)*newDataSize*
dataFormat:(int)*newDataFormat*
samplingRate:(double)*newSamplingRate*
channelCount:(int)*newChannelCount*
infoSize:(int)*newInfoSize*
Set's the Sound's data as specified
- **setSoundStruct**:(SNDSoundStruct *)*aStruct*
soundStructSize:(int)*size*
Set's the Sound's sound structure

Querying the Sound Data

- (SNDSoundStruct *)**soundStruct**
Returns the Sound's sound structure
- (int)**soundStructSize**
Gives the size of the Sound's sound structure
- (unsigned char *)**data**
Returns a pointer to the Sound's sound data
- (int)**dataFormat**
Returns the Sound's data format
- (int)**dataSize**
Returns the size in bytes of the Sound's data
- (int)**channelCount**
Returns the number of channels of sound
- (double)**samplingRate**
Returns the sound data's sampling rate
- (int)**sampleCount**
Returns the number of sample frames in the sound data
- (double)**duration**
Returns the sound's length in seconds
- (char *)**info**
Returns a pointer to the Sound's info string
- (int)**infoSize**
Returns the length in bytes of the Sound's info string
- (BOOL)**isEmpty**
Returns whether the Sound contains any sound data
- (BOOL)**compatibleWith**:*aSound*
Returns whether the Sound's format is compatible with that of *aSound*

Recording and playing

- (int)**pause**
Pauses the Sound's activity
- **pause**:*sender*
Pauses the Sound's activity
- (BOOL)**isPlayable**
Returns whether the Sound can be played
- (int)**play**
Plays the Sound
- **play**:*sender*
Plays the Sound
- (int)**record**
Records into the Sound
- **record**:*sender*
Records into the Sound
- (int)**resume**
Resumes the Sound's activity

- **resume:***sender* Resumes the Sound’s activity
- (int)**stop** Stops the Sound’s activity
- **stop:***sender* Stops the Sound’s activity
- (int)**samplesProcessed** Returns the number of sample frames played or recorded
- (int)**status** Returns the Sound’s activity code
- **soundBeingProcessed** Returns **self**
- (SNDSoundStruct *)**soundStructBeingProcessed** Returns the sound structure that’s being played or recorded
- (int)**processingError** Returns the last error code that was generated

Editing Sound Data

- (BOOL)**isEditable** Returns whether the Sound’s data can be edited
- (int)**copySamples:***aSound*
 at:(int)*startSample*
 count:(int)*sampleCount* Copies a range of samples from *aSound* into the receiver
- (int)**copySound:***aSound* Replaces the Sound’s data with that in *aSound*
- (int)**deleteSamples** Removes the Sound’s data
- (int)**deleteSamplesAt:**(int)*startSample*
 count:(int)*sampleCount* Removes a range of samples from the Sound’s data
- (int)**insertSamples:***aSound* **at:**(int)*startSample* Inserts *aSound*’s data into the Sound’s data
- (BOOL)**needsCompacting** Returns whether the Sound’s data needs to be compacted
- (int)**compactSamples** Compacts the Sound’s data

Archiving the Object

- **finishUnarchiving** Invoked automatically after unarchiving
- **read:**(NXTypedStream *)*stream* Unarchives the Sound from *stream*
- **write:**(NXTypedStream *)*stream* Archives the Sound to *stream*

Accessing the Delegate

- **setDelegate:***anObject* Sets the Sound’s delegate object
- **delegate** Returns the Sound’s delegate
- **tellDelegate:**(SEL)*theMessage* Sends *theMessage* to the delegate

Accessing the Sound Hardware

- + **getVolume:**(float *)*left* :(float *)*right* Returns the left and right volume settings by reference
- + **setVolume:**(float)*left* :(float)*right* Sets the left and right volumes (0.0 to 1.0)

+ (BOOL)isMuted
+ setMute:(BOOL)aFlag

Returns whether the internal speaker is muted
Sets whether the internal speaker is muted

Delegate Methods

– didPlay:sender
– didRecord:sender
– hadError:sender

– willPlay:sender
– willRecord:sender

Sent to the delegate when the Sound stops playing
Sent to the delegate when the Sound stops recording
Sent to the delegate if an error occurs during recording or playback
Sent to the delegate when the Sound begins to play
Sent to the delegate when the Sound begins to record

SoundMeter

Inherits From: View : Responder : Object

Initializing a SoundMeter

– initWithFrame:(const NXRect *)frameRect
Initializes the SoundMeter

Graphic Attributes

– setBackgroundGray:(float)aValue
Sets the SoundMeter’s background color
– (float)backgroundGray
Returns the SoundMeter’s background color
– setForegroundGray:(float)aValue
Sets the SoundMeter’s running bar color
– (float)foregroundGray
Returns the color of the running bar
– setBezeled:(BOOL)aFlag
Sets whether a bezeled border is drawn
– (BOOL)isBezeled
Returns whether the SoundMeter has a border
– setPeakGray:(float)aValue
Sets the SoundMeter’s peak bubble color
– (float)peakGray
Returns the SoundMeter’s peak bubble gray

Metering Attributes

– setSound:aSound
Sets the SoundMeter’s Sound object
– sound
Returns the Sound object that the SoundMeter is metering

- **setFloatValue:**(float)*aValue*
- **setHoldTime:**(float)*seconds*
- (float)**holdTime**

Sets the current running value
 Sets the SoundMeter’s peak value hold time in seconds
 Returns the SoundMeter’s peak hold time

Retrieving Meter Values

- (float)**floatValue**
- (float)**maxValue**
- (float)**minValue**
- (float)**peakValue**

Returns the current running amplitude value
 Returns the maximum running value so far
 Returns the minimum running value so far
 Returns the most recently detected peak value

Operating the Object

- **run:***sender*
- (BOOL)**isRunning**
- **stop:***sender*

Starts the SoundMeter running
 Returns whether the SoundMeter is currently running
 Stops the SoundMeter’s metering activity

Drawing the Object

- **drawCurrentValue**
- **drawSelf:**(const NXRect *)*rects* :(int)*rectCount*

Draws the SoundMeter’s running bar and peak bubble
 Draws all the components of the SoundMeter

Archiving

- **read:**(NXTypedStream *)*aStream*
- **write:**(NXTypedStream *)*aStream*

Unarchives the SoundMeter by reading it from *aStream*
 Archives the SoundMeter by writing it to *aStream*

SoundView

Inherits From: View : Responder : Object

Initializing and Freeing a SoundView

- **initWithFrame:**(const NXRect *)*frameRect*
- **free**

Initializes the SoundView
 Frees the SoundView

Modifying the Object

- **scaleToFit** Fits the sound data within the current frame
- **setBackgroundGray:(float)aGray** Sets the SoundView’s background gray
- **setBezeled:(BOOL)aFlag** Sets whether the SoundView has a bezeled border
- **setContinuous:(BOOL)aFlag** Sets the state of continuous action messages.
- **setDelegate:anObject** Sets the SoundView’s delegate
- **setDisplayMode:(int)aMode** Sets the SoundView’s display mode
- **setEnabled:(BOOL)aFlag** Enables or disables the SoundView
- **setForegroundGray:(float)aGray** Sets the SoundView’s foreground gray
- **setOptimizedForSpeed:(BOOL)flag** Sets whether the SoundView’s display mechanism is optimized
- **setSound:aSound** Sets the SoundView’s Sound object
- **sizeToFit** Resizes the SoundView’s frame to maintain a constant reduction factor

Querying the Object

- (float)**backgroundGray** Returns the SoundView’s background gray
- **delegate** Returns the SoundView’s delegate
- (int)**displayMode** Returns the SoundView’s display mode
- (float)**foregroundGray** Returns the SoundView’s foreground gray
- **getSelection:(int *)firstSample size:(int *)sampleCount** Returns the selection by reference
- (BOOL)**isAutoScale** Returns whether the SoundView is in autoscaling mode
- (BOOL)**isBezeled** Returns whether the SoundView has a bezeled border
- (BOOL)**isContinuous** Returns whether the SoundView responds to mouse-dragged events
- (BOOL)**isEnabled** Returns whether the SoundView is enabled
- (BOOL)**isOptimizedForSpeed** Returns whether the SoundView is optimized for speedy display
- (float)**reductionFactor** Returns the SoundView’s reduction factor
- **sound** Returns the SoundView’s Sound object.

Selecting and Editing the Sound Data

- **copy:sender** Copies the current selection to the pasteboard
- **cut:sender** Deletes the current selection
- **delete:sender** Deletes the current selection
- **mouseDown:(NXEvent *)theEvent** Allows a selection to be defined

- **paste:sender** Replaces the current selection
- **selectAll:sender** Creates a selection over the SoundView’s entire sound
- **setSelection:(int)firstSample size:(int)sampleCount** Sets the selection
- **(BOOL)isEditable** Returns whether the SoundView’s data can be edited.
- **setEditable:(BOOL)aFlag** Sets whether the SoundView can be edited

Pasteboard and Services Support

- **pasteboard:thePasteboard provideData:(const char *)pboardType** Places the SoundView’s sound on the given pasteboard
- **readSelectionFromPasteboard:thePasteboard** Replaces the SoundView’s current selection
- **validRequestorForSendType:(NXAtom)sendType andReturnType:(NXAtom)returnType** You never invoke this method
- **writeSelectionToPasteboard:thePasteboard types:(NXAtom *)pboardTypes** Places a copy of the SoundView’s current selection on the given pasteboard

Modifying the Display Coordinates

- **setAutoscale:(BOOL)aFlag** Sets the SoundView’s automatic scaling mode
- **setReductionFactor:(float)reductionFactor** Recomputes the size of the SoundView’s frame

Drawing the Object

- **drawSelf:(const NXRect *)rects :(int)rectCount** Displays the SoundView’s sound data
- **drawSamplesFrom:(int)first to:(int)last** Redisplays the given range of samples
- **hideCursor** Hides the SoundView’s cursor
- **showCursor** Displays the SoundView’s cursor
- **sizeTo:(NXCoord)width :(NXCoord)height** Sets the width and height of the SoundView’s frame

Responding to Events

- **(BOOL)acceptsFirstResponder** Returns YES
- **becomeFirstResponder** Promotes the SoundView to first responder
- **resignFirstResponder** Resigns the position of first responder

Performing the Sound Data

- **(BOOL)isPlayable** Returns whether the SoundView’s sound can be played
- **play:sender** Play the current selection
- **record:sender** Replaces the SoundView’s current selection

- **pause:***sender*
- **resume:***sender*
- **stop:***sender*
- **soundBeingProcessed**

Pauses the current playback or recording
 Resumes the current playback or recording
 Stops the SoundView’s current recording or playback
 Returns the Sound object that’s currently being played or recorded into

Archiving the Object

- **read:**(void *)*stream*
- **write:**(void *)*stream*

Unarchives the SoundView by reading it from *stream*
 Archives the SoundView by writing it to *stream*

Accessing the Delegate

- **didPlay:***sender*
- **didRecord:***sender*
- **hadError:***sender*
- **tellDelegate:**(SEL)*theMessage*
- **willPlay:***sender*
- **willRecord:***sender*

Used to redirect delegate messages
 Used to redirect delegate messages
 Used to redirect delegate messages
 Sends *theMessage* to the SoundView’s delegate
 Used to redirect delegate messages
 Used to redirect delegate messages

Delegate Methods

- **didPlay:***sender*
- **didRecord:***sender*
- **hadError:***sender*
- **selectionChanged:***sender*
- **soundDidChange:***sender*
- **willFree:***sender*
- **willPlay:***sender*
- **willRecord:***sender*

Sent to the delegate just after the SoundView is played.
 Sent to the delegate just after the SoundView is recorded into.
 Sent to the delegate if an error is encountered .
 Sent to the delegate when the SoundView’s selection changes.
 Sent to the delegate when the SoundView’s sound is edited
 Sent to the delegate when the SoundView is freed.
 Sent to the delegate just before the SoundView’s sound is played.
 Sent to the delegate just before the SoundView’s sound is recorded into.

Sound Functions

Accessing Sound Devices and Hardware

Access sound devices

```
int          SNDAcquire(int soundResource, int priority, int preempt, int timeout,
                      SNDNegotiationFun negFun, void *arg, port_t *devicePort,
                      port_t *ownerPort)
int          SNDReset(int soundResource, port_t devicePort, port_t ownerPort)
int          SNDRelease(int soundResource, port_t *devicePort, port_t *ownerPort)
```

Reserve sound devices for recording or playback

```
int          SNDReserve(int soundResource, int priority)
int          SNDUnreserve(int soundResource)
```

Set the host computer for subsequent playback or recording

```
int          SNDSetHost(char *newHostname)
```

Sound playback utilities

```
int          SNDSetVolume(int left, int right)
int          SNDGetVolume(int *left, int *right)
int          SNDSetMute(int speakerOn)
int          SNDGetMute(int *speakerOn)
int          SNDSetFilter(int filterOn)
int          SNDGetFilter(int *filterOn)
```

Recording and Playing

Play a soundfile

```
int SNDPlaySoundfile(char *path, int priority)
```

Recording and playing a sound

```
int          SNDStartPlaying(SNDSoundStruct *sound, int tag, int priority, int preempt,
                             SNDNotificationFun beginFun, SNDNotificationFun endFun)
int          SNDVerifyPlayable(SNDSoundStruct *sound)
int          SNDStartRecording(SNDSoundStruct *sound, int tag, int priority, int preempt,
                              SNDNotificationFun beginFun, SNDNotificationFun endFun)
int          SNDStartRecordingFile(char *fileName, SNDSoundStruct *sound, int tag,
                                   int priority, int preempt, SNDNotificationFun beginFun,
                                   SNDNotificationFun endFun)
int          SNDStop(int tag)
int          SNDWait(int tag)
int          SNDSamplesProcessed(int tag)
int          SNDModifyPriority(int tag, int newPriority)
```

Reading and Writing Soundfiles

Read a sound from a file

```
int          SNDReadSoundfile(char *path, SNDSoundStruct **sound)
int          SNDRead(int fd, SNDSoundStruct **sound)
int          SNDReadHeader(int fd, SNDSoundStruct **sound)
int          SNDReadDSPfile(char *path, SNDSoundStruct **sound, char *info)
```

Write a sound to a file

```
int          SNDWriteSoundfile(char *path, SNDSoundStruct *sound)
int          SNDWrite(int fd, SNDSoundStruct *sound)
int          SNDWriteHeader(int fd, SNDSoundStruct *sound)
```

Accessing Sound Data

Create and free a sound structure

```
int          SNDAlloc(SNDSoundStruct **sound, int dataSize, int dataFormat,
                    int samplingRate, int channelCount, int infoSize)
int          SNDFree(SNDSoundStruct *sound)
```

Gain access to sampled sound data

int SNDGetDataPointer(SNDSoundStruct *sound, char **ptr, int *size, int *width)

Measure samples in a sound

int SNDSampleCount(SNDSoundStruct *sound)

int SNDBytesToSamples(int byteCount, int channelCount, int dataFormat)

int SNDSamplesToBytes(int sampleCount, int channelCount, int dataFormat)

Accessing the DSP

Boot the DSP

int SNDBootDSP(port_t *devicePort, port_t *ownerPort, SNDSoundStruct *dspCore)

Run the DSP

int SNDRunDSP(SNDSoundStruct *dspCore, char *toDSP, int toCount, int toWidth, int toBufferSize, char **fromDSP, int *fromCount, int fromWidth, int negotiationTimeout, int flushTimeout, int conversionTimeout)

Compressing Sound Data

Compress or decompress a sound

int SNDCompressSound(SNDSoundStruct *fromSound, SNDSoundStruct **toSound, BOOL bitFaithful, int compressionAmount)

Query for frequency bands used by Audio Transform Compression

int SNDGetNumberOfATCBands(int *numBands)

int SNDGetATCBandFrequencies(int numBands, float *centerFreqs)

int SNDGetATCBandwidths(int numBands, float *bandwidths)

Editing Sound Data

Copy all or part of a sound

```
int          SNDCopySound(SNDSoundStruct **toSound, SNDSoundStruct *fromSound)
int          SNDCopySamples(SNDSoundStruct **toSound, SNDSoundStruct *fromSound,
                           int startSample, int sampleCount)
```

Edit a sampled sound

```
int          SNDInsertSamples(SNDSoundStruct *toSound, SNDSoundStruct *fromSound,
                              int startSample)
int          SNDDeleteSamples(SNDSoundStruct *sound, int startSample, int sampleCount)
int          SNDCompactSamples(SNDSoundStruct **toSound,
                              SNDSoundStruct *fromSound)
```

Sound Errors

Describe a sound error

```
char         *SNDSoundError(int err)
```

Driver Functions

DSP Functions

Start the DSP:

```
kern_return_t    snddriver_dsp_boot(port_t commandPort, int *bootImage, int imageSize,  
                                   int priority)  
kern_return_t    snddriver_dsp_reset(port_t commandPort, int priority)
```

Transfer data to and from the DSP via DMA

```
kern_return_t    snddriver_dsp_dma_write(port_t commandPort, int elementCount,  
                                         int dataFormat, pointer_t data)  
kern_return_t    snddriver_dsp_dma_read(port_t commandPort, int elementCount,  
                                        int dataFormat, pointer_t data)
```

Enqueue a DSP command

```
kern_return_t    snddriver_dsp_host_cmd(port_t commandPort, u_int hostCommand,  
                                        u_int priority)
```

Set the sound driver's protocol vis-a-vis the DSP

```
kern_return_t s  snddriver_dsp_protocol(port_t devicePort, port_t ownerPort, int protocol)
```

Set the DSP host flags

```
kern_return_t    snddriver_dsp_set_flags(port_t commandPort, u_int flagMask, u_int flagValue,  
                                         u_int priority)
```

Transfer data to and from the DSP

```
kern_return_t    snddriver_dsp_write(port_t commandPort, void *buffer, int elementCount, int  
                                     elementSize, int priority)  
kern_return_t    snddriver_dsp_read(port_t commandPort, void *buffer, int elementCount, int  
                                    elementSize, int priority)  
kern_return_t    snddriver_dsp_read_messages(port_t commandPort, void *buffer,  
                                             int elementCount, int elementSize, int priority)  
kern_return_t    snddriver_dsp_read_data(port_t commandPort, void **buffer, int elementCount,  
                                         int elementSize, int priority)
```

Request a DSP host interface register condition

```
kern_return_t    snddriver_dspcmd_req_condition(port_t commandPort, u_int registerMask,
                                                u_int conditionFlags, int priority, port_t replyPort)
```

Request the contents of the DSP-reply buffers

```
kern_return_t    snddriver_dspcmd_req_msg(port_t commandPort, port_t replyPort)
kern_return_t    snddriver_dspcmd_req_err(port_t commandPort, port_t replyPort)
```

Get the DSP command port

```
kern_return_t    snddriver_get_dsp_cmd_port(port_t devicePort, port_t ownerPort,
                                             port_t *commandPort)
```

Driver Setup and Access

Acquire ownership of sound resources

```
kern_return_t    snddriver_set_dsp_owner_port(port_t devicePort, port_t ownerPort,
                                              port_t *negotiationPort)
kern_return_t    snddriver_set_sndin_owner_port(port_t devicePort, port_t ownerPort,
                                              port_t *negotiationPort)
kern_return_t    snddriver_set_sndout_owner_port(port_t devicePort, port_t ownerPort,
                                              port_t *negotiationPort)
```

Reallocate the sound driver device port

```
kern_return_t    snddriver_new_device_port(port_t devicePort, port_t superuserPort,
                                           port_t *newDevicePort)
```

Respond to asynchronous sound driver messages

```
kern_return_t    snddriver_reply_handler(msg_header_t *reply, snddriver_handlers_t *handlers)
```


Types and Constants

Defined Types

NXSoundDeviceError

```
typedef enum _NXSoundDeviceError {  
    NX_SoundDeviceErrorNone = 0,  
    NX_SoundDeviceErrorKernel = NX_SOUNDDEVICE_ERROR_MIN,  
    NX_SoundDeviceErrorTimeout,  
    NX_SoundDeviceErrorLookUp,  
    NX_SoundDeviceErrorHost,  
    NX_SoundDeviceErrorNoDevice,  
    NX_SoundDeviceErrorNotActive,  
    NX_SoundDeviceErrorTag,  
    NX_SoundDeviceErrorMax = NX_SOUNDDEVICE_ERROR_MAX  
} NXSoundDeviceError
```

NXSoundStatus

```
typedef enum {  
    NX_SoundStopped,  
    NX_SoundRecording,  
    NX_SoundPlaying,  
    NX_SoundInitialized,  
    NX_SoundRecordingPaused,  
    NX_SoundPlayingPaused,  
    NX_SoundRecordingPending,  
    NX_SoundPlayingPending,  
    NX_SoundFreed = -1,  
} NXSoundStatus;
```

NXSoundStreamTime

```
typedef struct timeval NXSoundStreamTime;
```

SNDCompressionSubheader

```
typedef struct {  
    int originalSize  
    int method;  
    int numDropped;  
    int encodeLength;  
} SNDCompressionSubheader;
```

SNDError

```
typedef enum {  
    SND_ERR_NONE,  
    SND_ERR_NOT_SOUND,  
    SND_ERR_BAD_FORMAT,  
    SND_ERR_BAD_RATE,  
    SND_ERR_BAD_CHANNEL,  
    SND_ERR_BAD_SIZE ,  
    SND_ERR_BAD_FILENAME,  
    SND_ERR_CANNOT_OPEN,  
    SND_ERR_CANNOT_WRITE,  
    SND_ERR_CANNOT_READ,  
    SND_ERR_CANNOT_ALLOC,  
    SND_ERR_CANNOT_FREE,  
    SND_ERR_CANNOT_COPY,  
    SND_ERR_CANNOT_RESERVE,  
    SND_ERR_NOT_RESERVED,  
    SND_ERR_CANNOT_RECORD,  
    SND_ERR_ALREADY_RECORDING,  
    SND_ERR_NOT_RECORDING,  
    SND_ERR_CANNOT_PLAY,  
    SND_ERR_ALREADY_PLAYING,  
    SND_ERR_NOT_IMPLEMENTED,  
    SND_ERR_NOT_PLAYING,  
    SND_ERR_CANNOT_FIND,  
    SND_ERR_CANNOT_EDIT,  
    SND_ERR_BAD_SPACE,  
    SND_ERR_KERNEL,  
    SND_ERR_BAD_CONFIGURATION,  
    SND_ERR_CANNOT_CONFIGURE,  
    SND_ERR_UNDERRUN,  
    SND_ERR_ABORTED,  
    SND_ERR_BAD_TAG,
```

```
    SND_ERR_CANNOT_ACCESS,  
    SND_ERR_TIMEOUT,  
    SND_ERR_BUSY,  
    SND_ERR_CANNOT_ABORT,  
    SND_ERR_INFO_TOO_BIG,  
    SND_ERR_UNKNOWN,  
} SndError;
```

SNDNotificationFun

```
typedef int (*SNDNotificationFun)  
    (SNDSoundStruct *s,  
     int tag,  
     int err);
```

SNDSoundStruct

```
typedef struct {  
    int magic;  
    int dataLocation;  
    int dataSize;  
    int dataFormat;  
    int samplingRate;  
    int channelCount;  
    char info[4];  
} SNDSoundStruct;
```

snddriver_handlers

```
typedef struct snddriver_handlers {  
    void *arg;  
    int timeout;  
    sndreply_tagged_t started;  
    sndreply_tagged_t completed;  
    sndreply_tagged_t aborted;  
    sndreply_tagged_t paused;  
    sndreply_tagged_t resumed;  
    sndreply_tagged_t overflow;  
    sndreply_recorded_data_t recorded_data;  
    sndreply_dsp_cond_true_t condition_true;  
    sndreply_dsp_msg_t dsp_message;  
    sndreply_dsp_msg_t dsp_error;  
} snddriver_handlers_t;
```

sndreply_dsp_cond_true_t

```
typedef void (*sndreply_dsp_cond_true_t)
    (void *arg,
     unsigned int mask,
     unsigned int flags,
     unsigned int regs);
```

sndreply_dsp_msg_t

```
typedef void (*sndreply_dsp_msg_t)
    (void *arg,
     int data,
     int size);
```

sndreply_recorded_data_t

```
typedef void (*sndreply_recorded_data_t)
    (void *arg,
     int tag,
     void *data,
     int size);
```

sndreply_tagged_t

```
typedef void (*sndreply_tagged_t)
    (void *arg,
     int tag);
```

Symbolic Constants

ATC Frame Size

ATC_FRAME_SIZE

Compression Formats

SND_CFORMAT_BITS_DROPPED
SND_CFORMAT_BIT_FAITHFUL
SND_CFORMAT_ATC

DSP Host Commands

SNDDRIVER_DSP_HC_HOST_RD
SNDDRIVER_DSP_HC_HOST_WD
SNDDRIVER_DSP_HC_SYS_CALL

DSP Protocol Options

SNDDRIVER_DSP_PROTO_DSPERR
SNDDRIVER_DSP_PROTO_C_DMA
SNDDRIVER_DSP_PROTO_S_DMA
SNDDRIVER_DSP_PROTO_HFABORT
SNDDRIVER_DSP_PROTO_DSPMSG
SNDDRIVER_DSP_PROTO_RAW

Executable File Segment Name

NX_SOUND_SEGMENT_NAME

Null Notification Function

SND_NULL_FUN

Sound Devices

SND_ACCESS_IN
SND_ACCESS_DSP
SND_ACCESS_OUT

Devices

Sound-in
DSP
Sound-out

Sampling Rates

SND_RATE_CODEC
SND_RATE_LOW
SND_RATE_HIGH

Rates

8012.8210513 Hz
22050.0 Hz
44100.0 Hz

Sound Device Timeout Limit

NX_SOUNDDEVICE_TIMEOUT_MAX

Sound Device Error Code Limits

NX_SOUNDDEVICE_ERROR_MIN

NX_SOUNDDEVICE_ERROR_MAX

Sound Stream Control Codes

SNDDRIVER_AWAIT_STREAM

SNDDRIVER_ABORT_STREAM

SNDDRIVER_PAUSE_STREAM

SNDDRIVER_RESUME_STREAM

Sound Stream Null Time

NXSOUNDSTREAM_TIME_NULL

Sound Stream Path Codes

SNDDRIVER_STREAM_FROM_SNDIN

SNDDRIVER_STREAM_TO_SNDOUT_22

SNDDRIVER_STREAM_TO_SNDOUT_44

SNDDRIVER_STREAM_FROM_DSP

SNDDRIVER_STREAM_TO_DSP

SNDDRIVER_STREAM_DSP_TO_SNDOUT_22

SNDDRIVER_STREAM_DSP_TO_SNDOUT_44

SNDDRIVER_STREAM_THROUGH_DSP_TO_SNDOUT_22

SNDDRIVER_STREAM_THROUGH_DSP_TO_SNDOUT_44

SNDDRIVER_DMA_STREAM_TO_DSP

SNDDRIVER_DMA_STREAM_FROM_DSP

SNDDRIVER_DMA_STREAM_THROUGH_DSP_TO_SNDOUT_22

SNDDRIVER_DMA_STREAM_THROUGH_DSP_TO_SNDOUT_44

Sound Structure Formats

SND_FORMAT_UNSPECIFIED
SND_FORMAT_MULAW_8
SND_FORMAT_LINEAR_8
SND_FORMAT_LINEAR_16
SND_FORMAT_LINEAR_24
SND_FORMAT_LINEAR_32
SND_FORMAT_FLOAT
SND_FORMAT_DOUBLE
SND_FORMAT_INDIRECT
SND_FORMAT_DSP_CORE
SND_FORMAT_DSP_DATA_8
SND_FORMAT_DSP_DATA_16
SND_FORMAT_DSP_DATA_24
SND_FORMAT_DSP_DATA_32
SND_FORMAT_DISPLAY
SND_FORMAT_MULAW_SQUELCH
SND_FORMAT_EMPHASIZED
SND_FORMAT_COMPRESSED
SND_FORMAT_COMPRESSED_EMPHASIZED
SND_FORMAT_DSP_COMMANDS

Sound Structure Magic Number

SND_MAGIC

SoundView Display Modes

NX_SOUNDVIEW_MINMAX
NX_SOUNDVIEW_WAVE

Global Variables

NXSoundPboardType

extern NXAtom NXSoundPboardType;

17 *3D Graphics Kit*

Classes

N3DCamera

Inherits From: View : Responder : Object

Initializing and Freeing

- | | |
|--|---------------------------------------|
| – init | Initializes with 0 frame size |
| – initWithFrame:(const NXRect *)fRect | Initializes with specified frame size |
| – free | Frees the N3DCamera |

All Drawing

- | | |
|--|--|
| – (BOOL) lockFocus | YES if PostScript and RenderMan drawing lock on camera |
| – unlockFocus | Unlocks PostScript and RenderMan drawing |
| – drawSelf:(NXRect *)rects :(int)nRects | Performs all RIB and PostScript drawing |

RenderMan Drawing

- **render**
- **renderSelf**:(RtToken)*context*
- **setFlushRIB**:(BOOL)*flag*
- **doesFlushRIB**
- **flushRIB**

Renders the camera and any content shapes
Override to perform custom rendering in the camera
Sets the receiver to invoke **flushRIB** within **render**
Tests whether the receiver invokes **flushRIB** within **render**
Waits until all RIB code has been rendered

PostScript Drawing

- **drawPS**:(NXRect *)*rects* :(int)*nRects*

Override to perform custom PostScript drawing

Background Color

- **setBackground****Color**:(NXColor)*color*
- (NXColor)**background****Color**
- **setDrawBackground****Color**:(BOOL)*flag*
- (BOOL)**doesDrawBackground****Color**

Sets the NXColor filled behind all drawing
The NXColor filled behind all drawing
If *flag* is YES, fills color behind all drawing
YES if camera fills color behind all drawing

PostScript Transformation Management

- **setFrame**:(const NXRect *)*fRect*
- **moveTo**:(NXCoord)*x* :(NXCoord)*y*
- **moveBy**:(NXCoord)*deltaX* :(NXCoord)*deltaY*
- **sizeTo**:(NXCoord)*width* :(NXCoord)*height*
- **sizeBy**:(NXCoord)*deltaWidth*
:(NXCoord)*deltaHeight*
- **rotateTo**:(NXCoord)*angle*
- **rotateBy**:(NXCoord)*deltaAngle*

Sets frame for both PostScript and RenderMan coordinate systems
Moves both PostScript and RenderMan coordinate systems
Moves both PostScript and RenderMan coordinate systems
Resizes both PostScript and RenderMan coordinate systems
Resizes both PostScript and RenderMan coordinate systems
Prevents rotation of PostScript coordinate system
Prevents rotation of PostScript coordinate system

Shape Hierarchy Management

- **setWorldShape**:*a3DShape*
- **worldShape**

Sets world shape
Returns world shape

Global Light Management

- **addLight**:*aLight*
- **removeLight**:*aLight*
- **lightList**

Adds an N3DLight to the camera's global light list
Removes an N3DLight to the camera's global light list
Returns the camera's global light list

Picking

– **selectShapesIn**:(const NXRect *)*selectionRect* Returns a List of N3DShapes in *selectionRect*

Projection Rectangle

– **setProjectionRectangle**:(float)*left* Sets the 3D coordinate system projection rectangle
:(float)*right*
:(float)*top*
:(float)*bottom*

– **getProjectionRectangle**:(float *)*left* Returns the 3D coordinate system’s projection
:(float *)*right* rectangle
:(float *)*top*
:(float *)*bottom*

Selecting Projection Type

– **setProjection**:(N3DProjectionType)*aProjection* Sets the projection type
– (N3DProjectionType)**projectionType** Returns the projection type

Pretransform Matrix

– **setPreTransformMatrix**:(RtMatrix)*theMatrix* Sets the camera’s pretransformation matrix
– **getPreTransformMatrix**:(RtMatrix)*theMatrix* Returns the camera’s pretransformation matrix
– **setUsePreTransformMatrix**:(BOOL)*flag* Sets the camera to use its pretransformation matrix
– (BOOL)**usesPreTransformMatrix** YES if camera uses its pretransformation matrix

Setting Viewpoint

– **setEyeAt**:(RtPoint)*fromPoint* Sets the eye-to-viewpoint vector and roll
:(RtPoint)*toward*
:(float)*aRollAngle*

– **getEyeAt**:(RtPoint *)*fromPoint* Gets the eye-to-viewpoint vector and roll
:(RtPoint *)*toward*
:(float *)*aRollAngle*

– **moveEyeBy**:(float)*sDistance* Moves camera in its own coordinate system
:(float)*tDistance*
:(float)*uDistance*

– **rotateEyeBy**:(float)*dElev*:(float)*dAzim* Rotates the camera in its own coordinates
:(RtPoint)*pivotPtr*

Clipping Planes

- **setClipPlanesNear:**(float)*aNearPlane*
far:(float)*aFarPlane* Sets the camera's near and far clipping planes
- **getClipPlanesNear:**(float *)*aNearPlane*
far:(float *)*aFarPlane* Returns the camera's near and far clipping planes

Field of View

- **setFieldOfViewByAngle:**(float)*viewAngle* Sets the viewing angle of the camera
- **setFieldOfViewByFocalLength:**(float)*aLength*
(float)*fieldOfView* Converts a focal length into a viewing angle for the camera
- (float)*fieldOfView* Returns the viewing angle of the camera

Pixel Aspect Ratio

- **setPixelAspectRatio:**(float)*theRatio* Sets the pixel aspect ratio for the camera
- (float)*pixelAspectRatio* Returns the pixel aspect ratio for the camera

Converting Coordinates

- **convertPoints:**(RtPoint *)*points*
count:(int)*n*
fromSpace:*aShape* Converts an array of points to PostScript coordinates
- **convertPoints:**(NXPoint *)*mcoords*
count:(int)*pointCount*
toWorld:(RtPoint *)*wcoords* Converts PostScript points to world coordinates

Crop Windows

- (int)**numCropWindows** Count of rectangle divisions for photoreal rendering
- **cropInRects:**(NXRect *)*theRects*
nRects:(int)*rectCount* Returns rectangles representing horizontal strips of camera image

Frame Number

- (int)**frameNumber** Returns 1

Printing

- (BOOL)**canPrintRIB** Returns YES

Copying RIB

– **copyRIBCode:**(NXStream *)*stream* Copies RIB code generated by the receiver

Setting World Attributes

– **worldBegin:**(RtToken)*theContext* Calls **RiWorldBegin()**

– **worldEnd:**(RtToken)*theContext* Calls **RiWorldEnd()**

Setting and Getting the Delegate

– **setDelegate:***cameraDelegate* Sets the receiver's delegate

– **delegate** Returns the receiver's delegate

Setting the Hider

– (N3DHider)**hider** Returns the receiver's N3DHider

– **setHider:**(N3DHider)*cameraHider* Sets the receiver's N3DHider

– **setSurfaceTypeForAll:**(N3DSurfaceType)*surface*
 chooseHider:(BOOL)*flag* Sets surface type for shapes in world shape hierarchy

Rendering Photorealistically

– (int)**renderAsEPS** Begins rendering, returns identifier for rendering session

– (int)**renderAsTIFF** Begins rendering, returns identifier for rendering session

Archiving

– **read:**(NXTypedStream *)*theStream* Reads the camera from the stream

– **write:**(NXTypedStream *)*theStream* Writes the camera to the stream

– **awake** Performs additional initialization after unarchiving

Methods Implemented by the Delegate

– **camera:***theCamera* Handles images generated by photoreal rendering
 didRenderStream:(NXStream *)*imageStream* methods
 tag:(int)*theJob*
 frameNumber:(int)*currentFrame*

N3DContextManager

Inherits From: Object

Initializing and Freeing

- + new Returns (creating if necessary) one instance per application
- free Destroys all contexts, frees the receiver

Getting the Main Context

- (RtToken)mainContext Returns (creating if necessary) the application's main context

Creating Other Contexts

- (RtToken)createContext:(const char *)name Creates a named context
- (RtToken)createContext:(const char *)name
withRenderer:(RtToken)renderer Creates a named context for a specific renderer
- (RtToken)createContext:(const char *)name
toFile:(const char *)ribFile Creates a named context on a file
- (RtToken)createContext:(const char *)name
toStream:(NXStream *)stream Does nothing, returns NULL

Managing the Current Context

- (RtToken)currentContext The application's current context
- (RtToken)setCurrentContext:(RtToken)context Sets the current context, returns previous context
- (RtToken)setCurrentContextByName:(const char *)name Sets the current context by name, returns previous context

Destroying a Context

- (void)destroyContext:(RtToken)context Destroys the context
- (void)destroyContextByName:(const char *)name Destroys a context with *name*

Archiving

- awake Performs additional initialization after unarchiving

N3DLight

Inherits From: N3DShape : Object

Initializing

– **init** Initializes the receiver as an N3D_AmbientLight

Setting Light Type

– **setType:(N3DLightType)aType** Sets the receiver's light type
– **(N3DLightType)type** Returns the receiver's light type
– **makeAmbientWithIntensity:(RtFloat)intensity** Sets type N3D_AmbientLight with appropriate parameter
– **makePointFrom:(RtPoint)from intensity:(RtFloat)intensity** Sets type N3D_PointLight with appropriate parameters
– **makeDistantFrom:(RtPoint)fromPoint to:(RtPoint)toPoint intensity:(RtFloat)i** Sets type N3D_PointLight with appropriate parameters
– **makeSpotFrom:(RtPoint)fromPoint to:(RtPoint)toPoint coneAngle:(RtFloat)coneAngle coneDelta:(RtFloat)deltaAngle beamDistribution:(RtFloat)distribution intensity:(RtFloat)intensity** Sets type N3D_SpotLight with appropriate parameters

Setting Light Parameters

– **setFrom:(RtPoint)fromPoint** Sets the from point
– **setFrom:(RtPoint)fromPoint to:(RtPoint)toPoint** Sets the from and to points
– **getFrom:(RtPoint *)fromPoint to:(RtPoint *)toPoint** Returns the from and to points
– **setConeAngle:(RtFloat)coneAngle coneDelta:(RtFloat)coneDelta beamDistribution:(RtFloat)distribution** Sets the cone angle, cone delta, and beam distribution
– **getConeAngle:(RtFloat *)coneAngle coneDelta:(RtFloat *)coneDelta beamDistribution:(RtFloat *)distribution** Gets the cone angle, cone delta, and beam distribution
– **setIntensity:(RtFloat)intensity** Sets the intensity
– **(RtFloat)intensity** Returns the intensity

Rendering

- **renderSelf:**(N3DCamera *)*theCamera* Renders the light as a local light
- **renderGlobal:**(N3DCamera *)*theCamera* Renders the light as a global light

Global Light Management

- **setGlobal:**(BOOL)*flag* Override to add behavior on becoming/resigning global
- (BOOL)**isGlobal** YES if light is global

Switching On and Off

- **switchLight:**(BOOL)*flag* If flag is YES, turns receiver on

Setting Color

- **setColor:**(NXColor)*theColor* Sets the receiver's color
- (NXColor)**color** Returns the receiver's color

Archiving

- **read:**(NXTypedStream *)*theStream* Reads the receiver from the stream
- **write:**(NXTypedStream *)*theStream* Writes the receiver to the stream
- **awake** Performs additional initialization after unarchiving

N3DMovieCamera

Inherits From: N3DCamera : View : Responder : Object

Initializing

- **initWithFrame:**(const NXRect *)*fRect* Initializes the receiver

RenderMan Drawing

- **render** Renders current frame; if printing, renders current page

Interactive Display

- `displayMovie` Displays movie, first frame to last, onscreen

Frame Numbers

- `setFrameNumber:(int)aFrameNumber` Sets the current frame number
- `(int)frameNumber` Returns the current frame number
- `setStartFrame:(int)start` Sets counters for movie
- `endFrame:(int)end`
- `incrementFramesBy:(int)skip`
- `(int)startFrame` Returns the first frame number
- `(int)endFrame` Returns the last frame number
- `(int)frameIncrement` Returns the frame increment

Archiving

- `read:(NXTypedStream *)theStream` Reads the receiver from the stream
- `write:(NXTypedStream *)theStream` Writes the receiver to the stream
- `awake` Performs additional initialization after unarchiving

N3DRenderPanel

Inherits From: Panel : Window : Responder : Object

Initializing the Class

- + `initialize` Initializes the class with data from the defaults database
- + `new` Creates, if necessary, and returns an N3DRenderPanel

Setting Accessory View

- `accessoryView` Returns the accessory view
- `setAccessoryView:aView` Sets the accessory view

Running Modal

- `(int)runModal` Presents the render panel in a modal loop

Resolution

- (int)**resolution** Returns the resolution set by the user in the panel

Host Management

- (int)**numSelectedHosts** Returns the number of hosts selected by the user
- (char **)**hostNames** Returns an array of strings for selected host names

Browser Delegate Method

- (int)**browser:sender**
 fillMatrix:theMatrix
 inColumn:(int)col Fills the panel's browser with host names

N3DRIBImageRep

Inherits From: NXImageRep : Object

Initializing and Freeing

- **initWithFile:(const char *)ribfile** Initializes the receiver from a file
- **initWithStream:(NXStream *)ribStream** Initializes the receiver from a stream
- **free** Frees the receiver

Declaring Data Types

- + (const char *const *)**imageUnfilteredFileTypes** Returns supported file types
- + (const NXAtom *)**imageUnfilteredPasteboardTypes** Returns supported pasteboard types
- + (BOOL)**canLoadFromStream:(NXStream *)theStream** Tests *theStream* for RIB data

Drawing

- (BOOL)**drawAt:(const NXPoint *)point** Returns YES if the image is successfully drawn at *point*
- (BOOL)**drawIn:(const NXRect *)rect** Returns YES if the image is successfully drawn in *rect*
- (BOOL)**draw** Returns YES if the image is successfully drawn

Size

- **getBoundingBox:**(NXRect *)*rectangle* Returns the rectangle that bounds the image
- **getSize:**(NXSize *)*theSize* Returns the size of the image

Background Color

- (NXColor)**backgroundColor** Returns the background color
- **setBackgroundColor:**(NXColor)*theColor* Sets the background color

Hidden Surface Removal Type

- (N3DHider)**hider** Returns the hider type for rendering images
- **setHider:**(N3DHider)*theHider* Sets the hider type for rendering images

Surface Type

- **setSurfaceType:**(N3DSurfaceType)*type* Sets the surface type used for rendering images
- (N3DSurfaceType)**surfaceType** Returns the surface type used for rendering images

Archiving

- **read:**(NXTypedStream *)*theStream* Reads the receiver from the stream
- **write:**(NXTypedStream *)*theStream* Writes the receiver to the stream

N3DRotator

Inherits From: Object

Initializing

- **init** Initializes the receiver
- **initWithCamera:***myCamera* Initializes the receiver and sets its camera

Setting Parameters

- **setCamera:***myCamera* Sets the receiver's camera
- **setCenter:**(const NXPoint *)*center*
andRadius:(float)*radius* Sets the receiver's center point and radius

Axes of Rotation

- **setRotationAxis:**(N3DAxis)*axis* Sets the axes about which the receiver rotates
- (N3DAxis)**rotationAxis** Returns the axis about which the receiver rotates

Mouse Tracking

- **trackMouseFrom:**(const NXPoint *)*lastPoint*
to:(const NXPoint *)*thisPoint*
rotationMatrix:(RtMatrix)*theRotation*
andInverse:(RtMatrix)*theInverse* Accepts two points in camera coordinates, returns rotations based on their offset

Archiving

- **read:**(NXTypedStream *)*theStream* Reads the receiver from the stream
- **write:**(NXTypedStream *)*theStream* Writes the receiver to the stream

N3DShader

Inherits From: Object

Initializing and Freeing

- **init** Initializes the receiver with no shader file
- **initWithShader:**(const char *)*aShader* Initializes the receiver with a shader file
- **free** Frees the receiver

Shader Language Object File

- **setShader:**(const char *)*aShader* Sets the receiver's shader
- (const char *)**shader** Returns the receiver's shader

Shader Color

- **setColor:**(NXColor)*aColor* Sets the receiver's color
- (NXColor)**color** Returns the receiver's color
- **setUseColor:**(BOOL)*flag* Sets the receiver to apply its color
- (BOOL)**doesUseColor** YES if receiver applies color

Shader Transparency

- **setTransparency:**(float)*alphaValue* Sets the receiver’s transparency
- (float)**transparency** Returns the receiver’s transparency

Shader Function Argument Handling

- (int)**shaderArgCount** The number of arguments for the shader function
- (const char *)**shaderArgNameAt:**(int)*argIndex* The name of the indicated argument
- (SLO_TYPE)**shaderArgType:**(const char *)*argName* The type of the named argument
- (BOOL)**isShaderArg:**(const char *)*argName* YES if *argName* is an argument to the shader function
- **setShaderArg:**(const char *)*floatName*
floatValue:(float)*floatValue* Sets the specified argument to the specified value
- **setShaderArg:**(const char *)*stringName*
stringValue:(const char *)*stringValue* Sets the specified argument to the specified value
- **setShaderArg:**(const char *)*pointName*
pointValue:(RtPoint)*pointValue* Sets the specified argument to the specified value
- **setShaderArg:**(const char *)*colorName*
colorValue:(NXColor)*colorValue* Sets the specified argument to the specified value
- **getShaderArg:**(const char *)*floatName*
floatValue:(float *)*floatValue* Gets the specified value for the specified argument
- **getShaderArg:**(const char *)*stringName*
stringValue:(const char **)*stringValue* Gets the specified value for the specified argument
- **getShaderArg:**(const char *)*pointName*
pointValue:(RtPoint *)*pointValue* Gets the specified value for the specified argument
- **getShaderArg:**(const char *)*colorName*
colorValue:(NXColor *)*colorValue* Gets the specified value for the specified argument
- **resetShaderArg:**(const char *)*argName* Resets the specified argument to its default value

Shader Type

- (SLO_TYPE)**shaderType** Returns the type of the shader

Setting

- **set** Applies the shader function during rendering

Archiving

- **read:**(NXTypedStream *)*theStream* Reads the receiver from the stream
- **write:**(NXTypedStream *)*theStream* Writes the receiver to the stream

N3DShape

Inherits From: Object

Initializing and Freeing

- **init** Initializes and returns the receiver
- **free** Frees the receiver and its descendants
- **freeAll** Frees the receiver, its next peer and its descendants

Rendering the N3DShape

- **render:(N3DCamera *)theCamera** Renders the shape and its descendants
- **renderSelf:(N3DCamera *)theCamera** Override to implement actual rendering
- **renderSelfAsBox:(N3DCamera *)theCamera** Renders only the shape's bounding box

Traversing the Shape Hierarchy

- **nextPeer** Returns the shape "to the right"
- **previousPeer** Returns the shape "to the left"
- **firstPeer** Returns the shape "to the far left" of receiver's peer group
- **lastPeer** Returns the shape "to the far right" of receiver's peer group
- **descendant** Returns the shape "below" the receiver
- **lastDescendant** Returns the farthest descendant below the receiver
- **ancestor** Returns the shape "above the receiver"
- **firstAncestor** Returns the shape at the top of the receiver's hierarchy
- **(BOOL)isWorld** YES if the receiver is at the top of its hierarchy

Managing the Shape Hierarchy

- **linkPeer:aPeer** Inserts *aPeer* between receiver and its next peer
- **linkDescendant:aDescendant** Inserts *aDescendant* between receiver and its descendant
- **linkAncestor:anAncestor** Sets receiver's ancestor, returns previous ancestor
- **unlink** Unlinks the receiver, reconnects peers and descendants
- **group:anAncestor** Makes the receiver a descendant of *anAncestor*
- **ungroup** Removes receiver from hierarchy, promotes descendant

Shader

- **setShader:***aShader* Sets an N3DShader for the shape
- **shaderType:**(SLO_TYPE)*type* Returns the N3DShader of *type*

Surface

- (N3DSurfaceType)**surfaceType** Returns the receiver's surface type
- **setSurfaceType:**(N3DSurfaceType)*theSurface*
andDescendants:(BOOL)*flag* Sets receiver's surface type; if flag is YES, sets descendants surface types

Bounding Box

- **getBoundingBox:**(RtBound *)*boundingBox* Returns by reference the receiver's bounding box
- **setDrawAsBox:**(BOOL)*flag* Sets the receiver to draw only its bounding box
- (BOOL)**doesDrawAsBox** YES if the receiver draws only its bounding box
- **getBounds:**(NXRect *)*boundingRect*
inCamera:*theCamera* Returns the rectangle representing the receiver's bounding box in the coordinates of the camera

Converting Points

- **convertObjectPoints:**(RtPoint *)*points*
count:(int)*n*
toCamera:*camera* Converts points from the receiver's coordinate system to that of *camera*
- **convertPoints:**(RtPoint *)*points*
count:(int)*n*
fromAncestor:(N3DShape *)*theShape* Converts points from an ancestor's coordinate system to that of the receiver
- **convertPoints:**(RtPoint *)*points*
count:(int)*n*
toAncestor:(N3DShape *)*theShape* Converts points from the receiver's coordinate system to that of an ancestor

Selectability

- **setSelectable:**(BOOL)*flag* Sets whether the receiver can be selected
- (BOOL)**isSelectable** YES if the receiver can be selected

Visibility

- **setVisible:**(BOOL)*flag* Sets the receiver to render when **renderSelf:** is invoked
- (BOOL)**isVisible** YES if receiver renders when **renderSelf:** is invoked

Naming Shapes

- **setShapeName:**(const char *)*aName* Sets the name of the shape
- (const char *)**shapeName** Returns the name of the shape

Delegate for Rendering

- **setRenderDelegate:***aShape* Sets the rendering delegate for the receiver
- **removeRenderDelegate** Removes and returns the render delegate
- **renderDelegate** Returns the render delegate

Transformation Matrices

- **setTransformMatrix:**(RtMatrix)*newTransform* Sets the receiver's transformation matrix
- **getTransformMatrix:**(RtMatrix)*transform* Returns the receiver's transformation matrix
- **concatTransformMatrix:**(RtMatrix)*theMatrix*
premultiply:(BOOL)*flag* Multiplies the transform matrix with *theMatrix*,
premultiplying by *theMatrix* if flag is YES
- **getCompositeTransformMatrix:**(RtMatrix)*theMatrix*
relativeToAncestor:(N3DShape *)*theAncestor* Returns the matrix representing the transformation
between *theAncestor*'s space and the receiver's
- **getInverseCompositeTransformMatrix:**(RtMatrix)*theMatrix*
relativeToAncestor:(N3DShape *)*theAncestor* Returns the matrix representing the transformation
between the receiver's space and *theAncestor*'s

Rotation, Scaling, Translation

- **rotateAngle:**(float)*ang*
axis:(RtPoint)*referencePoint* Rotates the receiver about its origin and the point
- **preRotateAngle:**(float)*angle*
axis:(RtPoint)*referencePoint* Premultiplies the rotation of the receiver about its
origin and the point
- **scale:**(float)*xScaleFactor*
:(float)*yScaleFactor*
:(float)*zScaleFactor* Scales the receiver
- **prescale:**(float)*xScaleFactor*
:(float)*yScaleFactor*
:(float)*zScaleFactor* Scales the receiver, premultiplying the transformation
- **scaleUniformly:**(float)*scaleFactor* Scales the receiver uniformly in all axes
- **prescaleUniformly:**(float)*scaleFactor* Scales the receiver uniformly in all axes, premultiplying the
transformation

– **translate:**(float)*xTranslation*
:(float)*yTranslation*
:(float)*zTranslation*

Translates the receiver

– **pretranslate:**(float)*xTranslation*
:(float)*yTranslation*
:(float)*zTranslation*

Translates the receiver, premultiplying the transformation

Archiving

– **read:**(NXTypedStream *)*theStream*
– **write:**(NXTypedStream *)*theStream*
– **awake**

Reads the camera from the stream

Writes the camera to the stream

Performs additional initialization after unarchiving

Functions

Data Component Functions

Return components of 3D data structures:

RtFloat	N3D_XComp(RtFloat <i>*theVector</i>)
RtFloat	N3D_YComp(RtFloat <i>*theVector</i>)
RtFloat	N3D_ZComp(RtFloat <i>*theVector</i>)
RtFloat	N3D_WComp(RtFloat <i>*theVector</i>)

Data Conversion Functions

Convert between RtPoints and RtBounds:

void	N3D_ConvertBoundToPoints(RtBound <i>theBound</i> , RtPoint <i>* thePoints</i>)
void	N3D_ConvertPointsToBound(RtPoint <i>* thePoints</i> , RtBound <i>theBound</i>)

Data Copying Functions

Efficiently copy 3D data types:

void	N3D_CopyBound(RtBound <i>sourceBounds</i> , RtBound <i>destBounds</i>)
void	N3D_CopyMatrix(RtMatrix <i>sourceMatrix</i> , RtMatrix <i>destMatrix</i>)
void	N3D_CopyPoint(RtPoint <i>sourcePoint</i> , RtPoint <i>destPoint</i>)

Intersection Testing Function

Test for intersection between line and plane:

void	N3DIntersectLinePlane(RtPoint <i>*endPoints</i> , RtPoint <i>planeNormal</i> , RtPoint <i>planePoint</i> , RtPoint <i>*intersection</i>)
------	--

Matrix Manipulation Functions

Efficient matrix multiplication:

```
void          N3DMultiplyMatrix(RtMatrix preTransform, RtMatrix postTransform,  
                                RtMatrix resultTransform)  
float        N3DInvertMatrix(RtMatrix theTransform, RtMatrix theInverse)
```

Transformation Functions

Transform between coordinate systems:

```
void          N3DMult3DPoint(RtPoint thePoint, RtMatrix theTransform, RtPoint newPoint)  
void          N3DMult3DPoints(RtPoint *thePoints, int pointCount,  
                               RtMatrix theTransform, RtPoint *newPoints)
```

Types and Constants

Defined Types

N3DProjectionType

```
typedef enum {  
    N3D_Perspective,  
    N3D_Orthographic  
} N3DProjectionType
```

N3DLightType

```
typedef enum {  
    N3D_AmbientLight,  
    N3D_PointLight,  
    N3D_DistantLight,  
    N3D_SpotLight  
} N3DLightType;
```

N3DAxis

```
typedef enum {  
    N3D_AllAxes,  
    N3D_XAxis,  
    N3D_YAxis,  
    N3D_ZAxis,  
    N3D_XYAxes,  
    N3D_XZAxes,  
    N3D_YZAxes  
} N3DAxis;
```

N3DHider

```
typedef enum {  
    N3D_HiddenRendering = 0,  
    N3D_InOrderRendering,  
    N3D_NoRendering  
} N3DHider
```

N3DShapeName

```
typedef struct {
    char id[6];
    char name;
} N3DShapeName
```

N3DSurfaceType

```
typedef enum {
    N3D_PointCloud = 0,
    N3D_WireFrame,
    N3D_ShadedWireFrame,
    N3D_FacetedSolids,
    N3D_SmoothSolids
} N3DSurfaceType;
```

SLOArgs

```
typedef struct {
    SLO_VISSYMDEF symb;
    union {
        float fval;
        RtPoint pval;
        NXColor cval;
        char *sval;
    } value;
} SLOArgs
```

Symbolic Constants

Matrix Constants

```
N3D_BOTH_CLEAN
N3D_CTM_DIRTY
N3D_CTM_INVERSE_DIRTY
N3D_CTM_BOTH_DIRTY
```

Global Variables

N3DIdentityMatrix

const RtMatrix N3DIdentityMatrix

N3DOrigin

const RtPoint N3DOrigin

N3DRIBPboardType

NXAtom N3DRIBPboardType

18 *Video*

Classes

NXLiveVideoView

Inherits From: View : Responder : Object

Initializing an NXLiveVideoView

– **initWithFrame:**(const NXRect *)*frameRect* Initializes the receiver

Freeing an NXLiveVideoView

– **free** Frees the receiver

Starting and Stopping Video Display

– **start:***sender* Starts video display in the video view

– **stop:***sender* Stops video display in the video view

Determining the Active State

– (BOOL)**isVideoActive** YES if the view is actively displaying video

Capturing Video as an NXImage

- (NXImage *)**grab** Grabs and returns an image from video in the view
- **grabIn:**(NXImage *)*theImage*
fromRect:(NXRect *)*sourceRect*
toRect:(NXRect *)*destRect* Grabs the image in the *sourceRect* and places it in *theImage* in *destRect*
- (BOOL)**doesGrabOnStop** YES if the receiver grabs an image when video stops
- **setGrabOnStop:**(BOOL)*flag* Sets the receiver to grab an image when video stops

Finding the Video Resource

- + (BOOL)**doesRectSupportVideo:**(const NXRect)*theRect*
standard:(int *)*theStandard* YES if the specified rectangle can display video
size:(NXSize *)*theSize*
- + (BOOL)**doesScreenSupportVideo:**(const NXScreen *)*theScreen*
standard:(int *)*theStandard* YES if the specified screen can display video
size:(NXSize *)*theSize*
- + (BOOL)**doesWindowSupportVideo:***theWindow*
standard:(int *)*theStandard* YES if the specified window can display video
size:(NXSize *)*theSize*
- + (const NXScreen *)**videoScreen** The screen best suited for video display

Getting the Video Rectangle

- **getSourceVideoRect:**(NXRect *)*sourceRect* Returns the visible portion of the video view

Selecting the Video Input Port

- **selectInput:**(int)*inputPortNumber* Sets the video input port
- (int)**numInputs** Returns the number of video ports

Setting the Output Mode:

- **setOutputMode:**(int)*outputMode* Sets the output mode of the video view

Controlling Input Video Quality

- **setInputBrightness:**(float)*brightness* Sets the input brightness value
- (float)**inputBrightness** Returns the input brightness value
- **setInputGamma:**(float)*inputGamma* Sets the input gamma value
- (float)**inputGamma** Returns the input gamma value
- **setInputHue:**(float)*hue* Sets the input hue
- (float)**inputHue** Returns the input hue

- **setInputSaturation:***(float)saturation* Sets the input saturation
- **(float)inputSaturation** Returns the input saturation
- **setInputSharpness:***(float)sharpness* Sets the input sharpness
- **(float)inputSharpness** Returns the input sharpness
- **resetPictureDefaults** Restores all input video settings to their defaults

Controlling Output Video Quality

- **setOutputGamma:***(float) outputGamma* Sets the output gamma
- **(float)outputGamma** Returns the output gamma

Getting the Video Standard

- **getVideoStandard:***(int *)standard
size:(NXSize *)vidRectSize* Returns the video standard and size

Setting Output Genlock

- **setOutputGenlocked:***(BOOL)locked* Locks video output to input signal
- **(BOOL)outputGenlocked** YES if video output is genlocked to input signal

Drawing

- **drawSelf:***(const NXRect *)rects :(int)rectCount* Overridden to assure video updates correctly
- **drawVideoBackground:***(const NXRect *)rects
:(int)rectCount* Invoked by drawSelf:: to assure correct video display

Setting a Delegate

- **delegate** Returns the receiver's delegate
- **setDelegate:** *anObject* Sets the receiver's delegate

Archiving

- **read:***(NXTypedStream *)stream* Reads the receiver from the stream
- **write:***(NXTypedStream *)stream* Writes the receiver to the stream

Delegate Method

- **videoDidActivate:***sender* Notifies the delegate when video activates
- **videoDidSuspend:***sender* Notifies the delegate when video stops

Types and Constants

Symbolic Constants

Input Selection

NX_VIDEOIN1
NX_VIDEOIN2
NX_VIDEOIN3

Output Source

NX_FROMINPUT
NX_FROMVIEW

Video Standard

NX_NTSCSIGNAL
NX_PALSIGNAL

Managing Changes

– **ok:***sender*

Implement in your subclass to commit the changes that the user has made to the selected item

– **revert:***sender*

Implement in your subclass to load data into the inspector's display

– **textDidChange:***sender*

Sends the `WMInspector` a **touch:** message

– **touch:***sender*

Changes the image in the Inspector panel's close box to a broken "X"

20 *Mach Functions*

C-Thread Functions

This section contains a summary of the C-thread functions, which are described in detail in the *NeXTSTEP Operating System Software* manual.

To use the C-thread functions, include in your source files the header file **mach/threads.h**:

```
#include <mach/threads.h>
```

Basic C-Thread Functions

Control a thread:

pthread_t	pthread_fork (any_t (*function)(), any_t arg)
any_t	pthread_join (pthread_t t)
void	pthread_detach (pthread_t t)
void	pthread_yield ()
kern_return_t	pthread_abort (pthread_t t)
void	pthread_exit (any_t result)

Get access to a thread:

pthread_t	pthread_self ()
pthread_t	pthread_thread (pthread_t t)

Associate a string with a thread:

```
char *      pthread_name(pthread_t t)
void       pthread_set_name(pthread_t t, char *name)
```

Associate data with a thread:

```
any_t      pthread_data(pthread_t t)
void       pthread_set_data(pthread_t t, any_t data)
```

Change the priority of a thread:

```
kern_return_t pthread_max_priority(pthread_t t, processor_set_t processor_set, int
                                     max_priority)
kern_return_t pthread_priority(pthread_t t, int priority, boolean_t set_max)
```

Get or set the UNIX™ error number of this thread:

```
int         pthread_errno()
void       pthread_set_errno_self(int error)
```

Get or set the maximum number of threads in this task:

```
int         pthread_limit()
void       pthread_set_limit(int limit)
int        pthread_count()
```

Mutex Functions

Control a mutex:

```
mutex_t     mutex_alloc()
void       mutex_init(struct mutex *m)
void       mutex_clear(struct mutex *m)
void       mutex_free(mutex_t m)
```

Associate a string with a mutex:

```
char *      mutex_name(mutex_t m)
void        mutex_set_name(mutex_t m, char *name)
```

Synchronize a mutex:

```
void        mutex_lock(mutex_t m)
int         mutex_try_lock(mutex_t m)
void        mutex_unlock(mutex_t m)
```

Condition Functions

Control a condition variable:

```
condition_t condition_alloc()
void         condition_init(struct condition *c)
void         condition_clear(struct condition *c)
void         condition_free(condition_t c)
```

Associate a string with a condition variable:

```
char *      condition_name(condition_t c)
void        condition_set_name(condition_t c, char *name)
```

Synchronize a condition variable:

```
void        condition_broadcast(condition_t c)
void        condition_signal(condition_t c)
void        condition_wait(condition_t c, mutex_t m)
```

Mach Kernel Functions

The Mach kernel is introduced in the *Operating System Software* manual. This section contains a summary of the Mach kernel functions, which are described in detail in *Operating System Software*.

To use the Mach kernel functions, include in your source files the header file **mach/mach.h**:

```
#include <mach/mach.h>
```

To use the Mach message functions, also include in your source files the header file **mach/message.h**:

```
#include <mach/mach.h>
#include <mach/message.h>
```

To use the Mach error string functions, include the header file **mach/error.h**, in addition to **mach/mach.h**:

```
#include <mach/mach.h>
#include <mach/error.h>
```

Task Functions

Control a task:

kern_return_t	task_create (task_t <i>parent_task</i> , boolean_t <i>inherit_memory</i> , task_t * <i>child_task</i>)
kern_return_t	task_suspend (task_t <i>target_task</i>)
kern_return_t	task_resume (task_t <i>target_task</i>)
kern_return_t	task_terminate (task_t <i>target_task</i>)

Access a task or its threads:

kern_return_t	task_threads (task_t <i>target_task</i> , thread_array_t * <i>thread_list</i> , unsigned int * <i>thread_count</i>)
kern_return_t	task_info (task_t <i>target_task</i> , int <i>flavor</i> , task_info_t <i>task_info</i> , unsigned int * <i>task_info_count</i>)

Access a task's special ports:

kern_return_t	task_get_special_port (task_t <i>task</i> , int <i>which_port</i> , port_t * <i>special_port</i>)
kern_return_t	task_set_special_port (task_t <i>task</i> , int <i>which_port</i> , port_t <i>special_port</i>)
port_t	task_notify ()
task_t	task_self ()

Translate between Mach task and UNIX process ID:

kern_return_t	task_by_unix_pid (task_t <i>task</i> , int <i>pid</i> , task_t * <i>result_task</i>)
kern_return_t	unix_pid (task_t <i>target_task</i> , int * <i>pid</i>)

Set a task's scheduling priority:

kern_return_t	task_priority (task_t <i>task</i> , int <i>priority</i> , boolean_t <i>change_threads</i>)
---------------	--

Multiprocessor functions (not useful on single-processor systems):

kern_return_t	task_assign (task_t <i>task</i> , processor_set_t <i>new_processor_set</i> , boolean_t <i>assign_threads</i>)
kern_return_t	task_assign_default (task_t <i>task</i> , boolean_t <i>assign_threads</i>)
kern_return_t	task_get_assignment (task_t <i>task</i> , processor_set_t * <i>processor_set</i>)

Thread Functions

Control a thread:

kern_return_t	thread_create (task_t <i>parent_task</i> , thread_t * <i>child_thread</i>)
kern_return_t	thread_suspend (thread_t <i>target_thread</i>)
kern_return_t	thread_resume (thread_t <i>target_thread</i>)
kern_return_t	thread_terminate (thread_t <i>target_thread</i>)
kern_return_t	thread_abort (thread_t <i>target_thread</i>)

Get information about a thread:

kern_return_t	thread_info (thread_t <i>target_thread</i> , int <i>flavor</i> , thread_info_t <i>thread_info</i> , unsigned int * <i>thread_info_count</i>)
---------------	---

Access a thread's state:

```
kern_return_t    thread_get_state(thread_t target_thread, int flavor, thread_state_data_t old_state,
                                unsigned int *old_state_count)
kern_return_t    thread_set_state(thread_t target_thread, int flavor, thread_state_data_t new_state,
                                unsigned int new_state_count)
```

Access a thread's special ports:

```
kern_return_t    thread_get_special_port(thread_t thread, int which_port, port_t *special_port)
kern_return_t    thread_set_special_port(thread_t thread, int which_port, port_t special_port)
port_t          thread_reply()
thread_t        thread_self()
```

Affect the scheduling policy or priority of a thread:

```
kern_return_t    thread_policy(thread_t thread, int policy, int data)
kern_return_t    thread_priority(thread_t thread, int priority, boolean_t set_max)
kern_return_t    thread_max_priority(thread_t thread, processor_set_t processor_set, int priority)
kern_return_t    thread_switch(thread_t new_thread, int option, int time)
```

Multiprocessor functions (not useful on single-processor systems):

```
kern_return_t    thread_assign(thread_t thread, processor_set_t new_processor_set)
kern_return_t    thread_assign_default(thread_t thread)
kern_return_t    thread_get_assignment(thread_t thread, processor_set_t *processor_set)
```

Port Functions

Control a port:

```
kern_return_t    port_allocate(task_t task, port_name_t *port_name)
kern_return_t    port_deallocate(task_t task, port_name_t port_name)
kern_return_t    port_rename(task_t task, port_name_t old_name, port_name_t new_name)
kern_return_t    port_set_backlog(task_t task, port_name_t port_name, int backlog)
```

Give a task rights to a port:

```
kern_return_t    port_insert_receive(task_t task, port_t my_port, port_name_t its_name)
kern_return_t    port_insert_send(task_t task, port_t my_port, port_name_t its_name)
```

Remove a task's rights to a port:

```
kern_return_t    port_extract_receive(task_t task, port_name_t its_name, port_t *its_port)
kern_return_t    port_extract_send(task_t task, port_name_t its_name, port_t *its_port)
```

Get information about a port:

```
kern_return_t    port_status(task_t task, port_name_t port_name, port_set_name_t
                    *port_set_name, int *num_msgs, int *backlog, boolean_t *owner,
                    boolean_t *receiver)
kern_return_t    port_type(task_t task, port_name_t port_name, port_type_t *port_type)
```

Get information about a task's port name space:

```
kern_return_t    port_names(task_t task, port_name_array_t *port_names,
                    unsigned int *port_names_count, port_type_array_t *port_types,
                    unsigned int *port_types_count)
```

Control a port set:

```
kern_return_t    port_set_allocate(task_t task, port_set_name_t *set_name)
kern_return_t    port_set_backup(task_t task, port_name_t port_name, port_t backup,
                    port_t *previous)
kern_return_t    port_set_deallocate(task_t task, port_set_name_t set_name)
```

Access or modify a port set:

```
kern_return_t    port_set_add(task_t task, port_set_name_t set_name, port_name_t port_name)
kern_return_t    port_set_remove(task_t task, port_name_t port_name)
kern_return_t    port_set_status(task_t task, port_set_name_t set_name,
                    port_name_array_t *members, unsigned int *members_count)
```

Message Functions

Send or receive a message:

```
msg_return_t    msg_send(msg_header_t *header, msg_option_t option, msg_timeout_t timeout)
msg_return_t    msg_receive(msg_header_t *header, msg_option_t option, msg_timeout_t timeout)
msg_return_t    msg_rpc(msg_header_t *header, msg_option_t option, msg_size_t rcv_size,
                    msg_timeout_t send_timeout, msg_timeout_t rcv_timeout)
```


Get information about a processor set:

```
kern_return_t    processor_set_info(processor_set_t processor_set, int flavor, host_t *host,
                                   processor_set_info_t processor_set_info,
                                   unsigned int *processor_set_info_count)
kern_return_t    processor_set_tasks(processor_set_t processor_set, task_array_t *task_list,
                                   unsigned int *task_count)
kern_return_t    processor_set_threads(processor_set_t processor_set, thread_array_t *thread_list,
                                   unsigned int *thread_count)
```

Multiprocessor functions (not useful on single-processor systems):

```
kern_return_t    processor_assign(processor_t processor, processor_set_t new_processor_set,
                                   boolean_t wait)
kern_return_t    processor_control(processor_t processor, processor_info_t info, long *count)
kern_return_t    processor_exit(processor_t processor)
kern_return_t    processor_get_assignment(processor_t processor,
                                   processor_set_t *processor_set)
kern_return_t    processor_start(processor_t processor)
kern_return_t    processor_set_create(host_t host, port_t *new_set, port_t *new_name)
kern_return_t    processor_set_destroy(processor_set_t processor_set)
kern_return_t    processor_set_max_priority(processor_set_t processor_set, int max_priority,
                                   boolean_t change_threads)
```

Exception Functions

Raise or handle exceptions:

```
kern_return_t    exception_raise(port_t exception_port, port_t clear_port, port_t thread, port_t task,
                                   int exception, int code, int subcode)
void             mach_NeXT_exception(char *string, int exception, int code, int subcode)
char *          mach_NeXT_exception_string(int exception, int code, int subcode)
boolean_t       exc_server(msg_header_t *in, msg_header_t *out)
```

Error String Functions

Display or get a Mach error string:

```
void            mach_error(char *string, kern_return_t errno)
char *          mach_error_string(kern_return_t errno)
```

Network Name Server Functions

This section summarizes the Mach Network Name Server functions, which are not part of the Mach kernel. For more information see the *Operating System Software* manual. To use the Network Name Server functions, include in your source files the header files **mach/mach.h** and **servers/netname.h**:

```
#include <mach/mach.h>
#include <servers/netname.h>
```

Check a name into or out of the local name space:

```
kern_return_t    netname_check_in(port_t server_port, netname_name_t port_name,
                                   port_t signature, port_t port_id)
kern_return_t    netname_check_out(port_t server_port, netname_name_t port_name,
                                   port_t signature)
```

Look up a name on a specific host:

```
kern_return_t    netname_look_up(port_t server_port, netname_name_t host_name,
                                   netname_name_t port_name, port_t *port_id)
```

Bootstrap Server Functions

This section contains a summary of the Bootstrap Server functions, which are described in detail in the *Operating System Software* manual.

To use a Bootstrap Server function, include in your source files the header files **mach/mach.h** and **servers/bootstrap.h**:

```
#include <mach/mach.h>
#include <servers/bootstrap.h>
```

Look up a service:

```
kern_return_t    bootstrap_look_up(port_t bootstrap_port, name_t service_name,
                                   port_t *service_port)
kern_return_t    bootstrap_look_up_array(port_t bootstrap_port, name_array_t service_names,
                                         unsigned int service_names_count, port_array_t *service_port,
                                         unsigned int *service_ports_count,
                                         boolean_t *all_services_known)
```

Find out whether a service is active:

```
kern_return_t    bootstrap_status(port_t bootstrap_port, name_t service_name,
                                  boolean_t *service_active)
```

Get information about all known services:

```
kern_return_t    bootstrap_info(port_t bootstrap_port, name_array_t *service_names, unsigned
                                int *service_names_count, name_array_t *server_names,
                                unsigned int *server_names_count, bool_array_t *service_active,
                                unsigned int *service_active_count)
```

Check in a service:

```
kern_return_t    bootstrap_check_in(port_t bootstrap_port, name_t service_name,
                                     port_all_t *service_port)
kern_return_t    bootstrap_create_service(port_t bootstrap_port, name_t service_name,
                                         port_t *service_port)
kern_return_t    bootstrap_register(port_t bootstrap_port, name_t service_name, port_t
                                    service_port)
```

Create a new bootstrap port:

```
kern_return_t    bootstrap_subset(port_t bootstrap_port, port_t requestor_port,
                                   port_t *subset_port)
```

Kernel-Server Loader Functions

This section contains a summary of the kernel-server loader functions, which are described in detail in the *Operating System Software* manual. Use these functions in a user-level program to communicate with the kernel-server loader, which controls all loadable kernel servers.

To use these functions, include in your source files the header files **mach/mach.h** and **kernserv/kern_loader_types.h**. Most functions also require that you include the header file **kernserv/kern_loader.h**; the error functions require that you include the header file **kernserv/kern_loader_error.h**. Two other header files may be necessary: **kernserv/kern_loader_reply_handler.h** and **kernserv/kern_loader_reply.h**.

To use these functions, you must compile with the kernload library. For example:

```
cc myprog.c -lkernload
```

Get a port to use in other kernel-server loader functions:

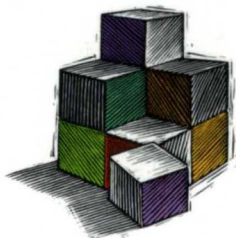
```
kern_return_t    kern_loader_look_up(port_t *loader_port)
kern_return_t    kern_loader_server_com_port(port_t loader_port, port_t task_port,
                                             server_name_t server_name, port_t *server_com_port)
```

Get or display an error string:

```
void             kern_loader_error(const char *string, kern_return_t errno)
const char *     kern_loader_error_string(kern_return_t errno)
```

Affect the runnability of a loadable kernel server:

```
kern_return_t    kern_loader_add_server(port_t loader_port, port_t task_port,
                                         server_reloc_t server_reloc)
kern_return_t    kern_loader_delete_server(port_t loader_port, port_t task_port,
                                           server_name_t server_name)
kern_return_t    kern_loader_load_server(port_t loader_port, server_name_t server_name)
kern_return_t    kern_loader_unload_server(port_t loader_port, port_t task_port,
                                           server_name_t server_name)
```

NEXTSTEP PROGRAMMING INTERFACE SUMMARY: RELEASE 3

NeXTSTEP is the object-oriented programming environment that speeds the development of all kinds of software—from mission-critical custom applications for business to advanced research projects for academia. NeXTSTEP offers building blocks that implement essential behavior in a variety of application areas—including database management, telecommunications and networking, and high-quality 2D and 3D graphics.

The **NeXTSTEP Programming Interface Summary** covers all components of NeXTSTEP, including:

- Application Kit
- Database and Indexing Kits
- NetInfo Kit
- 3D Graphics Kit

The **NeXTSTEP Developer's Library** is essential reading for every NeXTSTEP enthusiast, providing authoritative, in-depth descriptions of the NeXTSTEP programming environment. Other titles in the **NeXTSTEP Developer's Library** include:

- **NeXTSTEP General Reference: Release 3, Volumes 1 and 2**
- **NeXTSTEP User Interface Guidelines: Release 3**
- **NeXTSTEP Development Tools and Techniques: Release 3**
- **NeXTSTEP Operating System Software: Release 3**
- **NeXTSTEP Object-Oriented Programming and the Objective C Language: Release 3**
- **NeXTSTEP Network and System Administration: Release 3**

NeXT develops and markets the industry-acclaimed NeXTSTEP object-oriented software for industry-standard computer architectures.

NEXTSTEP

Object Oriented Software



9 780201 632538

ISBN 0-201-63253-5