



## PROGRAMMING INTERFACE SUMMARY



# NEXTSTEP<sup>TM</sup>

---

*Object-Oriented Software*

# NeXTSTEP<sup>TM</sup> 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. NXRTFDErrorHandler, 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.

<b>6-1</b>	<b>Chapter 6: Distributed Objects</b>
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.

<b>7-1</b>	<b>Chapter 7: Indexing Kit</b>
7-1	Classes IXAttributeParser, p. 7-1. IXAttributeQuery, p. 7-2. IXAttributeReader, p. 7-3. IXBTTree, p. 7-4. IXBTTreeCursor, 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. IXNameAnd FileAccess, 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. IBCConnectors, p. 8-5. IBDocuments, p. 8-5.  
IBDocumentControllers, p. 8-7. IBEEditors, 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. NXSenderIsValid, 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. NILoginPage, 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
  - WMIInspector, 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) <b>isEqual:anObject</b>	Returns whether the receiver and <i>anObject</i> are the same
– (unsigned int) <b>hash</b>	Returns an unsigned integer unique to the receiver
– <b>self</b>	Returns the receiver, an instance
– (const char *) <b>name</b>	Implemented by subclasses to return the receiver's name
– (void) <b>printForDebugger:(NXStream *)stream</b>	Writes information identifying the receiver to <i>stream</i>

## Testing Inheritance Relationships

- **(BOOL)isKindOfClass:*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

<code>- (BOOL)respondsToSelector:(SEL)aSelector</code>	Returns whether the receiver can respond to <i>aSelector</i>
<code>+ (BOOL)instancesRespondToSelector:(SEL)aSelector</code>	Returns whether instances can respond to <i>aSelector</i>

## Testing for Protocol Conformance

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

## Sending Messages Determined at Run Time

- **perform:(SEL)*aSelector***
- **perform:(SEL)*aSelector* with:*anObject***
- **perform:(SEL)*aSelector* with:*anObject* with:*anotherObject***

Sends an *aSelector* message to the receiver  
Sends an *aSelector* message with one argument  
Sends an *aSelector* message with two arguments

## Forwarding Messages

- **forward:(SEL)*aSelector* :(marg\_list)*argFrame***
- **performv:(SEL)*aSelector* :(marg\_list)*argFrame***

Implemented by subclasses to forward messages  
Sends an *aSelector* message with *argFrame* arguments

## Obtaining Method Information

- **(IMP)methodFor:(SEL)*aSelector***
- + **(IMP)instanceMethodFor:(SEL)*aSelector***
- **(struct objc\_method\_description \*)descriptionForMethod:(SEL)*aSelector***
- + **(struct objc\_method\_description \*)descriptionForInstanceMethod:(SEL)*aSelector***

Locates the receiver's implementation of *aSelector*  
Locates the implementation of *aSelector*  
Returns information about the *aSelector* method  
Returns information about the *aSelector* instance method

## Posing

- + **poseAs:*aClassObject***

Substitutes the receiving class for *aClassObject*

## Enforcing Intentions

- **notImplemented:(SEL)*aSelector***
- **subclassResponsibility:(SEL)*aSelector***

Indicates that *aSelector* isn't fully implemented  
Generates an error if *aSelector* isn't implemented

## Error Handling

- **doesNotRecognize:(SEL)*aSelector***
- **error:(const char \*)*aString*, ...**

Generates an unrecognized-selector error message  
Generates a formatted error message using *aString*

## Dynamic Loading

- + **finishLoading:(struct mach\_header \*)*header***
- + **startUnloading**

Implemented by a newly loaded class or category  
Implemented by a class or category about to be unloaded

## Archiving

– <b>read:(NXTypedStream *)stream</b>	Implemented by subclasses to read receiver from <i>stream</i>
– <b>write:(NXTypedStream *)stream</b>	Implemented by subclasses to write the receiver to <i>stream</i>
– <b>startArchiving:(NXTypedStream *)stream</b>	Implemented by subclasses to prepare for archiving
– <b>awake</b>	Implemented by subclasses to reinitialize the receiver
– <b>finishUnarchiving</b>	Implemented by subclasses to replace the receiver
+ <b>setVersion:(int)<i>aVersion</i></b>	Sets the class version number to <i>aVersion</i>
+ <b>(int)version</b>	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

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

#### Manipulating ActionCell Values

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

– (int)intValue	Returns the ActionCell's contents as an <b>int</b>
– <b>setStringValue:</b> (const char *) <i>aString</i>	Sets the ActionCell's contents to a copy of <i>aString</i>
– <b>setStringValueNoCopy:</b> (char *) <i>aString</i> <b>shouldFree:</b> (BOOL) <i>flag</i>	Sets the ActionCell's contents to a <i>aString</i> ; will free the string when freed if <i>flag</i> is YES
– (const char *) <b>stringValue</b>	Returns the ActionCell's contents as a string

## Displaying

– <b>drawSelf:</b> (const NXRect *) <i>cellFrame</i> <b>inView:</b> <i>controlView</i>	Draws the ActionCell in <i>controlView</i>
– <b>controlView</b>	Returns the View in which the ActionCell was most recently drawn (usually a Control)

## Target and Action

– <b>setAction:</b> (SEL) <i>aSelector</i>	Sets the ActionCell's action method to <i>aSelector</i>
– (SEL) <b>action</b>	Returns the ActionCell's action method
– <b>setTarget:</b> <i>anObject</i>	Sets the ActionCell's target object to <i>anObject</i>
– <b>target</b>	Returns the ActionCell's target object

## Assigning a Tag

– <b>setTag:</b> (int) <i>anInt</i>	Sets the ActionCell's tag to <i>anInt</i>
– (int) <b>tag</b>	Returns the ActionCell's tag

## Archiving

– <b>read:</b> (NXTypedStream *) <i>stream</i>	Reads the ActionCell from <i>stream</i>
– <b>write:</b> (NXTypedStream *) <i>stream</i>	Writes the ActionCell to <i>stream</i>

## Application

**Inherits From:** Responder : Object

### Initializing the Class Object

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

## Responding to Notification (Override in a Subclass)

- |   |   |
|---|---|
| – <b>applicationDidLaunch:<i>appName</i></b>    | Notice that <i>appName</i> launched; your arbitrary return    |
| – <b>applicationDidTerminate:<i>appName</i></b> | Notice that <i>appName</i> ended; your arbitrary return       |
| – <b>applicationWillLaunch:<i>appName</i></b>   | Notice that <i>appName</i> will launch; your arbitrary return |

## Changing the Active Application

- |  |   |
|--|---|
| – <b>(int)activate:(int)<i>contextNumber</i></b> | Makes <i>contextNumber</i> the active application |
| – <b>(int)activateSelf:(BOOL)<i>flag</i></b>     | Makes this the active application                 |
| – <b>(int)activeApp</b>                          | Returns context number of the active application  |
| – <b>becomeActiveApp</b>                         | Responds to activating the application            |
| – <b>deactivateSelf</b>                          | Deactivates the application                       |
| – <b>(BOOL)isActive</b>                          | Returns whether this is the active application    |
| – <b>resignActiveApp</b>                         | Responds to deactivating the application          |

## Running the Event Loop

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

## Getting and Peeking at Events

- |  |  |
|--|--|
| – <b>(NXEvent *)currentEvent</b>   | Returns pointer to the current event                   |
| – <b>(NXEvent *)getNextEvent:(int)<i>mask</i></b>  | Returns pointer to the next event matching <i>mask</i> |
| – <b>(NXEvent *)getNextEvent:(int)<i>mask</i> waitFor:(double)<i>timeout</i> threshold:(int)<i>level</i></b> | Returns pointer to the next event matching <i>mask</i> |

- (NXEvent \*)**peekAndGetNextEvent:(int)mask**
- (NXEvent \*)**peekNextEvent:(int)mask into:(NXEvent \*)eventPtr**
- (NXEvent \*)**peekNextEvent:(int)mask into:(NXEvent \*)eventPtr waitFor:(float)timeout threshold:(int)level**

Returns pointer to the next event matching *mask*  
 Returns pointer to the next event matching *mask*  
 Returns pointer to the next event matching *mask*

## Journaling

- (BOOL)**isJournalable**
- **setJournalable:(BOOL)flag**
- **masterJournaler**
- **slaveJournaler**

Returns whether the application can be journaled  
 Sets whether the application can be journaled  
 Returns the controlling NXJournaler object  
 Returns the controlled NXJournaler object

## Handling User Actions and Events

- **applicationDefined:(NXEvent \*)theEvent**
- **hide:sender**
- (BOOL)**isHidden**
- (int)**unhide**
- **unhide:sender**
- **unhideWithoutActivation:sender**
- **powerOff:(NXEvent \*)theEvent**
- (int)**powerOffIn:(int)ms andSave:(int)aFlag**
- **rightMouseDown:(NXEvent \*)theEvent**
- (int)**unmounting:(const char \*)fullPath ok:(int \*)flag**

Responds to an application-defined event  
 Hides all the application's windows  
 YES if windows are hidden  
 Responds to message to unhide windows  
 Restores hidden windows to the screen  
 Restores hidden windows without activating their owner  
 Responds to a power-off subevent  
 Responds to message from Workspace Manager  
 Causes main menu to pop up under the mouse  
 Responds to message from Workspace Manager

## Sending Action Messages

- (BOOL)**sendAction:(SEL)aSelector to:aTarget from:sender**
- (BOOL)**tryToPerform:(SEL)aSelector with:anObject**
- **calcTargetForAction:(SEL)theAction**

Sends an action message to *aTarget* or up the responder chain  
 Attempts to send a message to the application or the delegate  
 Looks up receiver for *theAction* message

## Remote Messaging

– <b>setAppListener:</b> <i>aListener</i>	Makes <i>aListener</i> the application's Listener
– <b>appListener</b>	Returns the Listener for this application
– <b>setAppSpeaker:</b> <i>aSpeaker</i>	Makes <i>aSpeaker</i> the application's Speaker
– <b>appSpeaker</b>	Returns the Speaker for this application
– (const char *) <b>appListenerPortName</b>	Returns name used to register Listener with name server
– (port_t) <b>replyPort</b>	Returns port for synchronous return messages

## Managing Windows

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

## Managing the Windows Menu

– <b>setWindowsMenu:</b> <i>aMenu</i>	Sets the Windows menu
– <b>windowsMenu</b>	Returns the Windows menu
– <b>arrangeInFront:</b>	Orders all registered Windows to the front
– <b>addWindowsItem:</b> <i>aWindow title:(const char *)aString filename:(BOOL)isFilename</i>	Adds a menu item for <i>aWindow</i>
– <b>removeWindowsItem:</b> <i>aWindow</i>	Removes the Window's menu item
– <b>changeWindowsItem:</b> <i>aWindow title:(const char *)aString filename:(BOOL)isFilename</i>	Changes the Window's menu item
– <b>updateWindowsItem:</b> <i>aWindow</i>	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*  
**andReturnType:**(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

- (DPSCContext)**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**
- **unmounted:(const char \*)fullPath**

Notice that device at fullPath has been mounted; returns 0  
(unless overridden in a subclass)

Notice that device at fullPath has been unmounted;  
returns 0 (unless overridden in a subclass)

## Printing

- **setPrintInfo:info**
- **printInfo**
- **runPageLayout:sender**

Makes *info* the application's PrintInfo object

Returns the application's PrintInfo object

Runs the application's PageLayout panel

## Color

- **orderFrontColorPanel:sender**
- **doesImportAlpha**
- **setImportAlpha:**

Brings up the color panel

YES if application responds to opacity in imported colors

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**
- **delegate**

Makes *anObject* the Application's delegate

Returns the Application's delegate

## Implemented by the Delegate

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

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

- **initFrame:(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**  
– **read:(NXTypedStream \*)stream**  
– **write:(NXTypedStream \*)stream**

Lays out the graphic elements of the Box  
Reads the Box object from the typed stream  
Writes the Box object to the typed stream

---

## Button

**Inherits From:** Control : View : Responder : Object

### Initializing the Button Factory

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

### Initializing a Button

– **init**  
– **initFrame:(const NXRect \*)frameRect**  
– **initFrame:(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 with title “Button”  
– **initFrame:(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*  
– **initFrame:(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*  
– **initFrame:(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 NXImage 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 NXImage named *iconName* the alternate icon  
– **(const char \*)altIcon** Returns the name of the Button's alternate icon  
– **setImage:*image*** Makes the NXImage *image* the Button's icon  
– **image** Returns the Button's image  
– **setAltImage:*altImage*** Makes the NXImage *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

– <b>setTransparent:(BOOL)<i>flag</i></b>	Sets whether the Button is transparent
– <b>(BOOL)isTransparent</b>	Returns whether the Button is transparent
– <b>setBordered:(BOOL)<i>flag</i></b>	Sets whether the Button has a bezeled border
– <b>(BOOL)isBordered</b>	Returns whether the Button has a bezeled border

## Displaying

– <b>display</b>	Displays the Button
– <b>highlight:(BOOL)<i>flag</i></b>	Highlights (or unhighlights) the Button according to <i>flag</i>

## Setting the Key Equivalent

– <b>setKeyEquivalent:(unsigned short)<i>charCode</i></b>	Makes <i>charCode</i> the Button's key equivalent
– <b>(unsigned short)keyEquivalent</b>	Returns the Button's key equivalent

## Handling Events and Action Messages

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

## Setting the Sound

– <b>setSound:<i>soundObject</i></b>	Sets the Sound played when the Button is pressed
– <b>sound</b>	Returns the Sound played when the Button is pressed

---

## ButtonCell

**Inherits From:** ActionCell : Cell : Object

### Initializing, Copying, and Freeing a ButtonCell

– <b>init</b>	Initializes a new ButtonCell with title “Button”
– <b>initTextCell:(const char *)<i>aString</i></b>	Initializes a new ButtonCell with title <i>aString</i>

- **initIconCell:(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)***aInt*
- **(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 *aInt*  
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***
- **(int)getParameter:(int)*aParameter***

Sets various flag values  
Returns various flag values

## Modifying Graphic Attributes

- **setBordered:(BOOL)*flag***
- **(BOOL)isBordered**
- **setTransparent:(BOOL)*flag***
- **(BOOL)isTransparent**
- **(BOOL)isOpaque**

Sets whether the ButtonCell has a bezeled border  
Returns whether the ButtonCell has a bezeled border  
Sets whether the ButtonCell is transparent  
Returns whether the ButtonCell is transparent  
Returns whether receiver is opaque

## Modifying Graphic Attributes

- **setType:(int)*aType***
- **setHighlightsBy:(int)*aType***
- **(int)highlightsBy**
- **setShowsStateBy:(int)*aType***
- **(int)showsStateBy**

Sets the ButtonCell's display behavior  
Sets how the ButtonCell highlights  
Returns how the ButtonCell highlights  
Sets how the ButtonCell shows its alternate state  
Returns how ButtonCell shows its alternate state

## Simulating a Click

- **performClick:*sender***

Simulates clicking the ButtonCell

## Displaying

- **drawInside:(const NXRect \*)*aRect*  
inView:*controlView***
- **drawSelf:(const NXRect \*)*cellFrame*  
inView:*controlView***
- **highlight:(const NXRect \*)*cellFrame*  
inView:*controlView*  
lit:(BOOL)*flag***

Draws the inside of the ButtonCell  
Draws the ButtonCell  
Highlights the ButtonCell

## Archiving

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

Reads the ButtonCell from *stream*  
Writes the ButtonCell to *stream*

# Cell

**Inherits From:** Object

## Initializing, Copying, and Freeing a Cell

- **init**
- **initIconCell:(const char \*)iconName**
- **initTextCell:(const char \*)aString**
- **copyFromZone:(NXZone \*)zone**
- **free**

Initializes a new Cell  
Initializes a new Cell with the NXImage named *iconName*  
Initializes a new Cell with title *aString*  
Returns a copy of the receiving Cell from *zone*  
Deallocates the Cell

## Determining Component Sizes

- **calcCellSize:(NXSize \*)theSize**
- **calcCellSize:(NXSize \*)theSize  
inRect:(const NXRect \*)aRect**
- **calcDrawInfo:(const NXRect \*)aRect**
- **getDrawRect:(NXRect \*)theRect**
- **getIconRect:(NXRect \*)theRect**
- **getTitleRect:(NXRect \*)theRect**

Returns the minimum size needed to display the Cell  
Returns the minimum size needed to display the Cell  
Implemented by subclasses to recalculate drawing sizes  
Returns the rectangle the Cell draws in  
Returns the rectangle that an icon is drawn in  
Returns the rectangle that a title is drawn in

## Setting the Cell's Type

- **setType:(int)aType**
- **(int)type**

Sets the Cell's type to *aType*  
Returns the Cell's type

## Setting the Cell's State

- **setState:(int)value**
- **incrementState**
- **(int)state**

Sets the state of the Cell to *value* (0 or 1)  
Increments the state of the Cell  
Returns the state of the Cell (0 or 1)

## Enabling and Disabling the Cell

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

Sets whether the Cell reacts to mouse events  
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**
- **setValue:(const char \*)aString**
- **setValueNoCopy:(const char \*)aString**
- **setValueNoCopy:(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

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

## Validating Input

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

## Formatting Data

– <b>setFloatingPointFormat:</b> (BOOL) <i>autoRange</i> <i>left:(unsigned)leftDigits</i> <i>right:(unsigned)rightDigits</i>	Sets the display format for floating point values
--	---

## Modifying Graphic Attributes

– <b>setBezeled:</b> (BOOL) <i>flag</i>	Sets whether the Cell has a beveled border
– (BOOL) <b>isBezeled</b>	Returns whether the Cell has a beveled border
– <b>setBordered:</b> (BOOL) <i>flag</i>	Sets whether the Cell has a plain border
– (BOOL) <b>isBordered</b>	Returns whether Cell has a plain border
– (BOOL) <b>isOpaque</b>	Returns whether the Cell is opaque

## Setting Parameters

– <b>setParameter:</b> (int) <i>aParameter</i> <b>to:</b> (int) <i>value</i>	Sets various Cell flags
– (int) <b>getParameter:</b> (int) <i>aParameter</i>	Returns various Cell flag values

## Displaying

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

## Target and Action

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

## Assigning a Tag

– <b>setTag:(int)anInt</b>	Implemented by subclasses to set an identifier tag
– <b>(int&gt;tag</b>	Implemented by subclasses to return the identifier tag

## Handling Keyboard Alternatives

– <b>(unsigned short)keyEquivalent</b>	Implemented by subclasses to return a key equivalent
--	--

## Tracking the Mouse

– <b>(BOOL)prefersTrackingUntilMouseUp</b>	Returns NO, so tracking stop when the mouse leaves the Cell; subclasses may override
– <b>(int)mouseDownFlags</b>	Returns the event flags set at the start of mouse tracking
– <b>getPeriodicDelay:(float*)delay andInterval:(float*)interval</b>	Returns repeat values for continuous sending of the action
– <b>(BOOL)trackMouse:(NXEvent *)theEvent inRect:(const NXRect *)cellFrame ofView:aView</b>	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  
inView:aView**      Returns whether tracking should continue based on *lastPoint* and *currentPoint* within *aView*
  - **stopTracking:(const NXPoint \*)lastPoint  
at:(const NXPoint \*)stopPoint  
inView:aView  
mouseIsUp:(BOOL)flag**      Allows the Cell to update itself to end tracking, based on *lastPoint*, *stopPoint*, within *aView*; *flag* is YES if the this method was invoked because mouse went up

## Managing the Cursor

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

## 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

- |                                       |                                     |
|---------------------------------------|-------------------------------------|
| - initFrame:(const NXRect *)frameRect | Initializes a new ClipView instance |
| - free                                | Releases the ClipView's storage     |

## Modifying the Frame Rectangle

- **moveTo:**(NXCoord)*x* :(NXCoord)*y*
- **rotateTo:**(NXCoord)*angle*
- **sizeTo:**(NXCoord)*width* :(NXCoord)*height*

Moves the origin of the frame rectangle  
Overridden to disable rotation  
Resizes the ClipView's frame

## Modifying the Coordinate System

- **rotate:**(NXCoord)*angle*
- **scale:**(NXCoord)*x* :(NXCoord)*y*
- **setDrawOrigin:**(NXCoord)*x* :(NXCoord)*y*
- **setDrawRotation:**(NXCoord)*angle*
- **setDrawSize:**(NXCoord)*width* :(NXCoord)*height*
- **translate:**(NXCoord)*x* :(NXCoord)*y*

Overridden to disable rotation  
Rescales the coordinate system  
Sets the origin of the coordinate system  
Disables rotation of the coordinate system  
Scales the coordinate system  
Shifts the coordinate system

## Managing Component Views

- **docView**
- **setDocView:***aView*
- **getDocRect:**(NXRect \*)*aRect*
- **getDocVisibleRect:**(NXRect \*)*aRect*
- **resetCursorRects**
- **setDocCursor:***anObj*

Returns the ClipView's document view  
Makes *aView* the ClipView's document view  
Returns the document rectangle  
Gets the visible portion of the document view  
Resets the cursor rectangle for the document view  
Sets the cursor for the document view

## Modifying Graphic Attributes and Displaying

- (float)**backgroundGray**
- **setBackgroundGray:**(float)*value*
- (NXColor)**backgroundColor**
- **setBackgroundColor:**(NXColor)*color*
- **drawSelf:**(const NXRect \*)*rects* :(int )*rectCount*

Returns the ClipView's background gray  
Sets the ClipView's background gray  
Returns the ClipView's background color  
Sets the ClipView's background color  
Fills the background gray where needed

## Scrolling

- **autoscroll:**(NXEvent \*)*theEvent*
- **constrainScroll:**(NXPoint \*)*newOrigin*
- **rawScroll:**(const NXPoint \*)*newOrigin*
- **setCopyOnScroll:**(BOOL)*flag*
- **setDisplayOnScroll:**(BOOL)*flag*

Scrolls in response to mouse-dragged events  
Prevents scrolling to an undesirable position  
Lowest-level unconstrained scrolling routine  
Sets how the visible areas are redrawn  
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

- **initFrame:(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** Returns the Control’s selected Cell
- **(int)selectedTag** Returns the tag of the Control’s selected Cell

## Setting the Control’s Value

- **setDoubleValue:(double)aDouble** Sets the Control’s value to *aDouble*
- **(double)doubleValue** Returns the Control’s value as a **double**
- **setFloatValue:(float)aFloat** Sets the Control’s value to *aFloat*
- **(float)floatValue** Returns the Control’s value as a **float**
- **setIntValue:(int)anInt** Sets the Control’s value to *anInt*
- **(int)intValue** Returns the Control’s value as an **int**
- **setValue:(const char \*)aString** Sets the Control’s value to *aString*
- **setStringValueNoCopy:(const char \*)aString** Sets the Control’s value to *aString*
- **setStringValueNoCopy:(char \*)aString shouldFree:(BOOL)flag** Sets the Control’s value to *aString*
- **(const char \*)stringValue** Returns the Control’s value as a string

## Interacting with Other Controls

- **takeDoubleValueFrom:sender** Sets the Control’s value to *sender*’s **doubleValue**
- **takeFloatValueFrom:sender** Sets the Control’s value to *sender*’s **floatValue**
- **takeIntValueFrom:sender** Sets the Control’s value to *sender*’s **intValue**
- **takeStringValueFrom:sender** Sets the Control’s value to *sender*’s **stringValue**

## Formatting Text

- **setAlignment:(int)mode** Sets the alignment of text in the Control to *mode*
- **(int)alignment** Returns the alignment of text in the Control
- **setFont:fontObject** Sets the Font used to draw text in the Control to *fontObject*
- **font** Returns the Font used to draw text in the Control
- **setFloatingPointFormat:(BOOL)autoRange left:(unsigned)leftDigits right:(unsigned)rightDigits** Sets the display format for floating point values

## Managing the Field Editor

- **abortEditing** Aborts editing of text displayed by the Control
- **currentEditor** Returns the object used to edit text in the Control
- **validateEditing** 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** Recalculates internal size information
- **sizeTo:(NXCoord)width :(NXCoord)height** Resizes the Control to *width* and *height*
- **sizeToFit** Resizes the Control to fit its Cell

## Displaying the Control and Cell

- **drawCell:*aCell*** Redraws *aCell* if it's the Control's Cell
- **drawCellInside:*aCell*** Redraws *aCell*'s inside if it's the Control's Cell
- **drawSelf:(const NXRect \*)*rects* :(int)*rectCount*** Draws the Control
- **selectCell:*aCell*** Selects *aCell* if it's the Control's cell
- **update** Redisplays the Control or marks it for later redisplay
- **updateCell:*aCell*** Redisplays *aCell* or marks it for redisplay
- **updateCellInside:*aCell*** Redisplays the inside of *aCell* or marks it for redisplay

## Target and Action

- **setAction:(SEL)*aSelector*** Sets the Control's action method to *aSelector*
- **(SEL)action** Returns the Control's action method
- **setTarget:*anObject*** Sets the Control's target object to *anObject*
- **target** Returns the Control's target object
- **setContinuous:(BOOL)*flag*** Sets whether the Control continuously sends its action
- **(BOOL)isContinuous** Returns whether the Control continuously sends its action
- **sendAction:(SEL)*theAction* to:*theTarget*** Has the Application object send *theAction* to *theTarget*
- **(int)sendActionOn:(int)*mask*** Determines when the action is sent while tracking

## Assigning a Tag

- **setTag:(int)*anInt*** Sets the Control's tag to *anInt*
- **(int)tag** Returns the Control's tag

## Tracking the Mouse

- **ignoreMultiClick:(BOOL)*flag*** Sets whether multiple clicks are ignored
- **mouseDown:(NXEvent \*)*theEvent*** Handles a mouse-down event in the Control
- **(int)mouseDownFlags** 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 *) <b>displayName</b>	Returns the full name of the font
– (const float *) <b>familyName</b>	Returns the name of the font's family
– (const char *) <b>name</b>	Returns the name of the font
– (int) <b>fontNum</b>	Returns the Window Server's font number
– (float) <b>getWidthOf:(const char *)string</b>	Returns the width of <i>string</i> in this font
– (BOOL) <b>hasMatrix</b>	Returns whether font differs from identity matrix
– (const float *) <b>matrix</b>	Returns a pointer to the font matrix
– (NXFontMetrics *) <b>metrics</b>	Returns pointer to a record of font information
– (float) <b>pointSize</b>	Returns the size of the font in points
– (NXFontMetrics *) <b>readMetrics:(int)flags</b>	Reads <i>flags</i> information into the font record
– <b>screenFont</b>	Returns the screen font for this font
– (int) <b>style</b>	Returns the font style

## Setting the Font

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

## Archiving

– <b>awake</b>	Reinitializes the Font object
– <b>finishUnarchiving</b>	Checks whether the Font object already exists
– <b>read:(NXTypedStream *)stream</b>	Reads the Font object from the typed stream
– <b>write:(NXTypedStream *)stream</b>	Writes the Font object to the typed stream

---

## FontManager

Inherits From: Object

### Creating a FontManager

+ <b>new</b>	Returns the application-wide FontManager
--------------	--

## Converting Fonts

– <b>convertFont:</b> <i>fontObj</i>	Converts the font in response to <b>changeFont</b> :
– <b>convertWeight:</b> (BOOL) <i>upFlag</i> <b>of:</b> <i>fontObj</i>	Raises or lowers the weight of the font
– <b>convert:</b> <i>fontObj</i> <b>toFace:</b> (const char *) <i>typeface</i>	Converts the font to the specified typeface
– <b>convert:</b> <i>fontObj</i> <b>toFamily:</b> (const char *) <i>family</i>	Converts the font to the specified family
– <b>convert:</b> <i>fontObj</i> <b>toSize:</b> (float) <i>size</i>	Converts the font to the specified point size
– <b>convert:</b> <i>fontObj</i> <b>toHaveTrait:</b> (NXFontTraitMask) <i>trait</i>	Converts the font to have the specified trait
– <b>convert:</b> <i>fontObj</i> <b>toNotHaveTrait:</b> (NXFontTraitMask) <i>trait</i>	Converts the font to remove the specified trait
– <b>findFont:</b> (const char *) <i>family</i> <b>traits:</b> (NXFontTraitMask) <i>traits</i> <b>weight:</b> (int) <i>weight</i> <b>size:</b> (float) <i>size</i>	Tries to find a font that matches the specified characteristics
– <b>getFamily:</b> (const char **) <i>family</i> <b>traits:</b> (NXFontTraitMask *) <i>traits</i> <b>weight:</b> (int *) <i>weight</i> <b>size:</b> (float*) <i>size</i> <b>offFont:</b> <i>fontObj</i>	Provides the characteristics of the given <i>fontObj</i>

## Setting Parameters

– <b>setAction:</b> (SEL) <i>aSelector</i>	Sets the action sent by the FontManager
+ <b>setFontPanelFactory:</b> <i>classId</i>	Sets the class used to create the Font panel
+ <b>setFontManagerFactory:</b> <i>classId</i>	Sets the class used to create the font manager
– <b>setSelFont:</b> <i>fontObj</i> <b>isMultiple:</b> (BOOL) <i>flag</i>	Notifies FontManager of selection's current font
– <b>setEnabled:</b> (BOOL) <i>flag</i>	Enables and disables the Font panel and menu

## Querying Parameters

– (SEL) <b>action</b>	Gets the action sent by the FontManager
– (char **) <b>availableFonts</b>	Provides a list of all available fonts
– <b>getFontMenu:</b> (BOOL) <i>create</i>	Returns the Font menu
– <b>getFontPanel:</b> (BOOL) <i>create</i>	Returns the Font panel
– (BOOL) <b>isMultiple</b>	Returns whether selection contains multiple fonts
– <b>selFont</b>	Returns the first font in the current selection
– (BOOL) <b>isEnabled</b>	Returns whether the Font panel and menu are enabled

## Target and Action Methods

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

## Assigning a Delegate

– <b>setDelegate:<i>anObject</i></b>	Sets the FontManager's delegate
– <b>delegate:</b>	Returns the FontManager's delegate

## Archiving the FontManager

– <b>finishUnarchiving</b>	Finishes unarchiving by creating the FontManager
----------------------------	--

---

# FontPanel

**Inherits From:** Panel : Window : Responder : Object

## Creating a FontPanel

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

## Setting the Font

– <b>setPanelFont:<i>fontObj</i> isMultiple:(BOOL)<i>flag</i></b>	Sets the Font panel's current font
– <b>panelConvertFont:<i>fontObj</i></b>	Converts <i>fontObj</i> to the user's choice from the panel

## Configuring the FontPanel

– accessoryView	Returns the application-customized view
– setAccessoryView: <i>aView</i>	Adds application-customized View to the FontPanel
– setEnabled:(BOOL) <i>flag</i>	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: <i>textObject</i> <i>endChar</i> :(unsigned short) <i>endChar</i>	Detects completion of size field editing
– textDidGetKeys: <i>textObject</i> isEmpty:(BOOL) <i>flag</i>	Detects empty size field

## Displaying the FontPanel

– orderWindow:(int) <i>place</i> relativeTo:(int) <i>otherWin</i>	Repositions panel and updates it if necessary
---	---

## Resizing the FontPanel

– windowWillResize: <i>sender</i> <i>toSize</i> :(NXSize *) <i>frameSize</i>	Constrains FontPanel resizing
---	-------------------------------

---

## Form

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

### Setting Form's Cell Class

+ setCellClass: <i>classId</i>	Sets the subclass of Cell used by Form
--------------------------------	--

### Initializing a Form

– initFrame:(const NXRect *) <i>frameRect</i>	Initializes a new Form in <i>frameRect</i>
---	--

## Laying Out the Form

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

## Assigning a Tag

<code>- setTag:(int)<i>aInt</i> at:(int)<i>index</i></code>	Sets the tag of the entry at <i>index</i> to <i>aInt</i>
---	--

## Finding Indices

<code>- (int)findIndexWithTag:(int)<i>aTag</i></code>	Returns the index for the entry with tag <i>aTag</i>
<code>- (int)selectedIndex</code>	Returns the index of the currently selected entry

## Modifying Graphic Attributes

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

## Setting Item Titles

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

## Setting Item Values

- |  |   |
|--|---|
| – <b>setDoubleValue:(double)</b> <i>aDouble</i> <b>at:(int)</b> <i>index</i> | Sets the value of the entry at <i>index</i> to <i>aDouble</i>     |
| – <b>(double)doubleValueAt:(int)</b> <i>index</i>                            | Returns the value of the entry at <i>index</i> as a <b>double</b> |
| – <b>setFloatValue:(float)</b> <i>aFloat</i> <b>at:(int)</b> <i>index</i>    | Sets the value of the entry at <i>index</i> to <i>aFloat</i>      |
| – <b>(float)floatValueAt:(int)</b> <i>index</i>                              | Returns the value of the entry at <i>index</i> as a <b>float</b>  |
| – <b>setIntValue:(int)</b> <i>aInt</i> <b>at:(int)</b> <i>index</i>          | Sets the value of the entry at <i>index</i> to a <i>aInt</i>      |
| – <b>(int)intValueAt:(int)</b> <i>index</i>                                  | Returns the value of the entry at <i>index</i> as an <b>int</b>   |
| – <b>setValue:(const char *)</b> <i>aString</i> <b>at:(int)</b> <i>index</i> | Sets the value of the entry at <i>index</i> to a <i>aString</i>   |
| – <b>(const char *)stringValueAt:(int)</b> <i>index</i>                      | Returns the value of the entry at <i>index</i> as a string        |

## Editing Text

- |  |   |
|--|---|
| – <b>selectTextAt:(int)</b> <i>index</i> | Selects the text in the entry at <i>index</i> |
|--|---|

## Resizing the Form

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

## Displaying

- |  |   |
|--|---|
| – <b>drawCellAt:(int)</b> <i>index</i> | Displays the Cell at the specified <i>index</i> |
|--|---|

## Target and Action

- |  |   |
|--|---|
| – <b>setAction:(SEL)</b> <i>aSelector</i> <b>at:(int)</b> <i>index</i> | Sets the action method of the entry at <i>index</i> to <i>aSelector</i> |
| – <b>setTarget:</b> <i>anObject</i> <b>at:(int)</b> <i>index</i>       | Sets the target object of the entry at <i>index</i> to <i>anObject</i>  |

# FormCell

Inherits From: ActionCell : Cell : Object

## Initializing, Copying, and Freeing a FormCell

- **init**
- **initTextCell:(const char \*)*aString***
- **copyFromZone:(NXZone \*)*zone***
- **free**

Initializes a new FormCell with “Field” as its title  
Initializes a new FormCell with *aString* as its title  
Returns a copy of the FormCell allocated from *zone*  
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***
- **(const char \*)title**
- **setTitleFont:*fontObject***
- **titleFont**
- **setTitleAlignment:(int)*mode***
- **(int)titleAlignment**
- **setTitleWidth:(NXCoord)*width***
- **(NXCoord)titleWidth:(const NXSize \*)*aSize***
- **(NXCoord)titleWidth**

Sets the FormCell’s title to *aString*  
Returns the FormCell’s title  
Sets the Font used to draw the title  
Returns the Font used to draw the title  
Sets the alignment of the title  
Returns the alignment of the title  
Sets the width of the FormCell’s title field to *width*  
Returns the width of the title, constrained to *aSize*  
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

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

## Providing for Program Control

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

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

## Receiving Remote Messages

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

## Assigning a Delegate

– setDelegate: <i>anObject</i>	Makes <i>anObject</i> the Listener's delegate
– delegate	Returns the Listener's delegate
– setServicesDelegate: <i>anObject</i>	Makes <i>anObject</i> the receiver of service requests
– servicesDelegate	Returns the object that receives service requests

## Archiving

– read:(NXTypedStream *) <i>stream</i>	Reads the Listener from <i>stream</i>
– write:(NXTypedStream *) <i>stream</i>	Writes the Listener to <i>stream</i>

## Matrix

**Inherits From:** Control : View : Responder : Object

### Initializing the Matrix Class

+ initialize	Initializes the Matrix class
+ setCellClass: <i>classId</i>	Sets the default class used to make Cells

## Initializing and Freeing a Matrix

- **initFrame:(const NXRect \*)frameRect**
- **initFrame:(const NXRect \*)frameRect mode:(int)aMode**
- cellClass:className**
- numRows:(int)numRows**
- numCols:(int)numCols**
- **initFrame:(const NXRect \*)frameRect mode:(int)aMode prototype:aCell**
- numRows:(int)numRows**
- numCols:(int)numCols**
- **free**

Initializes a new Matrix object in *frameRect*

Initializes a new Matrix object in *frameRect*,  
with *aMode* as the selection mode,  
*className* as the class used to make new Cells,  
and having *numRows* rows  
and *numCols* columns

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

Deallocates the Matrix and all its Cells

## Setting the Selection Mode

- **setMode:(int)aMode**
- **(int)mode**

Sets the selection mode of the Matrix

Returns the selection mode of the Matrix

## Configuring the Matrix

- **setEnabled:(BOOL)flag**
- **setEmptySelectionEnabled:(BOOL)flag**
- **(BOOL)isEmptySelectionEnabled**
- **setSelectionByRect:(BOOL)flag**
- **(BOOL)isSelectionByRect**

Sets whether the Matrix reacts to events

Sets whether there may be no Cells selected

Returns whether there may be no Cells selected

Sets whether a user can drag a rectangular selection  
(the default is YES)

Returns whether a user can drag a rectangular selection

## Setting the Cell class

- **setCellClass:className**
- **setPrototype:aCell**
- **prototype**

Sets the subclass of Cell used to make new Cells

Sets the prototype Cell copied to make new Cells

Returns the prototype Cell copied to make new Cells

## Laying Out the Matrix

- **addCol**
- **addRow**
- **insertColAt:(int)col**
- **insertRowAt:(int)row**

Adds a new column of Cells to the bottom of the Matrix

Adds a new row of Cells to the right of the Matrix

Inserts a new column of Cells at *col*, creating as many as  
needed to make the Matrix *col* columns wide

Inserts a new row of Cells at *row*, creating as many as  
needed to make the Matrix *row* rows wide

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

## Finding Matrix Coordinates

<b>– getRow:(int *)<i>row</i> andCol:(int *)<i>col</i> ofCell:<i>aCell</i></b>	Gets the <i>row</i> and <i>col</i> position of <i>aCell</i>
<b>– getRow:(int *)<i>row</i> andCol:(int *)<i>col</i> forPoint:(const NXPoint *)<i>aPoint</i></b>	Gets the <i>row</i> and <i>col</i> position corresponding to <i>aPoint</i> , and returns the Cell at that point

## Modifying Individual Cells

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

## Selecting Cells

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

## Finding Cells

– <b>findCellWithTag:(int)<i>aInt</i></b>	Returns the Cell with <i>aInt</i> as its tag
– <b>cellAt:(int)<i>row</i> :(int)<i>col</i></b>	Returns the Cell at row <i>row</i> and column <i>col</i>
– <b>cellList</b>	Returns the Matrix's List of Cells

## Modifying Graphic Attributes

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

– <b>setAutoScroll:(BOOL)</b> <i>flag</i>	Sets whether the Matrix automatically scrolls when dragged in
– <b>setScrollable:(BOOL)</b> <i>flag</i>	Makes all the Cells scrollable
– <b>scrollCellToVisible:(int)</b> <i>row</i> <b>:</b> (int) <i>col</i>	Scrolls Matrix so the Cell at <i>row</i> and <i>col</i> is visible

## Displaying

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

## Target and Action

– <b>setTarget:</b> <i>anObject</i>	Sets the target of the Matrix to <i>anObject</i>
– <b>target</b>	Returns the target of the Matrix
– <b>setAction:(SEL)</b> <i>aSelector</i>	Sets the action of the Matrix to <i>aSelector</i>
– (SEL) <b>action</b>	Returns the action of the Matrix
– <b>setDoubleAction:(SEL)</b> <i>aSelector</i>	Sets the action method used on double-clicks to <i>aSelector</i>
– (SEL) <b>doubleAction</b>	Returns the action method for double clicks
– <b>setErrorAction:(SEL)</b> <i>aSelector</i>	Sets the action method for editing errors to <i>aSelector</i>
– (SEL) <b>errorAction</b>	Returns the action method for editing errors
– <b>setTarget:</b> <i>anObject</i> <b>at:(int)</b> <i>row</i> <b>:</b> (int) <i>col</i>	Assigns <i>anObject</i> as the target of the Cell at <i>row</i> , <i>col</i>
– <b>setAction:(SEL)</b> <i>aSelector</i> <b>at:(int)</b> <i>row</i> <b>:</b> (int) <i>col</i>	Assigns <i>aSelector</i> as the action method of the Cell at <i>row</i> , <i>col</i>
– <b>sendAction</b>	Sends the selected Cell's action, or the Matrix's action if the Cell doesn't have one
– <b>sendAction:(SEL)</b> <i>theAction</i> <b>to:</b> <i>theTarget</i>	Has the Application object send <i>theAction</i> to <i>anObject</i>
– <b>sendAction:(SEL)</b> <i>aSelector</i> <b>to:</b> <i>anObject</i> <b>forAllCells:(BOOL)</b> <i>flag</i>	Sends <i>aSelector</i> to <i>anObject</i> , for all Cells if <i>flag</i> is YES
– <b>sendDoubleAction</b>	Sends the action corresponding to a double-click
– <b>setReaction:(BOOL)</b> <i>flag</i>	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 *) <i>theEvent</i>	Responds to a mouse-down event
– (int)mouseDownFlags	Returns the event flags in effect at start of tracking
– (BOOL)performKeyEquivalent:(NXEvent *) <i>theEvent</i>	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 *) <i>stream</i>	Reads the Matrix from <i>stream</i>
– write:(NXTypedStream *) <i>stream</i>	Writes the Matrix to <i>stream</i>

---

## **Menu**

**Inherits From:** Panel : Window : Responder : Object

### **Creating a Menu Zone**

+ setMenuZone:(NXZone *) <i>zone</i>	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”
– initTitle:(const char *) <i>aTitle</i>	Initializes a new Menu with <i>aTitle</i> as its title

## Setting Up the Menu Commands

- **addItem:**(const char \*)*aString*  
    action:(SEL)*aSelector*  
    keyEquivalent:(unsigned short)*charCode*
- **setItemList:***aMatrix*
- **itemList**

Adds a new item to the end of the Menu

Replaces the current Matrix of items with *aMatrix*  
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*
- **submenuAction:***sender*

Makes *aMenu* a submenu controlled by *aCell*  
Activates a submenu attached to *sender*'s Menu

## Managing Menu Windows

- **moveTopLeftTo:**(NXCoord)*x* :(NXCoord)*y*
- **windowMoved:**(NXEvent \*)*theEvent*
- **getLocation:**(NXPoint \*)*theLocation*  
    **forSubmenu:***a\_submenu*
- **sizeToFit**
- **close**

Moves the Menu's top left corner to *x, y*  
Handles a submenu being torn off its supermenu  
Determines where to display an attached submenu  
    when it's brought up  
Resizes the Menu to exactly fit the command items  
Removes the Menu (and any submenus) from the screen

## Displaying the Menu

- **display**
- **setAutoupdate:**(BOOL)*flag*
- **update**

Displays the Menu, resizing if needed  
Sets whether Menu reacts to **update** messages  
Updates each MenuCell item

## Handling Events

- **mouseDown:**(NXEvent \*)*theEvent*
- **rightMouseDown:**(NXEvent \*)*theEvent*

Tracks the cursor in the Menu and submenus  
Pops the main menu up under the cursor

## Archiving

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

Reads the Menu from *stream*  
Writes the Menu to *stream*  
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
- **initTextCell:(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

- <b>initFromSection:</b> (const char *) <i>name</i>	Initializes the new object from TIFF data in the section
- <b>initFromFile:</b> (const char *) <i>filename</i>	Initializes the new object from TIFF data in <i>filename</i>
- <b>initFromStream:</b> (NXStream *) <i>stream</i>	Initializes the new object from TIFF data in <i>stream</i>
- <b>initData:</b> (unsigned char *) <i>data</i> <b>fromRect:</b> (const NXRect *) <i>rect</i>	Initializes the new object using data read from an image
- <b>initData:</b> (unsigned char *) <i>data</i> <b>pixelsWide:</b> (int) <i>width</i> <b>pixelsHigh:</b> (int) <i>height</i> <b>bitsPerSample:</b> (int) <i>bps</i> <b>samplesPerPixel:</b> (int) <i>spp</i> <b>hasAlpha:</b> (BOOL) <i>alpha</i> <b>isPlanar:</b> (BOOL) <i>config</i> <b>colorSpace:</b> (NXColorSpace) <i>space</i> <b>bytesPerRow:</b> (int) <i>rowBytes</i> <b>bitsPerPixel:</b> (int) <i>pixelBits</i>	Initializes the new object from raw bitmap data
- <b>initDataPlanes:</b> (unsigned char **) <i>planes</i> <b>pixelsWide:</b> (int) <i>width</i> <b>pixelsHigh:</b> (int) <i>height</i> <b>bitsPerSample:</b> (int) <i>bps</i> <b>samplesPerPixel:</b> (int) <i>spp</i> <b>hasAlpha:</b> (BOOL) <i>alpha</i> <b>isPlanar:</b> (BOOL) <i>config</i> <b>colorSpace:</b> (NXColorSpace) <i>space</i> <b>bytesPerRow:</b> (int) <i>rowBytes</i> <b>bitsPerPixel:</b> (int) <i>pixelBits</i>	Initializes the new object from raw bitmap data in the <i>planes</i> data buffers

## Creating a List of NXBitmapImageReps

+ (List *) <b>newListFromSection:</b> (const char *) <i>name</i>	Returns a List of NXBitmapImageReps from <i>name</i> data
+ (List *) <b>newListFromSection:</b> (const char *) <i>name</i> <b>zone:</b> (NXZone *) <i>aZone</i>	Returns a List of NXBitmapImageReps from <i>name</i> data
+ (List *) <b>newListFromFile:</b> (const char *) <i>filename</i>	Returns a List of NXBitmapImageReps from <i>filename</i>
+ (List *) <b>newListFromFile:</b> (const char *) <i>filename</i> <b>zone:</b> (NXZone *) <i>aZone</i>	Returns a List of NXBitmapImageReps from <i>filename</i>

+ (List *) newListFromStream:(NXStream *) <i>stream</i>	Returns a List of NXBitmapImageReps from <i>stream</i> data
+ (List *) newListFromStream:(NXStream *) <i>stream</i> zone:(NXZone *) <i>aZone</i>	Returns a List of NXBitmapImageReps from <i>stream</i> data

## Reading Information from a Rendered Image

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

## Copying and Freeing an NXBitmapImageRep

- copyFromZone:(NXZone *) <i>zone</i>	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 *) <i>rect</i>	Modifies coordinates so image is drawn in <i>rect</i> 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

- **initFrame:(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

<b>– useScrollBars:(BOOL)<i>flag</i></b>	Sets whether Scrollers are used to scroll columns
<b>– useScrollButtons:(BOOL)<i>flag</i></b>	Sets whether buttons are used to scroll columns
<b>– setHorizontalScrollButtonsEnabled:(BOOL)<i>flag</i></b>	Sets whether buttons are used to scroll horizontally
<b>– (BOOL)areHorizontalScrollButtonsEnabled</b>	Returns whether buttons are used to scroll horizontally
<b>– setHorizontalScrollerEnabled:(BOOL)<i>flag</i></b>	Sets whether Scroller is used to scroll horizontally
<b>– (BOOL)isHorizontalScrollerEnabled</b>	Returns whether Scroller is used to scroll horizontally

## Setting the NXBrowser's Appearance

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

## Manipulating Columns

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

## Manipulating Column Titles

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

## Scrolling an NXBrowser

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

## Event Handling

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

## Getting Matrices and Cells

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

## Getting Column Frames

- (NXRect \*)**getFrame:(NXRect \*)theRect  
ofColumn:(int)column**
- (NXRect \*)**getFrame:(NXRect \*)theRect  
ofInsideOfColumn:(int)column**

Returns the rectangle containing the column at index *column*

Returns the rectangle containing the column at index *column*, not including borders

## Manipulating Paths

- **setPathSeparator:(unsigned short)charCode**
- **setPath:(const char \*)path**
- (char \*)**getPath:(char \*)thePath  
toColumn:(int)column**

Sets the path separator to *charCode*

Parses *path* and selects corresponding items in columns

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**
- **sizeToFit**

Resizes the NXBrowser to *width* and *height*

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**
- **browserDidScroll:sender**
- (int)**browser:sender  
fillMatrix:matrix  
inColumn:(int)column**
- (int)**browser:sender  
getNumRowsInColumn:(int)column**
- **browser:sender  
loadCell:cell  
atRow:(int)row  
inColumn:(int)column**
- (BOOL)**browser:sender  
selectCell:(const char \*)title  
inColumn:(int)column**

Returns whether the contents of the column are valid

Notifies the delegate when the NXBrowser has scrolled

Returns the number of rows in a column and loads NXBrowserCells in *matrix*

Returns the number of rows of data in the column at index *column*

Requests the delegate to load Cell at *row* in the column at index *column*

Requests the delegate to select the Cell with title *title* in the column at index *column*

---

– (const char *) <b>browser:sender</b>	Queries the delegate for the title to display above the column at index <i>column</i>
– <b>browserWillScroll:sender</b>	Notifies the delegate when the NXBrowser will scroll

## NXBrowserCell

**Inherits From:** Cell : Object

### Initializing a NXBrowserCell

– <b>init</b>	Initializes a new NXBrowserCell with “BrowserItem” as its title
– <b>initTextCell:(const char *)<i>aString</i></b>	Initializes a new NXBrowserCell with <i>aString</i> as its title

### Determining Component Sizes

– <b>calcCellSize:(NXSize *)<i>theSize</i> inRect:(const NXRect *)<i>aRect</i></b>	Calculates the size of the NXBrowserCell within <i>aRect</i>
--	--

### Accessing Graphic Attributes

– <b>(BOOL)isOpaque</b>	Returns YES, since an NXBrowserCell is opaque
+ <b>branchIcon</b>	Returns the NXImage for branch NXBrowserCells
+ <b>branchIconH</b>	Returns the NXImage for highlighted branches

### Displaying

– <b>drawInside:(const NXRect *)<i>cellFrame</i> inView:<i>aView</i></b>	Draws the inside of the NXBrowserCell in <i>aView</i>
– <b>drawSelf:(const NXRect *)<i>cellFrame</i> inView:<i>aView</i></b>	Draws the entire NXBrowserCell in <i>aView</i>
– <b>highlight:(const NXRect *)<i>cellFrame</i> inView:<i>aView</i> lit:(BOOL)<i>lit</i></b>	If <i>lit</i> is YES, highlights the NXBrowserCell in <i>aView</i>

### Placing in Browser Hierarchy

– <b>setLeaf:(BOOL)<i>flag</i></b>	Sets whether the NXBrowserCell is a leaf or a branch
– <b>(BOOL)isLeaf</b>	Returns whether the NXBrowserCell is a leaf or a branch

## Determining Loaded Status

- `setLoaded:(BOOL)flag`
- `(BOOL)isLoaded`

Sets whether the NXBrowserCell is loaded and displayable  
Returns whether the NXBrowserCell is loaded

## Setting State

- `set`
- `reset`

Highlights the NXBrowserCell and sets its state to 1  
Unhighlights the NXBrowserCell and sets its state to 0

---

# NXCachedImageRep

**Inherits From:**      NXImageRep : Object

## Initializing a New NXCachedImageRep

- `initFromWindow:(Window *)aWindow  
rect:(const NXRect *)aRect`
- `copyFromZone:(NXZone *)theZone`

Initializes the new NXCachedImageRep for an image to  
be drawn in *aWindow*

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`
- `write:(NXTypedStream *)stream`

Reads the NXCachedImageRep from *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> <i>style</i> :(int) <i>aStyle</i> <i>backing</i> :(int) <i>bufferingType</i> <i>buttonMask</i> :(int) <i>mask</i> <i>defer</i> :(BOOL) <i>flag</i>	Returns the shared NXColorPanel
+ newContent:(const NXRect *) <i>contentRect</i> <i>style</i> :(int) <i>aStyle</i> <i>backing</i> :(int) <i>bufferingType</i> <i>buttonMask</i> :(int) <i>mask</i> <i>defer</i> :(BOOL) <i>flag</i> <i>colorMask</i> :(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

– **initFromPickerMask:(int)theMask  
withColorPanel:thePanel**

Initializes the receiver for the specified mask and color panel

### Button Images

– **provideNewButtonImage**  
– **insertNewButtonImage:newImage  
in:newButtonCell**

Returns a new button image for the color picker  
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**  
– **detachColorList:colorList**

Override to attach a color list to a color picker  
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

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

### Event Handling

– **(BOOL)acceptsFirstMouse**  
– **mouseDown:(NXEvent \*)*theEvent***

Returns YES

Responds to mouse down in the NXColorWell

### Drawing

– **drawSelf:(const NXRect \*)*rects*  
          :(int)*rectCount***  
– **drawWellInside:(const NXRect \*)*insideRect***

Draws the entire NXColorWell, including borders

Draws the colored area inside the NXColorWell, without  
drawing borders

### Activating

– **deactivate**  
+ **deactivateAllWells**  
– **activate:(int)*exclusive***  
– **(BOOL)isActive**  
– **setEnabled:(BOOL)*enabled***

Deactivates and returns the NXColorWell

Deactivates all currently active NXColorWells

Activates and returns the NXColorWell

Returns YES if the NXColorWell is active

Enables the NXColorWell

### Managing Color

– **activeWellsTakeColorFrom:*sender***  
– **activeWellsTakeColorFrom:*sender*  
          continuous:(BOOL)*continuously***  
– **(NXColor)*color***  
– **takeColorFrom:*sender***  
– **acceptColor:(NXColor)*color*  
          atPoint:(NXPoint \*)*aPoint***  
– **setColor:(NXColor)*color***  
– **updateCustomColorList**

Changes color of all active wells to that of *sender*

Continuously changes color of all active, continuous  
wells to that of *sender*

Returns the color of the NXColorPanel

Changes color of the well to that of *sender*

Changes color of the well to *color* when *aPoint* is a  
point in the bounds of the NXColorWell

Sets the color of the well to *color*

Saves the current color list in NX\_COLORLISTMODE

## Target and Action

– (SEL)action	Returns the NXColorWell’s action message
– setAction:(SEL) <i>aSelector</i>	Sets the NXColorWell’s action message
– setTarget: <i>anObject</i>	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
– initFromImage: <i>image</i>	Initializes a new NXCursor object with <i>image</i>

### Defining the Cursor

– setImage: <i>newImage</i>	Sets the NXImage object that supplies the cursor image
– image	Returns the NXImage object that has the cursor image
– setHotSpot:(const NXPoint *) <i>spot</i>	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) <i>flag</i>	Determines whether <b>mouseEntered:</b> sets cursor
– setOnMouseExited:(BOOL) <i>flag</i>	Determines whether <b>mouseExited:</b> sets cursor
– mouseEntered:(NXEvent *) <i>theEvent</i>	Responds to a mouse-entered event
– mouseExited:(NXEvent *) <i>theEvent</i>	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 stream *stream*

---

## NXCustomImageRep

**Inherits From:**      NXImageRep : Object

### Initializing a New NXCustomImageRep

- **initDrawMethod:(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

- **initFromSection:(const char \*)name**
- **initFromFile:(const char \*)filename**
- **initFromStream:(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 NXEPSErrorRep**

+ (List *)newListFromSection:(const char *)name	Returns a List of NXEPSErrorRep from EPS in <i>name</i>
+ (List *)newListFromSection:(const char *)name zone:(NXZone *)aZone	Returns a List of NXEPSErrorRep from EPS in <i>name</i>
+ (List *)newListFromFile:(const char *)filename	Returns a List of NXEPSErrorRep from <i>filename</i> data
+ (List *)newListFromFile:(const char *)filename zone:(NXZone *)aZone	Returns a List of NXEPSErrorRep from <i>filename</i> data
+ (List *)newListFromStream:(NXStream *)stream	Returns a List of NXEPSErrorRep from EPS in <i>stream</i>
+ (List *)newListFromStream:(NXStream *)stream zone:(NXZone *)aZone	Returns a List of NXEPSErrorRep from EPS in <i>stream</i>

## **Copying and Freeing an NXEPSErrorRep**

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

## **Getting the Rectangle that Bounds the Image**

- getBoundingBox:(NXRect *)rect	Copies the EPS bounding box into the <i>rect</i> 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 <i>rect</i> rectangle

## **Archiving**

- read:(NXTypedStream *)stream	Reads the NXEPSErrorRep from <i>stream</i>
- write:(NXTypedStream *)stream	Writes the NXEPSErrorRep to <i>stream</i>

---

## NXHelpPanel

**Inherits From:** Panel : Window : Responder : Object

### Initializing and Freeing

- + new
- + newForDirectory:(const char \*)*helpDirectory*
- addSupplement:(const char \*)*helpDirectory*  
    inPath:(const char \*)*supplementPath*
- free

Creates, if necessary, and returns the NXHelpPanel object

Creates, if necessary, and returns the NXHelpPanel object

Adds supplemental help to the text displayed in the panel

Frees the NXHelpPanel and its storage

### Attaching Help to Objects

- + attachHelpFile:(const char \*)*filename*  
    markerName:(const char \*)*markerName*  
    to:*anObject*
- + detachHelpFrom:*anObject*

Associates the help file at *markerName* with *anObject*

Removes any help information associated with *anObject*

### Setting Click-for-Help

- + (BOOL)isClickForHelpEnabled
- + setClickForHelpEnabled:(BOOL)*enabled*

Returns whether the click-for-help feature is enabled

Sets whether the click-for-help feature is enabled

### Printing

- print:*sender*
- printPanel:*sender*

Prints the currently displayed help text

Prints the currently displayed help text

### Querying

- (NXAtom)helpDirectory
- (NXAtom) helpFile

Returns the absolute path of the help directory

Returns the path of the currently loaded help file

### Showing Help

- showFile:(const char \*)*filename*  
    atMarker:(const char \*)*markerName*
- (BOOL)showHelpAttachedTo:*anObject*

Causes the Help panel to display the help contained in  
*filename* at *markerName*

Causes the Help panel to display help attached to *anObject*

# NXImage

Inherits From: Object

## Initializing a New NXImage Instance

- **init**
- **initSize:(const NXSize \*)*aSize***
- **initFromSection:(const char \*)*name***
- **initFromFile:(const char \*)*filename***
- **initFromPasteboard:(Pasteboard \*)*pasteboard***
- **initFromStream:(NXStream \*)*stream***
- **initFromImage:(NXImage \*)*image*  
rect:(const NXRect \*)*rect***
- **copyFromZone:(NXZone \*)*zone***

Initializes the new NXImage without setting its size  
Initializes the new NXImage to the specified size  
Initializes the new object from the data in *name* section  
Initializes the new NXImage from the data in *filename*  
Initializes the new NXImage from the data in *pasteboard*  
Initializes the new NXImage from the data in *stream*  
Initializes the new NXImage *to be a subimage of image*  
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***
- **getSize:(NXSize \*)*theSize***

Sets the size of the image in base coordinates  
Provides the size of the image

## Referring to Images by Name

- **(BOOL)setName:(const char \*)*string***
- **(const char \*)*name***
- + **findImageNamed:(const char \*)*name***

Assigns *string* as the name of the NXImage object  
Returns the name of the NXImage object  
Returns the NXImage object with *name*

## Specifying the Image

- **(BOOL)useDrawMethod:(SEL)*aSelector*  
inObject:*anObject***
- **(BOOL)useFromSection:(const char \*)*name***
- **(BOOL)useFromFile:(const char \*)*filename***
- **(BOOL)useRepresentation:(NXImageRep \*)*imageRep***

Creates a representation that will use a delegated method  
to draw the image  
Creates representations for the data in the *name* section  
Creates representations for the data in *filename* file  
Adds *imageRep* to the List of representations

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

## Using the Image

– composite:(int) <i>op</i> <i>toPoint:</i> (const NXPoint *) <i>aPoint</i>	Composites the image to <i>aPoint</i>
– composite:(int) <i>op</i> <i>fromRect:</i> (const NXRect *) <i>aRect</i> <i>toPoint:</i> (const NXPoint *) <i>aPoint</i>	Composites the <i>aRect</i> portion of the image to <i>aPoint</i>
– dissolve:(float) <i>delta</i> <i>toPoint:</i> (const NXPoint *) <i>aPoint</i>	Composites the image using the <b>dissolve</b> operator
– dissolve:(float) <i>delta</i> <i>fromRect:</i> (const NXRect *) <i>aRect</i> <i>toPoint:</i> (const NXPoint *) <i>aPoint</i>	Composites the image using the <b>dissolve</b> operator

## Choosing Which Image Representation to Use

– setColorMatchPreferred:(BOOL) <i>flag</i>	Determines whether color matches are preferred
– (BOOL)isColorMatchPreferred	Returns whether color matches are preferred
– setEPSUsedOnResolutionMismatch:(BOOL) <i>flag</i>	Sets whether to use EPS representations on mismatch
– (BOOL)isEPSUsedOnResolutionMismatch	Returns whether to use EPS representations on mismatch
– setMatchedOnMultipleResolution:(BOOL) <i>flag</i>	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 *) <i>imageRep</i>	Removes <i>imageRep</i> from the List of representations

## Determining How the Image is Stored

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

## Determining How the Image is Drawn

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

## Assigning a Delegate

– <b>setDelegate:</b> <i>anObject</i>	Makes <i>anObject</i> the delegate of the NXImage
– <b>delegate</b>	Returns the delegate of the NXImage

## Producing TIFF Data for the Image

– <b>writeTIFF:</b> (NXTypedStream *) <i>stream</i>	Writes TIFF for the best representation to <i>stream</i>
– <b>writeTIFF:</b> (NXTypedStream *) <i>stream</i> <b>allRepresentations:</b> (BOOL) <i>flag</i>	Writes TIFF for all the representations to <i>stream</i>

## Managing NXImageRep subclasses

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

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

## Testing Image Data Sources

+ (BOOL)canInitFromPasteboard:(Pasteboard *) <i>pasteboard</i>	YES if NXImage can create a representation from <i>pasteboard</i>
+ (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 *) <i>stream</i>	Reads the NXImage and its representations from <i>stream</i>
– write:(NXTypedStream *) <i>stream</i>	Writes the NXImage and its representations to <i>stream</i>
– finishUnarchiving	Replaces the NXImage with one having the same name

## Methods Implemented by the Delegate

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

## NXImageRep

Inherits From: Object

### Initializing

– initFromPasteboard:(Pasteboard *) <i>pasteboard</i>	Initializes the receiver from <i>pasteboard</i>
---	---

### Checking data types

+ (BOOL)canInitFromPasteboard:(Pasteboard *) <i>pasteboard</i>	YES if NXImageRep can initialize itself from <i>pasteboard</i>
+ (BOOL)canLoadFromStream:(NXStream *) <i>stream</i>	YES if NXImageRep can initialize itself from <i>stream</i>

+ (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 *) <i>aSize</i>	Sets the size of the image
- getSize:(NXSize *) <i>theSize</i>	Copies the size of the image into the <i>theSize</i> structure

## Specifying Information about the Representation

- setNumColors:(int) <i>anInt</i>	Informs the object that there are <i>anInt</i> color components
- (int)numColors	Returns the number of color components
- setAlpha:(BOOL) <i>flag</i>	Informs object whether there is a coverage component
- (BOOL)hasAlpha	Returns whether there is a coverage component
- setBitsPerSample:(int) <i>anInt</i>	Informs object there are <i>anInt</i> bits/pixel in a component
- (int)bitsPerSample	Returns the number of bits per pixel in each component
- setPixelsHigh:(int) <i>anInt</i>	Informs object that data is for an image <i>anInt</i> pixels high
- (int)pixelsHigh	Returns the height specified in the image data
- setPixelsWide:(int) <i>anInt</i>	Informs object that data is for an image <i>anInt</i> 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 *) <i>point</i>	Modifies current coordinates so image is drawn at <i>point</i>
- (BOOL)drawIn:(const NXRect *) <i>rect</i>	Modifies current coordinates so image is drawn in <i>rect</i>

## Archiving

- read:(NXTypedStream *) <i>stream</i>	Reads the NXImageRep from <i>stream</i>
- write:(NXTypedStream *) <i>stream</i>	Writes the NXImageRep to <i>stream</i>

---

## 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*  
+ (NXPrinter \*)newForName:(const char \*)*name host:*(const char \*)*hostName*  
+ (NXPrinter \*)newForName:(const char \*)*name host:(const char \*)hostName domain:(const char \*)domain includeUnavailable:(BOOL)includeFlag*  
+ (NXPrinter \*)newForType:(const char \*)*type*  
+ (char \*\*)printerTypes:(BOOL)*normalFlag custom:(BOOL)customFlag*

Returns the NXPrinter with the given name  
Returns the NXPrinter with the given name *and host*  
Returns the NXPrinter with the given name, *host, and domain*  
Returns an NXPrinter object for a given printer type  
Returns the names of the recognized printer types

## Printer Attributes

- (const char \*)**domain**  
- (const char \*)**host**  
- (const char \*)**name**  
- (const char \*)**note**  
- (const char \*)**type**  
- (BOOL)**isReallyAPrinter**

Returns the name of the printer's domain  
Returns the name of the printer's host computer  
Returns the printer's name  
Returns the note associated with the printer  
**Returns the name of the printer's type**  
Returns whether the object corresponds to an actual printer

## Retrieving Specific Information

- (BOOL)**acceptsBinary**  
- (NXRect)**imageRectForPaper:**(const char \*)*paperType*  
- (NXSize)**pageSizeForPaper:**(const char \*)*paperType*  
  
- (BOOL)**isColor**  
- (BOOL)**isFontAvailable:**(const char \*)*name*  
- (BOOL)**isValid**

Returns YES if the printer accepts binary PostScript  
Returns the printing rectangle for the named paper type  
Returns the size of the page for the named paper type  
Returns whether the printer can print color  
Returns whether the named font is available to the printer  
Returns whether the NXPrinter is valid

– (int) <b>languageLevel</b>	Returns the PostScript Language Level recognized by the printer
– (BOOL) <b>isOutputStackInReverseOrder</b>	Returns whether the printer outputs pages in reverse page order

## Querying the NXPrinter Tables

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

## NXSpellChecker

Inherits From: Object

### Making A Checker Available

+ <b>sharedInstance</b>	Returns the NXSpellChecker to use
+ <b>sharedInstance: (BOOL)<i>flag</i></b>	Returns the NXSpellChecker to use but creates a new one only when <i>flag</i> is YES

## Managing The Spelling Panel

- |  |  |
|--|--|
| – <b>spellingPanel</b>                 | Returns the NXSpellChecker's panel           |
| – <b>accessoryView</b>                 | Returns the spell panel's accessory view     |
| – <b>setAccessoryView:<i>aView</i></b> | Makes a view an accessory of the spell panel |

## Checking Spelling

- |   |   |
|---|---|
| – (BOOL) <b>checkSpelling:(NXSpellCheckMode)<i>how</i></b><br><b>of:(id &lt;NXReadOnlyTextStream,</b><br><b>NXSelectRange&gt;)anObject</b>  | Starts the search for a misspelled word                           |
| – (BOOL) <b>checkSpelling:(NXSpellCheckMode)<i>how</i></b><br><b>of:(id &lt;NXReadOnlyTextStream,</b><br><b>NXSelectRange&gt;)anObject</b><br><b>wordCount:(int *)<i>theCount</i></b> | Starts the search for a misspelled word and the count of<br>words |

## Managing the Language Being Checked

- |   |                                       |
|---|---------------------------------------|
| – (const char*) <b>language</b>                     | Returns the current spelling language |
| – <b>setLanguage:(const char *)<i>aLanguage</i></b> | Sets the current spelling language    |

## Managing Ignored Words

- |   |  |
|---|--|
| – <b>closeSpellClient:<i>aClient</i></b>  | Notifies the NXSpellChecker that a document has closed |
| – (char **) <b>ignoredWordsForSpellClient:<i>aClient</i></b>  | Returns the list of ignored words for a document       |
| – <b>setIgnoredWords:(const char *const *)<i>someWords</i></b><br><b>forSpellClient:(int)<i>tag</i></b> | Initializes the list of ignored words for a document   |

---

## NXSpellServer

**Inherits From:** Object

## Checking in Your Service

- |   |
|---|
| – (BOOL) <b>registerLanguage:(const char *)<i>language</i></b><br><b>byVendor:(const char *)<i>vendor</i></b> |
|---|

## 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* caseSensitive:(BOOL)*flag*** Returns YES if the word is in any open user dictionary

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

# NXSplitView

Inherits From: View : Responder : Object

## Initializing an NXSplitView

– **initFrame:(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  
– **setAutoresizingMask:(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*  
    maxY:(NXCoord \*)*maxY*  
    ofSubviewAt:(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

- |  |   |
|--|---|
| – <b>perform:(SEL)<i>aSelector</i></b><br><b>with:<i>anObject</i></b><br><b>afterDelay:(int)<i>ms</i></b><br><b>cancelPrevious:(BOOL)<i>flag</i></b> | Sends an <i>aSelector</i> message to the receiver after <i>ms</i> delay |
|--|---|

---

## OpenPanel

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

### Creating and Freeing an OpenPanel

- |   |                                     |
|---|-------------------------------------|
| + <b>new</b>  | Returns the shared OpenPanel object |
| + <b>newContent:(const NXRect *)<i>contentRect</i></b><br><b>style:(int)<i>aStyle</i></b><br><b>backing:(int)<i>bufferingType</i></b><br><b>buttonMask:(int)<i>mask</i></b><br><b>defer:(BOOL)<i>flag</i></b> | Returns the shared OpenPanel object |
| – <b>free</b>   | Deallocates the OpenPanel object    |

### Setting the OpenPanel Class

- |   |  |
|---|--|
| + <b>setOpenPanelFactory:<i>class</i></b> | Sets class for initializing an OpenPanel |
|---|--|

### Filtering Files

- |   |   |
|---|---|
| – <b>allowMultipleFiles:(BOOL)<i>flag</i></b> | Sets whether the user can open multiple files |
|---|---|

### Querying the Chosen Files

- |   |                                      |
|---|--------------------------------------|
| – <b>(const char *const *)filenames</b> | Gets the names of the selected files |
|---|--------------------------------------|

## Running the OpenPanel

- (int)**runModalForDirectory:**(const char \*)*path* file:(const char \*)*name* Displays the panel and begins its event loop
- (int)**runModalForDirectory:**(const char \*)*path* file:(const char \*)*name* types:(const char \*const \*)*fileTypes* Displays the panel and begins its event loop
- (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* style:(int )*aStyle* backing:(int )*bufferingType* buttonMask:(int )*mask* defer:(BOOL )*flag* Used in PageLayout instantiation
- 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

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

## Communicating with the PrintInfo Object

<b>- readPrintInfo</b>	Reads the PageLayout's values from the PrintInfo object
<b>- writePrintInfo</b>	Writes the PageLayout's values to the PrintInfo object

## Panel

**Inherits From:** Window : Responder : Object

### Initializing a New Panel

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

### Handling Events

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

### Determining the Panel Interface

<b>- setBecomeKeyOnlyIfNeeded:</b> (BOOL) <i>flag</i>	Sets whether Panel waits to become key window
<b>- doesBecomeKeyOnlyIfNeeded</b>	Returns whether Panel waits to become key window
<b>- setFloatingPanel:</b> (BOOL) <i>flag</i>	Sets whether the Panel floats above other windows

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

## Pasteboard

Inherits From: Object

### Creating and Freeing a Pasteboard

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

### Getting Data in Different Formats

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

### Referring to a Pasteboard by Name

<code>+ newName:(const char *)<i>name</i></code>	Returns the Pasteboard object named <i>name</i>
<code>- (const char *)<b>name</b></code>	Returns the Pasteboard object's name

### Writing Data

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

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

## Discerning Types

– (const NXAtom *) <i>types</i>	Returns an array of the Pasteboard's data types
– (const char *) <b>findAvailableTypeFrom</b> : (const char *const *) <i>types</i>	Returns first type in <i>types</i> that matches a pasteboard type

## Reading Data

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

## Methods Implemented by the Owner

– <b>pasteboard</b> : <i>sender</i> <b>provideData</b> :(NXAtom) <i>type</i>	Implemented to write promised data to <i>sender</i> as <i>type</i>
– <b>pasteboardChangedOwner</b> : <i>sender</i>	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

- |  |   |
|--|---|
| <b>- addItem:(const char *)title</b>                               | Adds an item with <i>title</i> as its title to the end of the list      |
| <b>- insertItem:(const char *)title<br/>at:(unsigned int)index</b> | Inserts an item with <i>title</i> as its title at position <i>index</i> |
| <b>- removeItem:(const char *)title</b>                            | Removes the item matching <i>title</i>                                  |
| <b>- removeItemAt:(unsigned int)index</b>                          | Removes the item at the specified <i>index</i>                          |
| <b>- (int)indexOfItem:(const char *)title</b>                      | Returns the index of the item matching <i>title</i>                     |
| <b>- (unsigned int)count</b>                                       | 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

- |                                    |   |
|------------------------------------|---|
| <b>- setFont:<i>fontObject</i></b> | Sets the Font used to draw the items    |
| <b>- font</b>                      | Returns the Font used to draw the items |

## Target and Action

– <b>setAction:</b> (SEL) <i>aSelector</i>	Sets the PopUpList's action method to <i>aSelector</i>
– <b>(SEL)action</b>	Returns the PopUpList's action method
– <b>setTarget:</b> <i>anObject</i>	Sets the PopUpList's target object to <i>anObject</i>
– <b>target</b>	Returns the PopUpList's target object

## Resizing the PopUpList

– <b>sizeWindow:</b> (NXCoord) <i>width</i> :(NXCoord) <i>height</i>	Resizes the PopUpList to <i>width</i> , <i>height</i>
--	---

---

## PrintInfo

Inherits From: Object

### Initializing and Freeing a PrintInfo Instance

– <b>init</b>	Initializes the PrintInfo instance after it's allocated
– <b>free</b>	Deallocates the PrintInfo object

### Defining the Printing Rectangle

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

## Page Range

– <b>setFirstPage:(int)<i>anInt</i></b>	Sets the page number of first page to be printed
– <b>(int)firstPage</b>	Returns the page number of the first page to be printed
– <b>setLastPage:(int)<i>anInt</i></b>	Sets the page number of last page to be printed
– <b>(int)lastPage</b>	Returns the page number of the last page to be printed
– <b>setAllPages:(BOOL)<i>flag</i></b>	Sets whether all the pages are to be printed
– <b>(BOOL)isAllPages</b>	Returns whether all the pages are to be printed
– <b>(int)currentPage</b>	Returns the page number of the page being printed

## Pagination and Scaling

– <b>setHorizPagination:(int)<i>mode</i></b>	Sets the horizontal pagination mode
– <b>(int)horizPagination</b>	Returns the horizontal pagination mode
– <b>setVertPagination:(int)<i>mode</i></b>	Sets the vertical pagination mode
– <b>(int)vertPagination</b>	Returns the vertical pagination mode
– <b>setScalingFactor:(float)<i>aFloat</i></b>	Sets the scaling factor
– <b>(float)scalingFactor</b>	Returns the scaling factor

## Positioning the Image on the Page

– <b>setHorizCentered:(BOOL)<i>flag</i></b>	Sets whether the image is centered horizontally
– <b>(BOOL)isHorizCentered</b>	Returns whether the image is centered horizontally
– <b>setVertCentered:(BOOL)<i>flag</i></b>	Sets whether the image is centered vertically
– <b>(BOOL)isVertCentered</b>	Returns whether the image is centered vertically
– <b>setPagesPerSheet:(short)<i>aShort</i></b>	Sets the number of pages printed per sheet of paper
– <b>(short)pagesPerSheet</b>	Returns the number of pages printed per sheet of paper

## Print Job Attributes

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

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

## Specifying the Printer

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

## Spooling

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

## Archiving

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

## PrintPanel

**Inherits From:** Panel : Window : Responder : Object

### Creating and Freeing a PrintPanel

<b>+ new</b>	Returns a default PrintPanel object
<b>+ newContent:(const NXRect *)contentRect style:(int )aStyle backing:(int )bufferingType buttonMask:(int )mask defer:(BOOL )flag</b>	Returns a PrintPanel object
<b>- free</b>	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
- `rightMouseDragged:(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  
andReturnType:(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 style:(int)aStyle backing:(int)bufferingType buttonMask:(int)mask defer:(BOOL)flag	Creates and returns a SavePanel object
- free	Deallocates the SavePanel

### Setting the SavePanel Class

+ setSavePanelFactory: <i>class</i>	Sets class for initializing an SavePanel
-------------------------------------	--

### Customizing the SavePanel

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

### Setting Directory and File Type

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

### Running the SavePanel

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

– **initFrame:(const NXRect \*)frameRect** Initializes a new Scroller

## Laying out the Scroller

- `(NXRect *)calcRect:(NXRect *)aRect  
forPart:(int)partCode`
- `checkSpaceForParts`
- `setArrowsPosition:(int)where`

Gets the rectangle that encloses *partCode*

Checks for room for knob and scroll buttons

Sets position of scroll buttons in Scroller

## Setting Scroller values

- `(float)floatValue`
- `setFloatValue:(float)aFloat`
- `setFloatValue:(float)aFloat :(float)percent`

Returns Scroller's float value

Sets value; positions knob

Sets value; positions and sizes knob

## Resizing the Scroller

- `sizeTo:(NXCoord)width :(NXCoord)height`

Sizes the Scroller

## Displaying

- `drawArrow:(BOOL)whichButton :(BOOL)flag`
- `drawKnob`
- `drawParts`
- `drawSelf:(const NXRect *)rects :(int)rectCount`
- `highlight:(BOOL)flag`

Draws highlighted and unhighlighted arrows

Draws the knob

Caches Bitmaps for knob and scroll arrows

Draws the Scroller

Highlights scroll button that's under mouse

## Target and Action

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

Sets the Scroller's action to *aSelector*

Returns the Scroller's action

Sets the Scroller's target to *anObject*

Returns the Scroller's target

## Handling Events

- `(BOOL)acceptsFirstMouse`
- `(int)hitPart`
- `mouseDown:(NXEvent *)theEvent`
- `(int)testPart:(const NXPoint *)thePoint`
- `trackKnob:(NXEvent *)theEvent`
- `trackScrollButtons:(NXEvent *)theEvent`

Makes the Scroller respond to the first mouse event

Returns Scroller part that received mouse-down

Responds to mouse-down events

Returns Scroller part that's under *thePoint*

Responds to mouse-down events on the knob

Responds to mouse-down events on buttons

## Archiving

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

---

## ScrollView

**Inherits From:** View : Responder : Object

### Initializing a ScrollView

- **initFrame:(const NXRect \*)frameRect**

Initializes a new ScrollView

### Determining Component Sizes

- **getContentSize:(NXSize \*)contentViewSize**
- **getDocVisibleRect:(NXRect \*)aRect**

Gets the content view's size

Gets the visible portion of the document view

### Laying Out the ScrollView

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

## Managing Component Views

– <b>setDocView:</b> <i>aView</i>	Makes <i>aView</i> the ScrollView's document view
– <b>docView</b>	Returns the current document view
– <b>setHorizScroller:</b> <i>anObject</i>	Sets the horizontal Scroller object
– <b>horizScroller</b>	Returns the horizontal Scroller
– <b>setVertScroller:</b> <i>anObject</i>	Sets the vertical Scroller object
– <b>vertScroller</b>	Returns the vertical Scroller
– <b>reflectScroll:</b> <i>cView</i>	Updates the Scrollers

## Modifying Graphic Attributes

– <b>setBorderType:</b> (int) <i>aType</i>	Determines the border type of the ScrollView
– (int) <b>borderType</b>	Returns the border type
– <b>setBackgroundColor:</b> (NXColor) <i>color</i>	Sets the ScrollView's background color
– (NXColor) <b>backgroundColor</b>	Returns the ScrollView's background color
– <b>setBackgroundGray:</b> (float) <i>value</i>	Sets the ScrollView's background gray
– (float) <b>backgroundGray</b>	Returns the ScrollView's background gray

## Setting Scrolling Behavior

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

## Displaying

– <b>drawSelf:</b> (const NXRect *) <i>rects</i> :(int) <i>rectCount</i>	Draws the ScrollView
--	----------------------

## Managing the Cursor

– <b>setDocCursor:</b> <i>anObj</i>	Sets the cursor for the document view
-------------------------------------	---------------------------------------

## Archiving

– <b>read:</b> (NXTypedStream *) <i>stream</i>	Reads the ScrollView from the typed stream
– <b>write:</b> (NXTypedStream *) <i>stream</i>	Writes the ScrollView to the typed stream

---

## SelectionCell

Inherits From: Cell : Object

### Creating a SelectionCell

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

- **initFrame:**(const NXRect \*)*frameRect* Initializes a new Slider in *frameRect*

## Modifying a Slider's appearance

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

## Setting Value Limits

- <b>setMinValue:</b> (double) <i>aDouble</i>	Sets the Slider's minimum value to <i>aDouble</i>
- (double) <b>minValue</b>	Returns the Slider's minimum value
- <b>setMaxValue:</b> (double) <i>aDouble</i>	Sets the Slider's maximum value to <i>aDouble</i>
- (double) <b>maxValue</b>	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) <b>intValue</b>	Returns SliderCell’s value as an <b>int</b>
– <b>setStringValue:</b> (const char *) <i>aString</i>	Sets the SliderCell’s value to a number represented by <i>aString</i>
– (const char *) <b>stringValue</b>	Returns the the SliderCell’s value as a string

## Modifying Graphic Attributes

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

## Displaying the SliderCell

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

## Modifying Behavior

– <b>setEnabled:</b> (BOOL) <i>flag</i>	Sets whether the SliderCell reacts to events
– <b>setContinuous:</b> (BOOL) <i>flag</i>	Sets whether the Slider is continuous
– (BOOL) <b>isContinuous</b>	Returns whether the Slider is continuous

– <b>setAltIncrementValue:</b> (double) <i>incValue</i>	Sets how far the SliderCell moves when the knob is dragged one pixel with the Alternate key held down
– (double) <b>altIncrementValue</b>	Returns how far the SliderCell moves when alt-dragged

## Tracking the Mouse

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

## Archiving

– <b>read:</b> (NXTypedStream *) <i>stream</i>	Reads the SliderCell from <i>stream</i>
– <b>write:</b> (NXTypedStream *) <i>stream</i>	Writes the SliderCell to <i>stream</i>
– <b>awake</b>	Caches knob icons when the SliderCell is unarchived

## Speaker

**Inherits From:** Object

### Initializing a New Speaker Instance

– <b>init</b>	Initializes the Speaker after it has been allocated
---------------	---

### Freeing a Speaker

– <b>free</b>	Deallocates the Speaker (but not its ports)
---------------	---

## Setting Up a Speaker

– <b>setSendTimeout:(int)<i>ms</i></b>	Sets how long to wait for messages to be delivered
– <b>(int)sendTimeout</b>	Returns how long to wait for messages to be delivered
– <b>setReplyTimeout:(int)<i>ms</i></b>	Sets how long Speaker will wait for a reply
– <b>(int)replyTimeout</b>	Returns how long Speaker will wait for a reply

## Managing the Ports

– <b>setSendPort:(port_t)<i>aPort</i></b>	Makes <i>aPort</i> the port messages will be sent to
– <b>(port_t)sendPort</b>	Returns the port the Speaker will send messages to
– <b>setReplyPort:(port_t)<i>aPort</i></b>	Makes <i>aPort</i> the port where replies are received
– <b>(port_t)replyPort</b>	Returns the port where Speaker receives replies

## Standard Remote Methods

– <b>(int)openFile:(const char *)<i>fullPath</i>     ok:(int *)<i>flag</i></b>	Sends a remote message to open <i>fullPath</i> file
– <b>(int)openTempFile:(const char *)<i>fullPath</i>     ok:(int *)<i>flag</i></b>	Sends a remote message to open <i>fullPath</i> file

## Providing for Program Control

– <b>(int)msgCalc:(int *)<i>flag</i></b>	Sends message to update the current window
– <b>(int)msgCopyAsType:(const char *)<i>aType</i>     ok:(int *)<i>flag</i></b>	Sends message to copy selection as <i>aType</i> data
– <b>(int)msgCutAsType:(const char *)<i>aType</i>     ok:(int *)<i>flag</i></b>	Sends message to cut selection as <i>aType</i> data
– <b>(int)msgDirectory:(char *const *)<i>fullPath</i>     ok:(int *)<i>flag</i></b>	Sends message requesting the current directory
– <b>(int)msgFile:(char *const *)<i>fullPath</i>     ok:(int *)<i>flag</i></b>	Sends message requesting the current document
– <b>(int)msgPaste:(int *)<i>flag</i></b>	Sends message to paste data from pasteboard
– <b>(int)msgPosition:(char *const *)<i>aString</i>     posType:(int *)<i>anInt</i>     ok:(int *)<i>flag</i></b>	Sends message requesting selection information
– <b>(int)msgPrint:(const char *)<i>fullPath</i>     ok:(int *)<i>flag</i></b>	Sends message to print <i>fullPath</i> file
– <b>(int)msgQuit:(int *)<i>flag</i></b>	Sends remote message for application 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>	Sends message requesting the current selection
– (int)msgSetPosition:(const char *) <i>aString</i> posType:(int) <i>anInt</i> andSelect:(int) <i>sflag</i> ok:(int *) <i>flag</i>	Sends message to scroll so <i>aString</i> is visible
– (int)msgVersion:(char *const *) <i>aString</i> ok:(int *) <i>flag</i>	Sends message requesting version information

## Sending Remote Messages

– (int)performRemoteMethod:(const char *) <i>methodName</i>	Sends remote <i>methodName</i> message
– (int)performRemoteMethod:(const char *) <i>methodName</i> with:(const char *) <i>data</i> length:(int) <i>numBytes</i>	Sends remote message with <i>numBytes</i> of <i>data</i>
– (int)selectorRPC:(const char *) <i>methodName</i> paramTypes:(char *) <i>params</i> ,	Sends remote message with variable arguments
– (int)sendOpenFileMsg:(const char *) <i>fullPath</i> ok:(int *) <i>flag</i> andDeactivateSelf:(BOOL) <i>deactivateFirst</i>	Sends an <b>openFile:ok:</b> remote message
– (int)sendOpenTempFileMsg:(const char *) <i>fullPath</i> ok:(int *) <i>flag</i> andDeactivateSelf:(BOOL) <i>deactivateFirst</i>	Sends an <b>openTempFile:ok:</b> remote message

## Assigning a Delegate

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

## Archiving

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

---

## Text

**Inherits From:** View : Responder : Object

**Conforms To:** NXChangeSpelling  
NXIgnoreMisspelledWords  
NXReadOnlyTextStream  
NXSelectText

### Initializing the Class Object

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

### Initializing a New Text Object

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

### Freeing a Text Object

- <b>free</b>	Frees the Text object and its storage
---------------	---------------------------------------

### Modifying the Frame Rectangle

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

- <b>sizeToFit</b>	Resizes the frame to accommodate the text
- <b>resizeText:(const NXRect *)oldBounds           :(const NXRect *)maxRect</b>	Used by Text object to resize and redisplay itself
- <b>moveTo:(NXCoord)x :(NXCoord)y</b>	Moves the Text object to (x, y)
 <b>Laying Out the Text</b>	
- <b>setMarginLeft:(NXCoord)leftMargin           right:(NXCoord)rightMargin           top:(NXCoord)topMargin           bottom:(NXCoord)bottomMargin</b>	Adjusts margins around the text
- <b>getMarginLeft:(NXCoord *)leftMargin           right:(NXCoord *)rightMargin           top:(NXCoord *)topMargin           bottom:(NXCoord *)bottomMargin</b>	Gets dimensions of margins around the text
- <b>getMinWidth:(NXCoord *)width           minHeight:(NXCoord *)height           maxWidth:(NXCoord)widthMax           maxHeight:(NXCoord)heightMax</b>	Calculates area needed to display the text
- <b>setAlignment:(int)mode</b>	Sets how text is aligned at margins
- <b>(int)alignment</b>	Returns how text is aligned at margins
- <b>alignSelLeft:sender</b>	Aligns the text to the left margin
- <b>alignSelCenter:sender</b>	Aligns the text between the margins
- <b>alignSelRight:sender</b>	Aligns the text to the right margin
- <b>setSelProp:(NXParagraphProp)prop           to:(NXCoord)val</b>	Sets the paragraph style for one or more paragraphs
- <b>changeTabStopAt:(NXCoord)oldX           to:(NXCoord)newX</b>	Resets the position of the specified tab stop
- <b>(int)calcLine</b>	Calculates line breaks
- <b>setCharWrap:(BOOL)flag</b>	Returns whether extra long words are wrapped
- <b>(BOOL)charWrap</b>	Sets whether extra long words are wrapped
- <b>setNoWrap</b>	Disables word wrap
- <b>setParaStyle:(void *)paraStyle</b>	Sets paragraph style for the entire text
- <b>(void *)defaultParaStyle</b>	Returns the default paragraph style
- <b>(void *)calcParagraphStyle:fontId           :(int)alignment</b>	Recalculates paragraph style
- <b>setLineHeight:(NXCoord)value</b>	Sets height of a line of text
- <b>(NXCoord)lineHeight</b>	Returns height of a line of text
- <b>setDescentLine:(NXCoord)value</b>	Sets distance from base line to bottom of line
- <b>(NXCoord)descentLine</b>	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)saveRTFDTTo:(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*
- writeRTFDTTo:(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

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

## Reusing a Text Object

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

## Displaying

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

## Assigning a Tag

- |                                   |  |
|-----------------------------------|--|
| – <b>setTag:(int)<i>anInt</i></b> | Makes <i>anInt</i> the Text object's tag |
| – <b>(int)tag</b>                 | Returns the Text object's tag            |

## Handling Event Messages

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

## Displaying Graphics within the Text

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

## Using the Services Menu

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

## Setting Tables and Functions

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

## Printing

– <b>adjustPageHeightNew:</b> (float *) <i>newBottom</i> <i>top</i> :(float) <i>oldTop</i> <i>bottom</i> :(float) <i>oldBottom</i> <i>limit</i> :(float) <i>bottomLimit</i>	Assists automatic pagination of text
--	--------------------------------------

## Archiving

– <b>read:</b> (NXTypedStream *) <i>stream</i>	Reads the Text object from the typed stream
– <b>write:</b> (NXTypedStream *) <i>stream</i>	Writes the Text object to the typed stream

## Assigning a Delegate

– <b>setDelegate:</b> <i>anObject</i>	Makes <i>anObject</i> the Text object's delegate
– <b>delegate</b>	Returns the Text object's delegate

## Implemented by the Delegate

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

## Implemented by an Embedded Graphic Object

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

---

## TextField

**Inherits From:** Control : View : Responder : Object

### Initializing the TextField Class

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

### Initializing a new TextField

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

Responds to a message from the field editor  
Responds to a message from the field editor

Responds to a message from the field editor  
Responds to a message from the field editor  
Responds to a message from the field editor

## Modifying Graphic Attributes

- setTextColor:(NXColor)*aColor*
- (NXColor)textColor
- setTextGray:(float)*value*
- (float)textGray
- setBackgroundColor:(NXColor)*aColor*
- (NXColor)backgroundColor
- setBackgroundGray:(float)*value*
- (float)backgroundGray
- setBackgroundTransparent:(BOOL)*flag*
- (BOOL)isBackgroundTransparent
- setBezeled:(BOOL)*flag*
- (BOOL)isBezeled
- setBordered:(BOOL)*flag*
- (BOOL)isBordered

Sets the color of the TextField's text to *aColor*  
Returns the color of the TextField's text  
Sets the gray of the TextField's text to *value*  
Returns the gray of the TextField's text  
Sets the color of the background to *aColor*  
Returns the color of the background  
Sets the gray of the background to *value*  
Returns the gray of the background  
Sets whether the TextField background is transparent  
Returns whether the TextField background is transparent  
Sets whether the TextField has a bezeled border  
Returns whether the TextField has a bezeled border  
Sets whether the TextField has a plain border  
Returns whether the TextField has a plain border

## Target and Action

- setErrorAction:(SEL)*aSelector*
- (SEL)errorAction

Sets the action method sent for an invalid value entered  
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
- mouseDown:(NXEvent \*)*theEvent*

Returns YES if text is editable or selectable  
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”
- **initTextCell:(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
- **setBackgroundColor:(NXColor)aColor**      Sets the color of the background to *aColor*
- **(NXColor)backgroundColor**      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

## Displaying

- drawSelf:(const NXRect \*)cellFrame  
inView:*controlView*** Draws the TextFieldCell
  - drawInside:(const NXRect \*)cellFrame  
inView:*controlView*** Draws the inside of the TextFieldCell

## Tracking the Mouse



## Archiving

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

## View

**Inherits From:** Responder : Object

## Initializing and Freeing View Objects

- |  |                                       |
|--|---------------------------------------|
| <code>- initFrame:(const NXRect *)frameRect</code> | Initializes a new View object         |
| <code>- init</code>                                | Initializes a new View object         |
| <code>- free</code>                                | Deallocates the View and its subviews |

## Managing the View Hierarchy

- |   |  |
|---|--|
| <b>– addSubview:<i>aView</i></b>  | Makes <i>aView</i> a subview of the receiving View           |
| <b>– addSubview:<i>aView</i><br/>:(int)<i>place</i><br/>relativeTo:<i>otherView</i></b> | Makes <i>aView</i> a subview of the receiving View           |
| <b>– findAncestorSharedWith:<i>aView</i></b>  | Returns the ancestor shared by <i>aView</i> and the receiver |
| <b>– (BOOL)isDescendantOf:<i>aView</i></b>  | Returns whether <i>aView</i> is an ancestor of the receiver  |
| <b>– opaqueAncestor</b>   | Returns the receiver's nearest opaque ancestor               |
| <b>– removeFromSuperview</b>  | Removes the receiver from the view hierarchy                 |

<code>- replaceSubview:<i>oldView</i> with:<i>newView</i></code>	Replaces <i>oldView</i> with <i>newView</i>
<code>- subviews</code>	Returns a List of the View's subviews
<code>- superview</code>	Returns the receiving View's superview
<code>- window</code>	Returns the Window in which the View is displayed
<code>- windowChanged:<i>newWindow</i></code>	Notifies the View that the Window it's in is changing

## Modifying the Frame Rectangle

<code>- (float)frameAngle</code>	Returns the angle of frame rectangle rotation
<code>- getFrame:(NXRect *)<i>theRect</i></code>	Gets the View's frame rectangle
<code>- moveBy:(NXCoord)<i>deltaX</i> :(NXCoord)<i>deltaY</i></code>	Moves the View by <i>deltaX</i> and <i>deltaY</i>
<code>- moveTo:(NXCoord)<i>x</i> :(NXCoord)<i>y</i></code>	Moves the View to ( <i>x</i> , <i>y</i> )
<code>- rotateBy:(NXCoord)<i>deltaAngle</i></code>	Rotates the View's frame rectangle by <i>deltaAngle</i>
<code>- rotateTo:(NXCoord)<i>angle</i></code>	Rotates the View's frame rectangle to <i>angle</i>
<code>- setFrame:(const NXRect *)<i>frameRect</i></code>	Assigns the View a new frame rectangle
<code>- sizeBy:(NXCoord)<i>deltaWidth</i> :(NXCoord)<i>deltaHeight</i></code>	Resizes the View by <i>deltaWidth</i> and <i>deltaHeight</i>
<code>- sizeTo:(NXCoord)<i>width</i> :(NXCoord)<i>height</i></code>	Resizes the View to <i>width</i> and <i>height</i>

## Modifying the Coordinate System

<code>- (float)boundsAngle</code>	Returns the rotation of the View's coordinate system
<code>- drawInSuperview</code>	Makes the View use its superview's coordinate system
<code>- getBounds:(NXRect *)<i>theRect</i></code>	Gets the View's bounds rectangle
<code>- (BOOL)isFlipped</code>	Returns whether the View is flipped
<code>- (BOOL)isRotatedFromBase</code>	Returns whether the View is rotated
<code>- (BOOL)isRotatedOrScaledFromBase</code>	Returns whether the View is rotated or scaled
<code>- rotate:(NXCoord)<i>angle</i></code>	Rotates the View's coordinate system by <i>angle</i>
<code>- setDrawRotation:(NXCoord)<i>angle</i></code>	Rotates the View's coordinate system to <i>angle</i>
<code>- scale:(NXCoord)<i>x</i> :(NXCoord)<i>y</i></code>	Scales the View's coordinate system
<code>- setDrawSize:(NXCoord)<i>width</i> :(NXCoord)<i>height</i></code>	Resizes the View's coordinate system to <i>width</i> and <i>height</i>
<code>- translate:(NXCoord)<i>x</i> :(NXCoord)<i>y</i></code>	Shifts the View's coordinate system to ( <i>x</i> , <i>y</i> )
<code>- setDrawOrigin:(NXCoord)<i>x</i> :(NXCoord)<i>y</i></code>	Sets the View's origin to ( <i>x</i> , <i>y</i> )
<code>- setFlipped:(BOOL)<i>flag</i></code>	Sets whether polarity of y-axis is reversed

## Converting Coordinates

<code>- centerScanRect:(NXRect *)<i>aRect</i></code>	Converts the rectangle to lie on center of pixels
--	---

<b>– convertPoint:(NXPoint *)<i>aPoint</i> fromView:<i>aView</i></b>	Converts the point to the receiver's coordinates
<b>– convertPoint:(NXPoint *)<i>aPoint</i> toView:<i>aView</i></b>	Converts the point to <i>aView</i> 's coordinates
<b>– convertPointFromSuperview:(NXPoint *)<i>aPoint</i></b>	Converts the point to the receiver's coordinates
<b>– convertPointToSuperview:(NXPoint *)<i>aPoint</i></b>	Converts the point to the superview's coordinates
<b>– convertRect:(NXRect *)<i>aRect</i> fromView:<i>aView</i></b>	Converts the rectangle to the receiver's coordinates
<b>– convertRect:(NXRect *)<i>aRect</i> toView:<i>aView</i></b>	Converts the rectangle to <i>aView</i> 's coordinates
<b>– convertRectFromSuperview:(NXRect *)<i>aRect</i></b>	Converts the rectangle to the receiver's coordinates
<b>– convertRectToSuperview:(NXRect *)<i>aRect</i></b>	Converts the rectangle to the superview's coordinates
<b>– convertSize:(NXSize *)<i>aSize</i> fromView:<i>aView</i></b>	Converts the size to the receiver's coordinates
<b>– convertSize:(NXSize *)<i>aSize</i> toView:<i>aView</i></b>	Converts the size to <i>aView</i> 's coordinates

## Notifying Ancestor Views

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

## Resizing Subviews

<b>– resizeSubviews:(const NXSize *)<i>oldSize</i></b>	Initiates <b>superviewSizeChanged:</b> messages
<b>– setAutoresizingMask:(BOOL)<i>flag</i></b>	Sets whether to notify subviews of resizing
<b>– setAutosizing:(unsigned int)<i>mask</i></b>	Determines automatic resizing behavior
<b>– (unsigned int)autosizing</b>	Returns the View's autosizing mask
<b>– superviewSizeChanged:(const NXSize *)<i>oldSize</i></b>	Notifies subviews that superview changed size

## Graphics State Objects

<b>– allocateGState</b>	Allocates a graphics state object (when next focused upon)
<b>– freeGState</b>	Frees the View's graphics state object
<b>– (int)gState</b>	Returns the View's graphics state object
<b>– initGState</b>	Initializes the View's graphics state object
<b>– renewGState</b>	Reinitializes the View's graphics state object
<b>– notifyToInitGState:(BOOL)<i>flag</i></b>	Determines whether <b>initGState</b> message is sent

## Focusing

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

## Displaying

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

## Scrolling

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

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

## Managing the Cursor

– <b>addCursorRect:</b> (const NXRect *) <i>aRect</i> <b>cursor:</b> <i>anObj</i>	Adds a cursor rectangle to the View
– <b>discardCursorRects</b>	Removes all cursor rectangles in the View
– <b>removeCursorRect:</b> (const NXRect *) <i>aRect</i> <b>cursor:</b> <i>anObj</i>	Removes a cursor rectangle from the View
– <b>resetCursorRects</b>	Resets the View's cursor rectangles

## Assigning a Tag

– <b>findViewWithTag:</b> (int) <i>aTag</i>	Returns the subview with <i>aTag</i> as its tag
– (int) <b>tag</b>	Returns the View's tag

## Aiding Event Handling

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

## Dragging

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

– <b>dragImage:</b> <i>anImage</i>	Instigates an image-dragging session
<i>at:(NXPoint *)location</i> <i>offset:(NXPoint *)initialOffset</i> <i>event:(NXEvent *)event</i> <i>pasteboard:(Pasteboard *)pboard</i> <i>source:sourceObject</i> <i>slideBack:(BOOL)slideFlag</i>	
– <b>dragFile:</b> (const char *) <i>filename</i>	Instigates a file-dragging session
<i>fromRect:(NXRect *)rect</i> <i>slideBack:(BOOL) aFlag</i> <i>event:(NXEvent *)event</i>	

## Printing

– <b>printPSCode:</b> <i>sender</i>	Prints the View and its subviews
– <b>faxPSCode:</b> <i>sender</i>	Faxes the View and its subviews
– <b>faxPSCode:</b> <i>sender</i>	Faxes the View and its subviews
<i>toList:(const char *const *)names</i> <i>numberList:(const char *const *)numbers</i> <i>sendAt:(time_t)time</i> <i>wantsCover:(BOOL)coverFlag</i> <i>wantsNotify:(BOOL)notifyFlag</i> <i>wantsHires:(BOOL)hiresFlag</i> <i>faxName:(const char *)string</i>	
– <b>copyPSCodeInside:</b> (const NXRect *) <i>rect</i>	Generates PostScript code for the rectangle
<i>to:(NXStream *)stream</i>	
– <b>writePSCodeInside:</b> (const NXRect *) <i>rect</i>	Places PostScript code for the rectangle on the pasteboard
<i>to:pasteboard</i>	
– <b>openSpoolFile:</b> (char *) <i>filename</i>	Opens <i>filename</i> for print spooling
– <b>spoolFile:</b> (const char *) <i>filename</i>	Spools <i>filename</i> to the printer
– (BOOL) <b>canPrintRIB</b>	Indicates whether the View can print RIB files

## Setting Up Pages

– (BOOL) <b>knowsPagesFirst:</b> (int *) <i>firstPageNum</i>	Returns whether the View paginates itself
<i>last:(int *)lastPageNum</i>	
– (BOOL) <b>getRect:</b> (NXRect *) <i>theRect</i>	Provides how much of the View will print on <i>page</i>
<i>forPage:(int)page</i>	
– <b>placePrintRect:</b> (const NXRect *) <i>aRect</i>	Locates the printing rectangle on the page
<i>offset:(NXPoint *)location</i>	
– (float) <b>heightAdjustLimit</b>	Returns how much of a page can go on the next page
– (float) <b>widthAdjustLimit</b>	Returns how much of a page can go on the next page

## Writing Conforming PostScript

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

## Archiving

– <b>awake</b>	Initializes the View after reading
– <b>read:</b> (NXTypedStream *) <i>stream</i>	Reads the View from the typed stream
– <b>write:</b> (NXTypedStream *) <i>stream</i>	Writes the View to the typed stream

## Window

**Inherits From:** Responder : Object

### Initializing a New Window Object

- **init**
  - **initContent:(const NXRect \*)contentRect  
style:(int)aStyle  
backing:(int)bufferingType  
buttonMask:(int)mask  
defer:(BOOL)flag**
  - **initContent:(const NXRect \*)contentRect  
style:(int)aStyle  
backing:(int)bufferingType  
buttonMask:(int)mask  
defer:(BOOL)flag  
screen:(const NXScreen \*)aScreen**
- Initializes the new Window with default parameters  
Initializes the new Window object as specified  
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**
  - + **getContentRect:(NXRect \*)content  
forFrameRect:(const NXRect \*)frame  
style:(int)aStyle**
  - + **(NXCoord)minFrameWidth:(const char \*)aTitle  
forStyle:(int)aStyle  
buttonMask:(int)aMask**
- Gets frame rectangle for given content rectangle  
Gets content rectangle for given frame rectangle  
Returns minimum frame width needed for *aTitle*

### Accessing the Frame Rectangle

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

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

## Accessing the Content View

– <b>setContentView:</b> <i>aView</i>	Makes <i>aView</i> the Window's content view
– <b>contentView</b>	Returns the Window's content view

## Querying Window Attributes

– (int) <b>windowNum</b>	Returns the window number
– (int) <b>buttonMask</b>	Returns the mask identifying Window controls
– (int) <b>style</b>	Returns the Window's border and title bar style
– (BOOL) <b>worksWhenModal</b>	Returns NO

## Window Graphics

– <b>setTitle:</b> (const char *) <i>aString</i>	Makes <i>aString</i> the Window's title
– <b>setTitleAsFilename:</b> (const char *) <i>aString</i>	Formats <i>aString</i> and makes it the Window's title
– (const char *) <b>title</b>	Returns the Window's title string
– <b>setBackgroundColor:</b> (NXColor) <i>color</i>	Sets the Window's background color to <i>color</i>
– (NXColor) <b>backgroundColor</b>	Returns the Window's background color
– <b>setBackgroundGray:</b> (float) <i>value</i>	Sets the gray <i>value</i> for Window's background gray
– (float) <b>backgroundGray</b>	Returns the Window's background gray

## Window Device Attributes

– <b>setBackingType:</b> (int) <i>backing</i>	Sets the type of window device backing
– (int) <b>backingType</b>	Returns the window device backing type
– <b>setOneShot:</b> (BOOL) <i>flag</i>	Sets whether memory for the window should be freed

- **(BOOL)isOneShot**  
Returns whether memory for the window is freed
- **setFreeWhenClosed:(BOOL)*flag***  
Sets whether closing the Window also frees it

## Graphics State Objects

- **(int)gState**  
Returns the graphics state object for the Window

## The Miniwindow

- **counterpart**  
Returns the receiver's miniwindow/Window companion
- **setMiniwindowIcon:(const char \*)*name***  
Sets the icon that's displayed in the miniwindow
- **(const char \*)miniwindowIcon**  
Returns the icon that's displayed in the miniwindow
- **setMiniwindowImage:*image***  
Sets the image that's displayed in the miniwindow
- **(NXImage \*)miniwindowImage**  
Returns the image that's displayed in the miniwindow
- **setMiniwindowTitle:(const char \*)*title***  
Sets the title that's displayed in the miniwindow
- **(const char \*)miniwindowTitle**  
Sets the title that's displayed in the miniwindow

## The Field Editor

- **getFieldEditor:(BOOL)*flag* for:*anObject***  
Returns the Window's field editor
- **endEditingFor:*anObject***  
Ends the field editor's editing assignment

## Window Status

- **makeKeyWindow**  
Makes the receiver the key window
- **makeKeyAndOrderFront:*sender***  
Makes Window the key window and brings it forward
- **becomeKeyWindow**  
Records Window's new status as the key window
- **(BOOL)isKeyWindow**  
Returns whether Window is the key window
- **resignKeyWindow**  
Records that Window no longer is the key window
- **(BOOL)canBecomeKeyWindow**  
Returns whether Window can be the key window
- **becomeMainWindow**  
Records Window's new status as the main window
- **(BOOL)isMainWindow**  
Returns whether Window is the main window
- **resignMainWindow**  
Records that Window no longer is the main window
- **(BOOL)canBecomeMainWindow**  
Returns whether Window can be the main window

## Moving and Resizing the Window

- **moveTo:(NXCoord)*x* :(NXCoord)*y***  
Moves the Window to (x, y)
- **moveTo:(NXCoord)*x* :(NXCoord)*y* screen:(const NXScreen \*)*aScreen***  
Moves the Window to (x, y) relative to *aScreen*

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

## Reordering the Window

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

## Converting Coordinates

<b>– convertBaseToScreen:(NXPoint *)<i>aPoint</i></b>	Converts point from base to screen coordinates
<b>– convertScreenToBase:(NXPoint *)<i>aPoint</i></b>	Converts point from screen to base coordinates

## Managing the Display

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

## Screens and Window Depths

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

## Cursor Management

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

## Handling User Actions and Events

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

## Setting the Event Mask

– <b>(int)setEventMask:(int)<i>newMask</i></b>	Sets the Window's event mask to <i>newMask</i>
– <b>(int)addToEventMask:(int)<i>newEvents</i></b>	Adds <i>newEvents</i> to the event mask
– <b>(int)removeFromEventMask:(int)<i>oldEvents</i></b>	Removes <i>oldEvents</i> from the event mask
– <b>(int)eventMask</b>	Returns the Window's event mask

## Aiding Event Handling

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

## Dragging

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

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

## Archiving

– <b>read:</b> (NXTypedStream *) <i>stream</i>	Reads the Window from the typed stream
– <b>write:</b> (NXTypedStream *) <i>stream</i>	Writes the Window to the typed stream
– <b>awake</b>	Redisplays and reinitializes the Window

## Assigning a Delegate

– <b>setDelegate:</b> <i>anObject</i>	Makes <i>anObject</i> the Window's delegate
– <b>delegate</b>	Returns the Window's delegate

## Implemented by the Delegate

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

– <b>windowDidResize:</b> <i>sender</i>	Notifies delegate that the Window was resized
– <b>windowDidExpose:</b> <i>sender</i>	Notifies delegate that the Window was exposed
– <b>windowWillMove:</b> <i>sender</i>	Notifies delegate that the Window will move
– <b>windowDidMove:</b> <i>sender</i>	Notifies delegate that the Window did move
– <b>windowDidChangeScreen:</b> <i>sender</i>	Notifies delegate that the Window changed screens
– <b>windowDidBecomeKey:</b> <i>sender</i>	Notifies delegate that the Window is the key window
– <b>windowDidResignKey:</b> <i>sender</i>	Notifies delegate that the Window isn't the key window
– <b>windowDidBecomeMain:</b> <i>sender</i>	Notifies delegate that the Window is the main window
– <b>windowDidResignMain:</b> <i>sender</i>	Notifies delegate that the Window isn't the main window
– <b>windowWillMiniaturize:</b> <i>sender</i> <i>toMin&gt;window:</i> <i>miniwindow</i>	Notifies delegate that the Window will miniaturized
– <b>windowDidMiniaturize:</b> <i>sender</i>	Notifies delegate that the Window was miniaturized
– <b>windowDidDeminiaturize:</b> <i>sender</i>	Notifies delegate that the Window was restored to screen
– <b>windowDidUpdate:</b> <i>sender</i>	Notifies delegate that the Window was updated

# 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) <i>isLocal</i>	Returns a mask giving the operations that can be performed on the dragged image's data
---	--

## Informing the Source

– draggedImage:(NXImage *) <i>image</i> beganAt:(NXPoint *) <i>screenPoint</i>	Invoked when the dragged image is displayed but before it starts following the mouse
– draggedImage:(NXImage *) <i>image</i> endedAt:(NXPoint *) <i>screenPoint</i> deposited:(BOOL) <i>didDeposit</i>	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

---

## **NXRTFErrorHandler**

**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

# **NXServicesRequests**

**(informal protocol)**

**Category Of:** Object

## Pasteboard Read/Write

# 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	<b>NXDrawGroove</b> (const NXRect * <i>aRect</i> , const NXRect * <i>clipRect</i> )
void	<b>NXDrawWhiteBezel</b> (const NXRect * <i>aRect</i> , const NXRect * <i>clipRect</i> )
NXRect *	<b>NXDrawTiledRects</b> (NXRect * <i>boundsRect</i> , const NXRect * <i>clipRect</i> , const int * <i>sides</i> , const float * <i>grays</i> , int <i>count</i> )
void	<b>NXFrameRect</b> (const NXRect * <i>aRect</i> )
void	<b>NXFrameRectWithWidth</b> (const NXRect * <i>aRect</i> , NXCoord <i>frameWidth</i> )

#### Query an NXRect structure:

NXCoord	<b>NX_X</b> (NXRect * <i>aRect</i> )
NXCoord	<b>NX_Y</b> (NXRect * <i>aRect</i> )
NXCoord	<b>NX_WIDTH</b> (NXRect * <i>aRect</i> )
NXCoord	<b>NX_HEIGHT</b> (NXRect * <i>aRect</i> )
NXCoord	<b>NX_MAXX</b> (NXRect * <i>aRect</i> )
NXCoord	<b>NX_MAXY</b> (NXRect * <i>aRect</i> )
NXCoord	<b>NX_MIDX</b> (NXRect * <i>aRect</i> )
NXCoord	<b>NX_MIDY</b> (NXRect * <i>aRect</i> )

## Color Functions

#### Specify a color value:

NXColor	<b>NXConvertRGBAToColor</b> (float <i>red</i> , float <i>green</i> , float <i>blue</i> , float <i>alpha</i> )
NXColor	<b>NXConvertCMYKAToColor</b> (float <i>cyan</i> , float <i>magenta</i> , float <i>yellow</i> , float <i>black</i> , float <i>alpha</i> )
NXColor	<b>NXConvertHSBAToColor</b> (float <i>hue</i> , float <i>saturation</i> , float <i>brightness</i> , float <i>alpha</i> )
NXColor	<b>NXConvertGrayAlphaToColor</b> (float <i>gray</i> , float <i>alpha</i> )
NXColor	<b>NXConvertRGBToColor</b> (float <i>red</i> , float <i>green</i> , float <i>blue</i> )
NXColor	<b>NXConvertCMYKToColor</b> (float <i>cyan</i> , float <i>magenta</i> , float <i>yellow</i> , float <i>black</i> )
NXColor	<b>NXConvertHSBTоГColor</b> (float <i>hue</i> , float <i>saturation</i> , float <i>brightness</i> )
NXColor	<b>NXConvertGrayToColor</b> (float <i>gray</i> )

#### Convert a color value to its standard components:

void	<b>NXConvertColorToRGBA</b> (NXColor <i>color</i> , float * <i>red</i> , float * <i>green</i> , float * <i>blue</i> , float * <i>alpha</i> )
void	<b>NXConvertColorToCMYKA</b> (NXColor <i>color</i> , float * <i>cyan</i> , float * <i>magenta</i> , float * <i>yellow</i> , float * <i>black</i> , float * <i>alpha</i> )
void	<b>NXConvertColorToHSBA</b> (NXColor <i>color</i> , float * <i>hue</i> , float * <i>saturation</i> , float * <i>brightness</i> , float * <i>alpha</i> )
void	<b>NXConvertColorToGrayAlpha</b> (NXColor <i>color</i> , float * <i>gray</i> , float * <i>alpha</i> )
void	<b>NXConvertColorToRGB</b> (NXColor <i>color</i> , float * <i>red</i> , float * <i>green</i> , float * <i>blue</i> )

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

**Modify a color by changing one of its components:**

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

**Isolate one component of a color:**

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

**Test whether two colors are the same:**

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

**Get information about color space and window depth:**

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

int	<b>NXNumberOfColorComponents(NXColorSpace <i>space</i>)</b>
BOOL	<b>NXGetBestDepth(NXWindowDepth *<i>depth</i>, int <i>numColors</i>, int <i>bps</i>)</b>

**Read and write a color from a typed stream:**

NXColor	<b>NXReadColor(NXTypedStream *<i>stream</i>)</b>
void	<b>NXWriteColor(NXTypedStream *<i>stream</i>, NXColor <i>color</i>)</b>

**Read and write a color from a pasteboard:**

NXColor	<b>NXReadColorFromPasteboard(id <i>pasteboard</i>)</b>
void	<b>NXWriteColorToPasteboard(id <i>pasteboard</i>, NXColor <i>color</i>)</b>

**Set the current color:**

void	<b>NXSetColor(NXColor <i>color</i>)</b>
------	---

**Reading the color at a screen position:**

NXColor	<b>NXReadPixel(const NXPoint *<i>location</i>)</b>
---------	--

**Associate named colors with their color lists**

const char *	<b>NXColorListName (NXColor <i>color</i>)</b>
const char *	<b>NXColorName (NXColor <i>color</i>)</b>
BOOL	<b>NXFindColorNamed (const char *<i>colorList</i>, const char *<i>colorName</i>, NXColor *<i>color</i>)</b>

## Text Functions

**Filter characters entered into Text object:**

unsigned short	<b>NXFieldFilter(unsigned short <i>theChar</i>, int <i>flags</i>, unsigned short <i>charSet</i>)</b>
unsigned short	<b>NXEditorFilter(unsigned short <i>theChar</i>, int <i>flags</i>, unsigned short <i>charSet</i>)</b>

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

int	<b>NXScanALine(id <i>self</i>, NXLayInfo *<i>layInfo</i>)</b>
int	<b>NXDrawALine(id <i>self</i>, NXLayInfo *<i>layInfo</i>)</b>

### **Calculate font ascender, descender, and line height (in Text object):**

```
void NXTextFontInfo(id fontId, NXCoord *ascender, NXCoord *descender,  
NXCoord *lineHeight)
```

### **Access Text object's word tables:**

```
void NXReadWordTable(NXZone *zone, NXStream *stream,  
unsigned char **preSelSmart,  
unsigned char **postSelSmart,  
unsigned char **charCategories, NXFSM **wrapBreaks,  
int *wrapBreaksCount, NXFSM **clickBreaks,  
int *clickBreaksCount, BOOL *charWrap)  
void NXWriteWordTable(NXStream *stream, const unsigned char *preSelSmart,  
const unsigned char *postSelSmart,  
const unsigned char *charCategories,  
const NXFSM *wrapBreaks, int wrapBreaksCount,  
const NXFSM *clickBreaks, int clickBreaksCount,  
BOOL charWrap)
```

### **Provide table-driven string ordering service:**

```
int NXOrderStrings(const unsigned char *string1, const unsigned char *string2,  
BOOL caseSensitive, int length, NXStringOrderTable *table)  
NXStringOrderTable * NXDefaultStringOrderTable(void)
```

## **Imaging Functions**

### **Copy an image:**

```
void NXCopyBits(int gstate, const NXRect *aRect, const NXPoint *aPoint)  
void NXCopyBitmapFromGstate(int gstate, const NXRect *srcRect,  
const NXRect *destRect)
```

### **Render and read bitmap images:**

```
void NXDrawBitmap(const NXRect *rect, int pixelsWide, int pixelsHigh, int bps,  
int spp, int config, int mask, const void *data1, const void *data2,  
const void *data3, const void *data4, const void *data5)  
void NXReadBitmap(const NXRect *rect, int pixelsWide, int pixelsHigh, int bps,  
int spp, int config, int mask, void *data1, void *data2,  
void *data3, void *data4, void *data5)
```

```
void           NXSizeBitmap(const NXRect *rect, int *size, int *pixelsWide, int *pixelsHigh,
                           int *bps, int *spp, int *config, int *mask)
```

## Object Management Functions

### Refer to objects by name:

id	NXGetNamedObject(const char *name, id owner)
const char *	NXGetObjectNames(id theObject)
int	NXNameObject(const char *name, id theObject, id owner)
int	NXUnnameObject(const char *name, id owner)

### Get information about an application's windows:

void	NXCountWindows(int *count)
void	NXWindowList(int size, int list[])

### Convert local and global window numbers:

void	NXConvertWinNumToGlobal(int winNum, unsigned int *globalNum)
void	NXConvertGlobalToWinNum(int globalNum, unsigned int *winNum)

### Set up a pop-up list:

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

### Create or free an attention panel:

int	NXRunAlertPanel(const char *title, const char *msg, const char *defaultButton, const char *alternateButton, const char *otherButton, ...)
int	NXRunLocalizedAlertPanel(const char *table, const char *title, const char *msg, const char *defaultButton, const char *alternateButton, const char *otherButton, ...)
id	NXGetAlertPanel(const char *title, const char *msg, const char *firstButton, const char *alternateButton, const char *otherButton, ...)
void	NXFreeAlertPanel(id alertPanel)

## 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 *         NXCopyInputData(int parameter)
char *         NXCopyOutputData(int parameter)
```

### **Get send rights to an application port:**

port_t	<b>NXPortFromName(const char *name, const char *host)</b>
port_t	<b>NXPortNameLookup(const char *name, const char *host)</b>

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

<b>NXRemoteMethod *</b>	<b>NXRemoteMethodFromSel(SEL aSelector, NXRemoteMethod *methods)</b>
<b>id</b>	<b>NXResponsibleDelegate(id aListener, SEL aSelector)</b>

## **Services Menu Functions**

### **Determine whether an item is included in Services menus:**

<b>int</b>	<b>NXSetServicesMenuItemEnabled(const char *item, BOOL flag)</b>
<b>BOOL</b>	<b>NXIsServicesMenuItemEnabled(const char *item)</b>

### **Programmatically invoke a service:**

<b>BOOL</b>	<b>NXPerformService(const char *item, Pasteboard *pboard)</b>
-------------	---

### **Force Services menu to update based on new services:**

<b>void</b>	<b>NXUpdateDynamicServices(void)</b>
-------------	--------------------------------------

## **Event Function**

### **Access event record in event queue:**

<b>NXEvent *</b>	<b>NXGetOrPeekEvent(DPSContext context, NXEvent *anEvent, int mask, double timeout, int threshold, int peek)</b>
------------------	--

## **Memory Allocation Functions**

### **Macros to allocate memory:**

<b>type-name *</b>	<b>NX_MALLOC(type-name *var, type-name, int num)</b>
<b>type-name *</b>	<b>NX_REALLOC(type-name *var, type-name, int num)</b>
<b>void</b>	<b>NX_FREE(void *pointer)</b>

### **Allocate a variable-sized array:**

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

### **Macros to allocate zone memory:**

<code>type-name *</code>	<code>NX_ZONEMALLOC(NXZone *zone, type-name *var, type-name, int num)</code>
<code>type-name *</code>	<code>NX_ZONEREALLOC(NXZone *zone, type-name *var, type-name, int num)</code>

## **Other Application Kit Functions**

### **Get user's home directory and name:**

<code>const char *</code>	<code>NXHomeDirectory(void)</code>
<code>const char *</code>	<code>NXUserName(void)</code>

### **Synchronize the application with the Window Server:**

<code>void</code>	<code>NXPing(void)</code>
-------------------	---------------------------

### **Find dimensions of specified paper type:**

<code>const NXSize *</code>	<code>NXFindPaperSize(const char *paperName)</code>
-----------------------------	---

### **Play the system beep:**

<code>void</code>	<code>NXBeep(void)</code>
-------------------	---------------------------

### **Set up timer events:**

<code>NXTrackingTimer *</code>	<code>NXBeginTimer(NXTrackingTimer *timer, double delay, double period)</code>
<code>void</code>	<code>NXEndTimer(NXTrackingTimer *timer)</code>

**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,  
 NXLayInfo *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
<b>Kit-Defined Subtypes</b>	
NX_WINEXPOSED	0
NX_APPACT	1
NX_APPDEACT	2
NX_WINMOVED	4
NX_SCREENCHANGED	8
<b>System-Defined Subtype</b>	
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\_CELLEDITABLE  
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\_ANYTYPENX\_INNTYPENX\_POSINTTYPENX\_FLOATTYPENX\_POSFLOATTYPENX\_DOUBLETYPENX\_POSDOUBLETYPE

### **Cell sendActionOn: Flag**

NX\_PERIODICMASK

### **Cell Types**

NX\_NULLCELLNX\_TEXTCELLNX\_ICONCELL

### **Color Panel Modes**

NX\_GRAYMODENX\_RGBMODENX\_CMYKMODENX\_HSBMODENX\_CUSTOMPALETTEMODENX\_CUSTOMCOLORMODENX\_BEGINMODE

### **Color Panel Mode Masks**

NX\_GRAYMODEMASKNX\_RGBMODEMASKNX\_CMYKMODEMASKNX\_HSBMODEMASKNX\_CUSTOMPALETTEMODEMASKNX\_LISTMODEMASKNX\_WHEELMODEMASKNX\_ALLMODESMASK

**Color Picker Insertion**

<b>Order Constants</b>	<b>Value</b>
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_MODALRESPTHRESHOLD	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_REMOVEVTRAIT	7

## **Font Matrix Constants**

NX\_IDENTITYMATRIX  
NX\_FLIPPEDMATRIX

<b>Font Trait Constants</b>	<b>Value</b>
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\_FPREVERTBUTTON  
NX\_FPPREVIEWBUTTON  
NX\_FPSETBUTTON  
NX\_FPSIZETITLE  
NX\_FPCURRENTFIELD

<b>Gray Shades</b>	<b>Value</b>
NX_WHITE	1.0
NX_LTGRAY	2.0/3.0
NX_DKGRAY	1.0/3.0
NX_BLACK	0.0

<b>Icon and Token Window Dimensions</b>	<b>Value</b>
NX_ICONWIDTH	48.0
NX_ICONHEIGHT	48.0
NX_TOKENWIDTH	64.0
NX_TOKENHEIGHT	64.0

## **Image Representation Device Matching Constant**

**NX\_MATCHESDEVICE**

<b>Journaling Flags</b>	<b>Value</b>
NX_JOURNALFLAG	31
NX_JOURNALFLAGMASK	(1 << NX_JOURNALFLAG)
<b>Journaling Listener</b>	
<b>Name</b>	<b>Value</b>
NX_JOURNALREQUEST	"NXJournalerRequest"
<b>Journaling Recording Device</b>	<b>Value</b>
NX_CODEC	0
NX_DSP	1
<b>Journaling Status</b>	<b>Value</b>
NX_STOPPED	0
NX_PLAYING	1
NX_RECORDING	2
NX_NONABORTABLEFLAG	31
NX_NONABORTABLEMASK	(1 << NX_NONABORTABLEFLAG)
<b>Journaling Subevents</b>	<b>Value</b>
NX_WINDRAGGED	0
NX_MOUSELOCATION	1
NX_LASTJRNEVENT	2
<b>Journaling Window Encodings</b>	<b>Value</b>
NX_KEYWINDOW	(-1)
NX_MAINWINDOW	(-2)
NX_MAINMENU	(-3)
NX_MOUSEDOWNWINDOW	(-4)
NX_APPICONWINDOW	(-5)
NX_UNKNOWNWINDOW	(-6)

<b>Listener Maximum Message Size</b>	<b>Value</b>
NX_MAXMESSAGE	(2048–sizeof(msg_header_t)– sizeof(msg_type_t)–sizeof(int)– sizeof(msg_type_t)–8)
<b>Listener Maximum Parameters</b>	<b>Value</b>
NX_MAXMSGPARAMS	20
<b>Listener Position Types</b>	<b>Value</b>
NX_TEXTPOSTYPE	0
NX_REGEXPRPOSTYPE	1
NX_LINENUMPOSTYPE	2
NX_CHARNUMPOSTYPE	3
NX_APPPOSTYPE	4
<b>Listener Reserved Message Numbers</b>	<b>Value</b>
NX_SELECTORPMSG	35555
NX_SELECTORFMSG	35556
NX_RESPONSEMSG	35557
NX_ACKNOWLEDGE	35558
<b>Listener RPC Error Return Values</b>	
<b>Error</b>	<b>Value</b>
NX_INCORRECTMESSAGE	-20000
<b>Listener Timeout Default</b>	<b>Value</b>
NX_SENDTIMEOUT	10000
NX_RCVTIMEOUT	10000
<b>Mach Executable File Segment Names for Images</b>	<b>Segment Name</b>
NX_EPSSEGMENT	“__EPS”
NX_TIFFSEGMENT	“__TIFF”
NX_ICONSEGMENT	“__ICON”

## **Matrix Selection Mode Constants**

NX\_RADIOMODE  
NX\_HIGHLIGHTMODE  
NX\_LISTMODE  
NX\_TRACKMODE

<b>Modal Session Return Constants</b>	<b>Value</b>
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

<b>Open Panel Tag Constants</b>	<b>Value</b>
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

<b>Panel Button Tags</b>	<b>Value</b>
NX_OKTAG	1
NX_CANCELTAG	0

<b>Panel Return Constants</b>	<b>Value</b>
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**

<b>Constants</b>	<b>Value</b>
NX_SPICONBUTTON	150
NX_SPTITLEFIELD	151
NX_SPBROWSER	152
NX_SPCANCELBUTTON	NX_CANCELTAG
NX_SPOKBUTTON	NX_OKTAG
NX_SPFORM	155

### **Scroller Arrow Positions**

<b>Scroller Arrow Positions</b>	<b>Value</b>
NX_SCROLLARROWSMAXEND	0
NX_SCROLLARROWSMINEND	1
NX_SCROLLARROWSNONE	2

### **Scroller Part**

<b>Identification Constants</b>	<b>Value</b>
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**

<b>Scroller Usable Parts</b>	<b>Value</b>
NX_SCROLLERNOPARTS	0
NX_SCROLLEROONLYARROWS	1
NX_SCROLLERAALLPARTS	2

### **Scroller Width and Height**

NX\_SCROLLWIDTH (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\_CCITTFA3  
NX\_TIFF\_COMPRESSION\_CCITTFA4  
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

<b>Window Tiers</b>	<b>Value</b>
NX_NORMALLEVEL	0
NX_FLOATINGLEVEL	3
NX_DOCKLEVEL	5
NX_SUBMENULEVEL	10
NX_MAINMENULEVEL	20

<b>Workspace Request Constants (File Operations)</b>	<b>Value</b>
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 NXCCClickTable
const int NXCCClickTableSize
```

### **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

– <b>init</b>	Initializes a new, default HashTable
– <b>initKeyDesc:(const char *)aKeyDesc</b>	Initializes a new HashTable
– <b>initKeyDesc:(const char *)aKeyDesc valueDesc:(const char *)aValueDesc</b>	Initializes a new HashTable
– <b>initKeyDesc:(const char *)aKeyDesc valueDesc:(const char *)aValueDesc capacity:(unsigned int)aCapacity</b>	Initializes a new HashTable
– <b>free</b>	Deallocates the HashTable
– <b>freeObjects</b>	Deallocates the HashTable's objects
– <b>freeKeys:(void (*)(void *))keyFunc values:(void (*)(void *))valueFunc</b>	Conditionally frees the HashTable's associations
– <b>empty</b>	Empties the HashTable but retains its capacity

#### Copying a HashTable

– <b>copyFromZone:(NXZone *) zone</b>	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

- `(NXHashTable)initState` Begins process of iteration through the HashTable
- `(BOOL)nextState:(NXHashTable *)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*

– <b>addObject:</b> <i>anObject</i>	Adds <i>anObject</i> at the end of the List
– <b>removeObjectAt:</b> (unsigned int) <i>index</i>	Removes the object located at <i>index</i>
– <b>removeLastObject</b>	Removes the object at the end of the List
– <b>replaceObjectAt:</b> (unsigned int) <i>index</i> <b>with:</b> <i>newObject</i>	Puts <i>newObject</i> in place of the object at <i>index</i>
– <b>objectAt:</b> (unsigned int) <i>index</i>	Returns the object at <i>index</i>
– <b>lastObject</b>	Returns the object at the end of the List
– (unsigned int) <b>count</b>	Returns the number of objects in the List

## Manipulating Objects by id

– <b>addObject:</b> <i>anObject</i>	Adds <i>anObject</i> at the end of the List
– <b>addObjectIfAbsent:</b> <i>anObject</i>	Adds <i>anObject</i> to the List, if it's not already in the List
– <b>removeObject:</b> <i>anObject</i>	Removes first occurrence of <i>anObject</i> from the List
– <b>replaceObject:</b> <i>anObject</i> <b>with:</b> <i>newObject</i>	Puts <i>newObject</i> in the List in place of <i>anObject</i>
– (unsigned int) <b>indexOf:</b> <i>anObject</i>	Returns the index of <i>anObject</i>

## Comparing and Combining Lists

– (BOOL) <b>isEqual:</b> <i>anObject</i>	Returns whether the two Lists have the same contents
– <b>appendList:</b> (List *) <i>otherList</i>	Adds the objects in <i>otherList</i> to the receiving List

## Emptying a List

– <b>empty</b>	Empties the List of its contents, but doesn't free the objects
– <b>freeObjects</b>	Deallocates all the objects in the List

## Sending Messages to the Objects

– <b>makeObjectsPerform:</b> (SEL) <i>aSelector</i>	Sends an <i>aSelector</i> message to each object in the List
– <b>makeObjectsPerform:</b> (SEL) <i>aSelector</i> <b>with:</b> <i>anObject</i>	Sends <i>aSelector</i> message with an argument to each object

## Managing the Storage Capacity

– (unsigned int) <b>capacity</b>	Returns the number of objects the List can store
– <b>setAvailableCapacity:</b> (unsigned int) <i>numSlots</i>	Sets the capacity of the List to at least <i>numSlots</i> objects

## Archiving

– <b>write:</b> (NXTypedStream *) <i>stream</i>	Writes the List to the typed stream <i>stream</i>
– <b>read:</b> (NXTypedStream *) <i>stream</i>	Reads the List from the typed stream <i>stream</i>

---

## NXBundle

**Inherits From:** Object

### Initializing a New NXBundle object

– **initForDirectory:(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

- |   |   |
|---|---|
| – <b>setVersion:(int)<i>version</i></b> | Sets the version that resources must match    |
| – <b>(int)version</b>                   | Returns the version that resources must match |

---

## NXStringTable

**Inherits From:** HashTable : Object

### Initializing and Freeing an NXStringTable

- |               |                                 |
|---------------|---------------------------------|
| – <b>init</b> | Initializes a new NXStringTable |
| – <b>free</b> | Deallocates the NXStringTable   |

### Querying an NXStringTable

- |   |  |
|---|--|
| – <b>(const char *)valueForKey:(const char *)<i>aString</i></b> | Returns the value that corresponds to <i>aString</i> |
|---|--|

### Writing Elements

- |   |  |
|---|--|
| – <b>readFromFile:(const char *)<i>fileName</i></b> | Reads the keys and values from <i>fileName</i> |
| – <b>writeToFile:(const char *)<i>fileName</i></b>  | Writes the keys and values to <i>fileName</i>  |
| – <b>readFromStream:(NXStream *)<i>stream</i></b>   | Reads the keys and values from <i>stream</i>   |
| – <b>writeToStream:(NXStream *)<i>stream</i></b>    | Writes the keys and values to <i>stream</i>    |

---

## Storage

**Inherits From:** Object

### Initializing a New Storage Object

- |               |                                |
|---------------|--------------------------------|
| – <b>init</b> | 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	NXIsAlpha(unsigned int c)
int	NXIsUpper(unsigned int c)
int	NXIsLower(unsigned int c)
int	NXIsDigit(unsigned int c)
int	NXIsXDigit(unsigned int c)
int	NXIsAlNum(unsigned int c)
int	NXIsSpace(unsigned int c)
int	NXIsPunct(unsigned int c)
int	NXIsPrint(unsigned int c)
int	NXIsGraph(unsigned int c)
int	NXIsCntrl(unsigned int c)
int	NXIsAscii(unsigned int c)

### Convert NeXTSTEP-Encoded Characters:

unsigned char *	NXToAscii(unsigned int c)
int	NXToLower(unsigned int c)
int	NXToUpper(unsigned int c)

## Defaults System Functions

### Set or Read Default Parameters:

int	NXRegisterDefaults(const char *owner, const NXDefaultsVector vector)
const char *	NXGetDefaultValue(const char *owner, const char *name)
const char *	NXReadDefault(const char *owner, const char *name)
int	NXRemoveDefault(const char *owner, const char *name)
int	NXSetDefault(const char *owner, const char *name, const char *value)
const char *	NXUpdateDefault(const char *owner, const char *name)
void	NXUpdateDefaults(void)
int	NXWriteDefault(const char *owner, const char *name, const char *value)
int	NXWriteDefaults(const char *owner, NXDefaultsVector vector)
const char *	NXSetDefaultsUser(const char *newUser)

## Error-Handling Functions

### Macros to Raise an Exception:

void	<b>NX_RAISE</b> (int <i>code</i> , const void * <i>data1</i> , const void * <i>data2</i> )
void	<b>NX_RERAISE</b> (void)
val	<b>NX_VALRETURN</b> (val)
void	<b>NX_VOIDRETURN</b>

### Set and Return an Exception Raiser:

void	<b>NXDefaultExceptionRaiser</b> (int <i>code</i> , const void * <i>data1</i> , const void * <i>data2</i> )
void	<b>NXSetExceptionRaiser</b> (NXExceptionRaiser * <i>procedure</i> )
	<b>NXExceptionRaiser *</b> <b>NXGetExceptionRaiser</b> (void)

### Macros to Handle Uncaught Exceptions:

void	<b>NXSetUncaughtExceptionHandler</b> (NXUncaughtExceptionHandler * <i>proc</i> )
	<b>NXUncaughtExceptionHandler *</b> <b>NXGetUncaughtExceptionHandler</b> (void)

### Manage the Error Data Buffer:

void	<b>NXAllocErrorHandler</b> (int <i>size</i> , void ** <i>data</i> )
void	<b>NXResetErrorHandler</b> (void)

## Stream Functions

### Manipulate a Memory Stream:

NXStream *	<b>NXOpenMemory</b> (const char * <i>address</i> , int <i>size</i> , int <i>mode</i> )
NXStream *	<b>NXMapFile</b> (const char * <i>pathName</i> , int <i>mode</i> )
int	<b>NXSaveToFile</b> (NXStream * <i>stream</i> , const char * <i>name</i> )
void	<b>NXCloseMemory</b> (NXStream * <i>stream</i> , int <i>option</i> )
void	<b>NXGetMemoryBuffer</b> (NXStream * <i>stream</i> , char ** <i>streambuf</i> , int * <i>len</i> , int * <i>maxLen</i> )

### Open a File Stream or a Mach Port Stream:

NXStream *	<b>NXOpenFile</b> (int <i>fd</i> , int <i>mode</i> )
NXStream *	<b>NXOpenPort</b> (port_t <i>port</i> , int <i>mode</i> )

**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* NXOpenTypedStreamFromFile(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 *	<code>NXCreateHashTable(NXHashTablePrototype <i>prototype</i>, unsigned <i>capacity</i>, const void *<i>info</i>)</code>
NXHashTable *	<code>NXCreateHashTableFromZone(NXHashTablePrototype <i>prototype</i>, unsigned <i>capacity</i>, const void *<i>info</i>, NXZone *<i>zone</i>)</code>
void	<code>NXFreeHashTable(NXHashTable *<i>table</i>)</code>
void	<code>NXEmptyHashTable(NXHashTable *<i>table</i>)</code>
void	<code>NXResetHashTable(NXHashTable *<i>table</i>)</code>
NXHashTable *	<code>NXCopyHashTable(NXHashTable *<i>table</i>)</code>
BOOL	<code>NXCompareHashTables(NXHashTable *<i>table1</i>, NXHashTable *<i>table2</i>)</code>
unsigned	<code>NXPtrHash(const void *<i>info</i>, const void *<i>data</i>)</code>
unsigned	<code>NXStrHash(const void *<i>info</i>, const void *<i>data</i>)</code>
int	<code>NXPtrIsEqual(const void *<i>info</i>, const void *<i>data1</i>, const void *<i>data2</i>)</code>
int	<code>NXStrIsEqual(const void *<i>info</i>, const void *<i>data1</i>, const void *<i>data2</i>)</code>
void	<code>NXNoEffectFree(const void *<i>info</i>, void *<i>data</i>)</code>
void	<code>NXReallyFree(const void *<i>info</i>, void *<i>data</i>)</code>

## Manipulate the Elements of a Hash Table:

void *	<code>NXHashInsert(NXHashTable *<i>table</i>, const void *<i>data</i>)</code>
void *	<code>NXHashInsertIfAbsent(NXHashTable *<i>table</i>, const void *<i>data</i>)</code>
int	<code>NXHashMember(NXHashTable *<i>table</i>, const void *<i>data</i>)</code>
void *	<code>NXHashGet(NXHashTable *<i>table</i>, const void *<i>data</i>)</code>
void *	<code>NXHashRemove(NXHashTable *<i>table</i>, const void *<i>data</i>)</code>
unsigned	<code>NXCountHashTable(NXHashTable *<i>table</i>)</code>
NXHashTable	<code>NXInitHashTable(NXHashTable *<i>table</i>)</code>
int	<code>NXNextHashTable(NXHashTable *<i>table</i>, NXHashTable *<i>state</i>, void **<i>data</i>)</code>

## 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;
```

### NXHashTable

```
typedef struct {
    int i;
    int j;
} NXHashTable;
```

### 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

<b>NXStringTable Constants</b>	<b>Value</b>
MAX_NXSTRINGTABLE_LENGTH	1024

<b>Zone Constants</b>	<b>Value</b>
NX_NOZONE	(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

- **initFetchGroup:*aFetchGroup***  
  **expression:*anExpr***  
  **destination:*aDest***
- Initializes; associates *aFetchGroup* with *aDest*

#### Linking Expression and View

- **destination**
  - **fetchGroup**
  - **expression**
  - **setDestination:*newDestination***
- The association's user interface object  
The DBFetchGroup that owns the association  
The DBExpression that selects the properties displayed  
Sets the user association's user interface object

#### Methods to be Re-Defined in a Subclass of Association

- **contentsDidChange**
  - **currentRecordDidDelete**
- Notice to redisplay because the value changed  
Notice to redisplay because a record was deleted

<b>– endEditing</b>	Notification that editing the destination must end
<b>– getValue:value</b>	Gets an object containing the association's data
<b>– (unsigned int)selectedRowAfter:(unsigned int)previousRow</b>	The index of the next selected row
<b>– selectionDidChange</b>	Notice that the user has changed the selection
<b>– setValue:value</b>	Sets the association's data
<b>– validateEditing</b>	Notice to validate changes the user has made

### Methods to be Defined in the Destination of a Custom Association

<b>– association:association     getValue:(DBValue *)value</b>	Gets an object containing the association's data
<b>– association:association     setValue:(DBValue *)value</b>	Sets the association's data
<b>– associationContentsDidChange:association</b>	Notice to redisplay because the value changed
<b>– associationCurrentRecordDidDelete:association</b>	Notice to redisplay because a record was deleted
<b>– associationSelectionDidChange:association</b>	Notice that the user has changed the selection

## DBBINDER

**Inherits From:** Object

**Conforms To:** DBCursorPositioning

### Initializing

<b>– init</b>	Initializes a new DBBinder instance
<b>– initForDatabase:<i>aDBDatabase</i>     withProperties:(List *)<i>propertyList</i>     andQualifier:(DBQualifier *)<i>aDBQualifier</i></b>	Initializes database, properties, qualifier; frees DBBinder
<b>– free</b>	Frees the space allocated to a DBBinder

### Connecting to a Database

<b>– (DBDatabase *)database</b>	The DBBimder's DBDatabase
<b>– setDatabase:(DBDatabase *)<i>aDatabase</i></b>	Sets the DBBimder's DBDatabase

## Managing Properties

- **(List \*)getProperties:(List \*)*aList***
- **(List \*)setProperties:(List \*)*aList***
- **addProperty:*anObject***
- **removePropertyAt:(unsigned int)*index***

Gets and returns the DBBinder's properties  
Sets and returns the DBBinder's properties  
Adds an object to the list of properties  
Deletes one of the objects from the list of properties

## Managing the Qualifier

- **(DBQualifier \*)qualifier**
- **setQualifier:(DBQualifier \*)*aQualifier***

The DBBinder's qualifier  
Sets the DBBinder's qualifier

## Managing the Container

- **(id <DBContainers>)container**
- **setContainer:(id <DBContainers>)*anObject***
- **setFlushEnabled:(BOOL)*flag***
- **(BOOL)isFlushEnabled**
- **setFreeObjectsOnFlush:(BOOL)*flag***
- **(BOOL)areObjectsFreedOnFlush**

The DBBinder's container  
Sets the DBBinder's container  
Sets whether flushing the DBBinder is permitted  
Reports whether flushing is enabled; default YES  
Sets whether the DBBinder is freed when flushed  
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* withProperty:(id <DBProperties>)*aProperty*** Makes *variableName* report the value of *aProperty*
- **associateRecordSelectors:(SEL)*set* :(SEL)*get* withProperty:(id <DBProperties>)*aProperty*** Sets the selectors for storing and retrieving *aProperty*
- **(DBValue \*)valueForProperty:(id <DBProperties>)*aProperty*** *The value of aProperty for the current record*

## Ordering and Ignoring Records

- **addRetrieveOrder:(DBRetrieveOrder)*aOrder* 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`
- `(BOOL)sharesContext`

Set whether this binder uses the shared cursor

YES if this binder uses the shared cursor

## Managing General Resources

- `reset`
- `(BOOL)flush`
- `(NXZone *)scratchZone`

Cancels any fetch, then flushes and frees objects

If enabled, empties the container

The zone the DBBinder is now using for allocations

## Appointing a Delegate

- `delegate`
- `setDelegate:anObject`

The object that receives notification messages

Sets the object to receive notification messages

## Archiving

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

Creates an instance by reading from a typed 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 in sorted 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 *) <i>aAdaptorName</i> | List of database available through <i>aAdaptorName</i> |

### Initializing an Instance

- |  |   |
|--|---|
| - initFromFile:(const char *) <i>aPath</i> | 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 *) <i>aString</i>                           | 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 *) <i>aUser</i> | The the model's login string for user <i>aUser</i> |

### Describing the Database Model

- |  |   |
|--|---|
| - (id <DBEntities>)entityNamed:(const char *) <i>aName</i> | Returns an object embodying entity <i>aName</i>     |
| - (List *)getEntities:(List *) <i>aList</i>                | 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

+ <b>findDatabaseNamed:</b> (const char *) <i>aName</i> <b>connect:</b> (BOOL) <i>flag</i>	Returns a DBDatabase instance, after loading model <i>aName</i> and (and, if flag is YES, connecting to it)
- (BOOL) <b>connect</b>	Opens a connection to database using the default login
- (BOOL) <b>connectUsingString:</b> (const unsigned char *) <i>aString</i>	Opens database connection to <i>database</i> by sending <i>aString</i>
- (BOOL) <b>connectUsingAdaptor:</b> (const char *) <i>aClassname</i> <b>andString:</b> (const unsigned char *) <i>aString</i>	Opens database connection via <i>anAdaptor</i> and <i>aString</i>
- (BOOL) <b>disconnect</b>	Disconnects from the database
- (BOOL) <b>disconnectUsingString:</b> (const unsigned char *) <i>aString</i>	Disconnects from the database by sending it <i>aString</i>
- (BOOL) <b>isConnected</b>	YES if there is a valid connection to the database
- (const unsigned char *) <b>connectionName</b>	The name assigned to the current connection

## Managing Transactions

- (BOOL) <b>beginTransaction</b>	YES if a new transaction is successfully started
- (BOOL) <b>rollbackTransaction</b>	YES if the current transaction is successfully rolled back
- (BOOL) <b>commitTransaction</b>	YES if the current transaction is successfully committed
- (BOOL) <b>isTransactionInProgress</b>	YES is a transaction is in progress
- (BOOL) <b>areTransactionsEnabled</b>	YES if transactions are enabled
- (BOOL) <b>enableTransactions:</b> (BOOL) <i>flag</i>	Enable/disable transaction; returns YES if successful

## Using a Delegate

- <b>delegate</b>	The object that receives notification messages
- <b>setDelegate:</b> <i>anObject</i>	Sets the object that receives notification messages

## Evaluating an Arbitrary String

- (BOOL) <b>evaluateString:</b> <i>aString</i>	Returns YES if the adaptor evaluates the string
--	---

## Controlling the User Interface

- (BOOL) <b>arePanelsEnabled</b>	YES if UI panels can respond to problems
- <b>setPanelsEnabled:</b> (BOOL) <i>flag</i>	Enable/disable response by UI panels

## Archiving

- <b>read:</b> (NXTypedStream *) <i>stream</i>	Creates an instance by reading from a typed stream
- <b>write:</b> (NXTypedStream *) <i>stream</i>	Archives an instance by writing to a typed stream

## Methods Implemented in the Delegate

- <b>db:aDatabase log:(const char*)fmt, ...</b>	Notification of log message sent by <i>aDatabase</i>
- (BOOL) <b>db:aDatabase notificationFrom:anObject message:(const unsigned char*)msg code:(int)n</b>	Notification of a message received from <i>aDatabase</i> ; Returns YES when the user acknowledges the notification.
- (BOOL) <b>db:aDatabase willEvaluateString:(const char *)aString usingBinder:(const char *)aBinder</b>	Notice that <i>aString</i> will be evaluated; YES lets it proceed
- <b>dbDidRollbackTransaction:sender</b>	Notification that database rolled back a transaction
- <b>dbDidCommitTransaction:sender</b>	Notification that database committed a transaction
- <b>dbWillCommitTransaction:sender</b>	Notification that database will commit a transaction
- <b>dbWillRollbackTransaction:sender</b>	Notification that database will roll back a transaction

---

## DBEditableFormatter

**Inherits From:** DBFormatter : Object

### Initializing

- <b>init</b>	Initializes a new instance
- <b>free</b>	Frees the space an instance formerly used

### Manipulating Font

- <b>font</b>	Returns the font used in the editable display
- <b>setFont:aFont</b>	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*
  - (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 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
- 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**
  - (BOOL)**endEditing**
- Forces an end to editing and discards changes; returns self  
Ends editing when user clicks elsewhere Returns YES if that becomes first responder

## Archiving

- **read:**(NXTypedStream \*)*stream*
  - **write:**(NXTypedStream \*)*stream*
  - **finishUnarchiving**
- Creates an instance by reading from a typed stream  
Archives an instance by writing to a typed stream  
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  
fromDescription:(const unsigned char \*)descriptionFormat, ...** Initializes for *anEntity*, with description string shown
- **initForEntity:(id <DBEntities>)anEntity  
fromName:(const char \*)aName  
usingType:(const char \*)aType** Initializes *anEntity*, from property *aName*,  
to have data type *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  
andDescription:(const unsigned char \*)descriptionFormat, ...** Sets *anEntity*, with the description string shown

### 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 *) <b>name</b>	Returns the name (set to match the attribute it fetches)
– <b>module</b>	The DBModule that owns the fetch group
– <b>entity</b>	The DBEntity for which the fetch group fetches
– <b>recordList</b>	The DBRecordList in which fetched records are stored
– (unsigned int) <b>currentRecord</b>	The index within the DBRecordList of the current record
– (unsigned int) <b>recordCount</b>	The number of records in the DBRecordList

## Controlling Current Selection

– <b>setAutoselect:(BOOL)flag</b>	If YES, fetch selects first row, delete selects next rowf
– (BOOL) <b>doesAutoSelect</b>	Returns flag set by – <b>setAutoselect:</b> ; default YES
– <b>setCurrentRecord:(unsigned int)newIndex</b>	Sets a position within the DBRecordList
– <b>clearCurrentRecord</b>	Deselects current record
– (unsigned int) <b>selectedRowAfter:(unsigned int)previousRow</b>	Index of first selected row after <i>previousRow</i>
– <b>redisplayEverything</b>	Displays all of DBFetchGroup's DBAssociations

## Manipulating Contents

– <b>deleteCurrentSelection</b>	Deletes the selected records from the DBRecordList
– (BOOL) <b>insertNewRecordAt:(unsigned int)index</b>	Inserts a (default) record in the DBRecordList at <i>index</i>
– <b>fetchContentsOf:aSource usingQualifier:aQualifier</b>	Replaces all records by reading <i>aSource</i> using <i>aQualifier</i>

## Dealing with Changes

– (BOOL) <b>hasUnsavedChanges</b>	YES if the DBRecordList has been changed but not saved
– (BOOL) <b>validateCurrentRecord</b>	YES unless delegates for editor or DDMModule object
– <b>saveChanges</b>	Saves changes in this or subordinate fetch groups
– <b>discardChanges</b>	Discards changes in this and subordinate fetch groups

## Using Associations

– <b>addExpression:newExpression</b>	Adds <i>newExpression</i> to the list of expressions to fetch
– <b>takeValueFromAssociation:anAssociation</b>	Puts the displayed value into the DBRecordList
– <b>addAssociation:newAssociation</b>	Adds <i>newAssociation</i> to the list of associations
– <b>removeAssociation:anAssociation</b>	Removes <i>anAssociation</i> from the list of associations

## Using a Delegate

<ul style="list-style-type: none"><li>- <b>delegate</b></li><li>- <b>setDelegate:<i>anObject</i></b></li></ul>	The object that receives notification messages Sets the object to receive notification messages
<b>Methods Implemented by the Delegate</b>	
- <b>fetchGroup:fetchGroup didInsertRecordAt:(int)<i>index</i></b>	Notification of a new record in the DBRecordList
- <b>(BOOL)fetchGroup:fetchGroup willValidateRecordAt:(int)<i>index</i></b>	Notification of pending validation; YES lets it proceed
- <b>fetchGroup:fetchGroup willDeleteRecordAt:(int)<i>index</i></b>	Notification of pending deletion; YES.lets it proceed
- <b>(DBFailureResponse)fetchGroup:fetchGroup willFailForReason:(DBFailureCode)<i>code</i></b>	Returns constant to indicate response to failure notice
- <b>fetchGroupDidFetch:fetchGroup</b>	Notification of new contents in DBRecordList
- <b>fetchGroupDidSave:fetchGroup</b>	Notification that DBRecordList has been saved
- <b>fetchGroupWillChange:fetchGroup</b>	Notification that user made changes in the DBRecordList
- <b>fetchGroupWillFetch:fetchGroup</b>	Notification that fetch will change the DBRecordList
- <b>(BOOL)fetchGroupWillSave:fetchGroup</b>	Notification of pending save; YES lets it proceed

---

## DBFormatter

**(abstract superclass)**

**Inherits From:** Object

### Controlling the Data Source

<ul style="list-style-type: none"><li>- <b>dataSource</b></li><li>- <b>setDataSource:<i>newDataSource</i></b></li></ul>	Returns the DBRecordList (or other source) Makes <i>newDataSource</i> 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**

- |               |   |
|---------------|---|
| – <b>init</b> | Initializes a new instance                |
| – <b>free</b> | Frees the space an instance formerly used |

### **Default**

- |   |  |
|---|--|
| – <b>setDefaultImage:<i>anImage</i></b> | Set image to be shown when the data has none |
| – <b>defaultImage</b>                   | The image displayed when the data has none   |

### **Display**

- |   |  |
|---|--|
| – <b>drawFieldAt:(unsigned int) <i>row</i><br/>          :<i>column</i><br/>          inside:(NXRect *)<i>frame</i><br/>          inView:<i>aView</i><br/>          withAttributes:(id &lt;DBTableVectors&gt;) <i>rowAttrs</i><br/>          :(id &lt;DBTableVectors&gt;) <i>columnAttrs</i><br/>          usePositions:(BOOL)<i>useRowPos</i><br/>          :(BOOL)<i>useColumnPos</i></b> | Displays one image from the data source's current record taken from position <i>row</i> or <i>column</i> of the dynamic axis, using <i>rowAttrs</i> or <i>colAttrs</i> to identify static attributes, and flags <i>useRowPos</i> and <i>useColumnPos</i> to select which |
|---|--|

### **Archiving**

- |   |  |
|---|--|
| – <b>read:(NXTypedStream *)<i>stream</i></b>  | Creates an instance by reading from a typed stream |
| – <b>write:(NXTypedStream *)<i>stream</i></b> | Archives an instance by writing to a typed stream  |

---

## DBImageView

**Inherits From:** Control : View : Responder : Object

### Internals

- **initFrame:**(const NXRect \*)*frameRect*
- **drawSelf:**(const NXRect \*)*rects*  
:(int)*rectCount*

Initializes the view in the frame coordinates  
Called by **display** to draw the image

### Getting/Setting the Image

- **image**
- **setImage:***newImage*

Returns the image being displayed  
Makes *newImage* the image to display

### Getting/Setting the Border

- **setStyle:**(int)*newStyle*
- (int)**style**

Sets the style of border for the image  
Returns a constant indicating the border style

### Editing

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

YES if the image can be deleted or replaced  
Allow/prohibit deleting or replacing the image

---

## DBModule

**Inherits From:** Object

### Reporting the Context

- **database**
- **entity**
- **rootFetchGroup**
- **associationForObject:***anObject*

The DBModule's DBDatabase  
The DBModule's DBEntity  
The DBModule's root DBFetchGroup  
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

+ <b>initialize</b>	Automatically invoked to initialize the class
- <b>initForEntity:(id &lt;DBEntities&gt;)anEntity</b>	Initializes a new instance to select from <i>anEntity</i>
- <b>initForEntity:(id &lt;DBEntities&gt;)anEntity fromDescription:(const unsigned char *)descriptionFormat, ...</b>	Initializes to select from <i>anEntity</i> by <i>descriptionFormat</i>
- <b>copyFromZone:(NXZone*)z</b>	Returns a copy of the DBQualifier, allocating from <i>z</i>
- <b>free</b>	Frees space that a DBQualifier formerly used

### Modifying

- <b>addDescription:(const unsigned char *)descriptionFormat, ...</b>	Appends <i>descriptionFormat</i> to the qualifier descriptions
- <b>setEntity:(id &lt;DBEntities&gt;)anEntity andDescription:(const unsigned char *)descriptionFormat, ...</b>	Sets both <i>anEntity</i> and qualifying <i>descriptionFormat</i>
- <b>(BOOL)setName:(const char *)aName</b>	Assigns the DBQualifier <i>aName</i> and returns YES
- <b>(BOOL)empty</b>	Deletes the qualifying descriptions and returns YES

### Querying

- <b>(const char *)name</b>	Returns the name assigned to the DBQualifier
- <b>(id &lt;DBEntities&gt;)entity</b>	Returns the DBQualifier's entity
- <b>(BOOL)isEmpty</b>	Returns YES if the qualifying descriptions are empty

### Archiving

- <b>read:(NXTypedStream *)stream</b>	Creates an instance by reading from a typed stream
- <b>write:(NXTypedStream *)stream</b>	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) <i>aMode</i>	Sets the DBRecordList's retrieval strategy
– (DBRecordListRetrieveMode)currentRetrieveMode	Returns a constant identifying the retrieval strategy

### Fetching Data from the Database

– fetchRecordForRecordKey:(DBValue *) <i>aValue</i>	Fetches records qualified by matching <i>aValue</i>
– fetchUsingQualifier:(DBQualifier *) <i>aQualifier</i>	Empties, then fetches records selected by <i>aQualifier</i>
– fetchUsingQualifier:(DBQualifier *) <i>aQualifier</i> empty: <i>emptyFirst</i>	Fetches records selected by <i>aQualifier</i> ; if <i>emptyFirst</i> is YES, first empties the record list
– (unsigned int)recordLimit	Returns the maximum number of records to fetch
– setRecordLimit:(unsigned int) <i>count</i>	Sets the maximum number of records to fetch

### Accessing Data in the DBRecordList

– getValue:(DBValue *) <i>aValue</i> forProperty: <i>aProperty</i>	Puts the current record's value for <i>aProperty</i> into <i>aValue</i>
– getValue:(DBValue *) <i>aValue</i> forProperty: <i>aProperty</i> at:(unsigned int) <i>index</i>	Puts value of <i>aProperty</i> for the record at <i>index</i> into <i>aValue</i>
– getRecordKeyValue:(DBValue *) <i>aValue</i>	Puts the key value for the current record into <i>aValue</i>
– getRecordKeyValue:(DBValue *) <i>aValue</i> at:(unsigned int) <i>index</i>	Puts the key value for the record at <i>index</i> into <i>aValue</i>

## Modifying Data in the DBRecordList

– <b>setValue:(DBValue *)<i>aValue</i></b> <b>forProperty:<i>aProperty</i></b>	Sets the current record's value of <i>aProperty</i> to <i>aValue</i>
– <b>setValue:(DBValue *)<i>aValue</i></b> <b>forProperty:<i>aProperty</i></b> <b>at:(unsigned int)<i>index</i></b>	Sets value of <i>aProperty</i> for record at <i>index</i> to <i>aValue</i>
– <b>insertRecordAt:(unsigned int)<i>index</i></b>	Inserts a (default) record ahead of the record at <i>index</i>
– <b>appendRecord</b>	Inserts a (default) record after the last one
– <b>newRecord</b>	Inserts a (default) record to precede the current record
– <b>(BOOL)isNewRecord</b>	YES if the current record is one that has been inserted
– <b>(BOOL)isNewRecordAt:(unsigned int)<i>index</i></b>	YES if the record at <i>index</i> had been inserted
– <b>deleteRecord</b>	Deletes the current record
– <b>deleteRecordAt:(unsigned int)<i>index</i></b>	Deletes the record at <i>index</i>
– <b>(BOOL)isModified</b>	YES if the current record has been changed or inserted
– <b>(BOOL)isModifiedAt:(unsigned int)<i>index</i></b>	YES if the record at <i>index</i> has been changed or inserted
– <b>(BOOL)isModifiedForProperty:<i>aProperty</i></b> <b>at:(unsigned int)<i>index</i></b>	YES if <i>aProperty</i> of the record at <i>index</i> has been changed

## Using Record Indexes

– <b>(unsigned int)positionForRecordKey:(DBValue *)<i>aValue</i></b>	Returns the index of the record whose key is <i>aValue</i>
– <b>moveRecordAt:(unsigned int)<i>sourceIndex</i></b> <b>to:(unsigned int)<i>destinationIndex</i></b>	Moves record at <i>sourceIndex</i> to precede the record now at <i>destinationIndex</i>
– <b>swapRecordAt:(unsigned int)<i>anIndex</i></b> <b>withRecordAt:(unsigned int)<i>anotherIndex</i></b>	Transposes the positions of the two records

## Saving Data

– <b>(unsigned int)saveModifications</b>	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**
  - **free**
- Initializes a new instance  
Frees space formerly used by a DBRecordStream

### Setting up a DBRecordStream

- **addRetrieveOrder:(DBRetrieveOrder)anOrder for:(id <DBProperties>)aProperty**
  - **(List \*)setProperties:(List \*)propertyList ofSource:aSource**
  - **(List \*)getProperties:(List \*)propertyList**
  - **(List \*)setKeyProperties:(List \*)propertyList**
  - **(List \*)getKeyProperties:(List \*)keyList**
- Appends *anOrder* (up/down) to sort criteria for *aProperty*  
Sets/returns list of properties wanted from entity *aSource*  
Returns and puts into *propertyList* the stream's properties  
Sets and returns *propertyList* as the stream's key properties  
Returns/ puts into *propertyList* the stream's key properties

### Fetching Data

- **fetchUsingQualifier:(DBQualifier \*)aQualifier**
  - **cancelFetch**
  - **(DBRecordRetrieveStatus)currentRetrieveStatus**
- Starts fetching records that pass *aQualifier*  
Stops fetching and sends **fetchDone** to DBDatabase  
DB\_Ready/NotReady, DB\_FetchInProgress/Completed

### Accessing Data

- **getValue:(DBValue \*)aValue forProperty:aProperty**
  - **getRecordKeyValue:(DBValue \*)aValue**
  - **setNext**
- Puts current record's *aProperty*'s value into *aValue*  
Puts current record's key value into *aValue*  
Makes next record available; **nil** if none left

### Modifying Data

- **setValue:(DBValue \*)aValue forProperty:aProperty**
  - **newRecord**
  - **deleteRecord**
  - **(BOOL)isNewRecord**
- Sets the current record's *aProperty* to *aValue*  
Inserts new, empty record at the current record  
Deletes the current record  
YES if the current record is a new one

– <code>(BOOL)isModified</code>	YES if the current record is new or has been modified
– <code>(BOOL)isReadOnly</code>	YES if the record stream cannot be modified

## Saving Modifications

– <code>(unsigned int)saveModifications</code>	Writes current record's modifications to the database
--	---

## Resetting a DBRecordStream

– <code>clear</code>	Resets everything except the delegate
----------------------	---------------------------------------

## Assigning Delegates

– <code>delegate</code>	The object that receives notification messages
– <code>setDelegate:<i>anObject</i></code>	Sets the object that will receive notification messages
– <code>binderDelegate</code>	The object that receives notification messages for binders
– <code>setBinderDelegate:<i>anObject</i></code>	Sets the object to receive notification messages for binders

## Method Implemented by the Delegate

– <code>(BOOL)recordStream:<i>sender</i> willFailForReason:(DBFailureCode) <i>aCode</i></code>	Invoked when changes can't be saved; <i>aCode</i> tells why; YES acknowledges failure; NO tries to proceed with those records that are not affected
– <code>(BOOL)recordStreamPrepareCurrentRecordForModification:<i>aRecordStream</i></code>	Invoked when a record will be modified or deleted; YES permits modification to proceed

## DBTableVector

**Inherits From:** Object

**Conforms To:** DBTableVectors

## Creating the Object

– <code>initIdentifier:<i>anIdentifier</i></code>	Initialize a DBTableVector for property <i>anIdentifier</i>
– <code>free</code>	Free the space formerly allocated to a DBTableVector

---

## DBTableView

**Inherits From:** ScrollView : View : Responder : Object

### Initializing and Freeing

- **initFrame:(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  
to:(unsigned int)endColumn** Redraw because data changed in a block of columns

## 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 an 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 an 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*
  - **setEditable:**(BOOL)*flag*
  - (BOOL)**isEditable**
- Selects an item and invokes editor
- Enables/disables editing.
- YES of the DBTableView is editable.

## Handling the Selection

- **setMode:**(int)*newMode*
  - (int)**mode**
  - **allowEmptySel:**(BOOL)*flag*
  - (BOOL)**doesAllowEmptySel**
  - (unsigned int)**selectedRowCount**
  - (unsigned int)**selectedColumnCount**
  - (int)**selectedRow**
  - (int)**selectedColumn**
  - (BOOL)**isRowSelected:**(unsigned int)*row*
  - (BOOL)**isColumnSelected:**(unsigned int)*column*
  - **deselectAll:***sender*
  - **selectAll:***sender*
  - **setRowSelectionOn:**(unsigned int)*start*  
    :(unsigned int)*end*  
    **to:**(BOOL)*flag*
  - **setColumnSelectionOn:**(unsigned int)*start*  
    :(unsigned int)*end*  
    **to:**(BOOL)*flag*
  - **selectRow:**(unsigned int)*row*  
    **byExtension:**(BOOL)*flag*
  - **selectColumn:**(unsigned int)*column*  
    **byExtension:**(BOOL)*flag*
  - **deselectRow:**(unsigned int)*row*
  - **deselectColumn:**(unsigned int)*column*
  - (unsigned int)**selectedRowAfter:**(unsigned int)*aRow*
  - (unsigned int)**selectedColumnAfter:**(unsigned int)*aColumn*
  - **sendAction:**(SEL)*anAction*  
    **to:***anObject*  
    **forSelectedRows:**(BOOL)*flag*
- Make selection list mode, radio mode, or none.
- Returns DB\_NOSELECT/RADIOMODE/LISTMODE
- Allow/prohibit user to leave nothing selected
- YES if user may leave nothing selected
- Number of rows currently selected
- Number of columns currently selected
- The row number of the selected row
- The *column* number of the selected column
- YES if *row* is selected
- YES if column is selected
- Makes nothing selected.
- Makes all rows and columns selected.
- Sets block of rows to selected (YES) or deselected (NO)
- Sets block of columns to selected (YES) or deselected
- Selects row, or extends selection if flag is YES
- Selects row, or extends selection if flag is YES
- Deselects the indicated row
- Deselects the indicated column
- Index of the first selected row after *aRow*
- Index of the first selected column after *aColumn*
- 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**
  - **write:(NXTypedStream \*)stream**
  - **finishUnarchiving**
- Creates an instance by reading from a typed stream  
Archives an instance by writing to a typed stream  
Automatically invoked final step in unarchiving

---

## DBTextFormatter

Inherits From: DBFormatter : Object

### Initializing

- **init**
  - **free**
- Initializes a new DBTextFormatter instance  
Frees the space allocated to a DBTextFormatter.

### Manipulating Font

- **font**
  - **setFont:aFont**
- Returns the formatter's font  
Makes *aFont* the formatter's font

### Batching Format Requests

- **beginBatching:(id <DBTableVectors>) attrs**
  - **resetBatching:(id <DBTableVectors>) attrs**
  - **endBatching**
- The format *attrs* applies to all following records  
Begins batching if not already in effect  
Completes sequence of records in same format

### Archiving

- **read:(NXTypedStream \*)stream**
  - **write:(NXTypedStream \*)stream**
- Creates an instance by reading from a typed 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) <i>aDouble</i>	Sets the object's value to <i>aDouble</i>
- setFloatValue:(float) <i>aFloat</i>	Sets the object's value to <i>aFloat</i>
- setIntValue:(int) <i>anInt</i>	Sets the object's value to <i>anInt</i>
- setObjectValue:(id) <i>anObject</i>	Sets the object's value to <i>anObject</i>
- setObjectValueNoCopy:(id) <i>anObject</i>	Sets the object's value so that it points to <i>anObject</i>
- setValue:(const char *) <i>aString</i>	Sets the object's value to <i>aString</i>
- setValueNoCopy:(const char *) <i>aString</i>	Sets the object's value so that it points to <i>aString</i>
- setValueFrom:(DBValue *) <i>aValue</i>	Sets the object's to have the same value as <i>aValue</i>
- setNull	Sets the object's value to NULL

### Reporting Values

- (id <DBTypes>)valueType	Returns the type of value the object contains
- (BOOL)isEqual:(DBValue *) <i>anotherValue</i>	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

### Archivingt

- read:(NXTypedStream *) <i>stream</i>	Creates an instance by reading from a typed stream
- write:(NXTypedStream *) <i>stream</i>	Archives an instance by writing to a typed stream

# Protocols

---

## DBContainers

**Adopted By:** DBRecordList

### Mandatory Methods

– <b>addObject:</b> <i>anObject</i> <b>forBinder:</b> (DBBinder *) <i>aBinder</i>	Adds <i>anObject</i> to <i>aBinder</i> 's container
– (unsigned int) <b>count</b>	The number of objects in the container
– <b>empty</b>	Removes (but doesn't free) objects in the container
– <b>freeObjects</b>	Frees space formerly allocated to objects in the container
– <b>objectAt:</b> (unsigned) <i>index</i> <b>forBinder:</b> (DBBinder *) <i>aBinder</i>	Returns the <i>index</i> 'th object in <i>aBinder</i> 's container
– (unsigned int) <b>prepareForBinder:</b> (DBBinder *) <i>aBinder</i>	Readies the container to move data; returns <b>count</b>

### Optional Methods

– <b>binder:</b> (DBBinder *) <i>aBinder</i> <b>didAcceptObject:</b> <i>anObject</i>	Notification when <i>anObject</i> successfully stored in <i>aBinder</i>
– <b>binder:</b> (DBBinder *) <i>aBinder</i> <b>didRejectObject:</b> <i>anObject</i>	Notification when <i>anObject</i> could not be stored in <i>aBinder</i>

---

## DBCursorPositioning

**Adopted By:** DBBinder  
                  DBRecordList

### Setting the Position

– <b>setFirst</b>	Sets cursor to container's first record; returns the record
-------------------	---

– <b>setLast</b>	Sets cursor to container's last record; returns the record
– <b>setNext</b>	Sets cursor to container's next record; returns the record
– <b>setPrevious</b>	Sets cursor to container's previous record; returns the record
– <b>setTo:(long)<i>index</i></b>	Sets cursor to <i>index</i> ; returns the <i>index</i> 'th record

## Querying the Position

– <b>(long)currentPosition</b>	Returns the current index (of objects in the container)
--------------------------------	---

## **DBCustomAssociation** (informal protocol)

**Category Of:** Object

### Access to the Associated Value

– <b>association:association setValue:(DBValue *)<i>value</i></b>	Sets <i>value</i> into the associated data source
– <b>association:association getValue:(DBValue *)<i>value</i>;</b>	Puts the value of the associated data source into <i>value</i>

### Notifications to the Associated Display

– <b>associationContentsDidChange:<i>association</i></b>	Notification that the data source has changed
– <b>associationSelectionDidChange:<i>association</i></b>	Notification of a change in selection in the display
– <b>associationCurrentRecordDidDelete:<i>association</i></b>	Notification that current record has been deleted from the data source

---

## DBEntities

**Adopted By:** none

**Incorporates:** DBTypes

### Querying the Object

- |   |  |
|---|--|
| – (const char *) <b>name</b>                        | Returns the name of the entity                 |
| – (DBDatabase *) <b>database</b>                    | The DBDatabase object that created the entity  |
| – <b>getProperties:</b> (List *) <i>aList</i>       | Puts the entity's properties into <i>aList</i> |
| – <b>propertyNamed:</b> (const char *) <i>aName</i> | Returns the DBProperty named <i>aName</i>      |

### Comparing the Object

- |  |  |
|--|--|
| – (BOOL) <b>matchesEntity:</b> (id <DBEntities>) <i>anEntity</i> | YES if receiver is equivalent to <i>anEntity</i> |
|--|--|

---

## DBExpressionValues

**Adopted By:** DBExpression  
DBQualifier  
DBValue

### Methods

- |   |  |
|---|--|
| – (const char *) <b>expressionValue</b> | Returns the receiver's query-language expression     |
| – (BOOL) <b>isDeferredExpression</b>    | 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`  
`sender:sender` Notification when a formatter will change a value
- **(BOOL)formatterWillChangeValueFor:rowIdentifier**  
`:columnIdentifier`  
`to:aValue`  
`sender:sender` Notification when a formatter will change a value,  
with the proposed new value

## Notification Using the Changed Cell's Position

- **formatterDidChangeValueFor:identifier**  
`at:(unsigned int)position`  
`sender:sender` Notification when a formatter has changed a value  
in one of the fields of a display; returns `self`
- **(BOOL)formatterWillChangeValueFor:identifier**  
`at:(unsigned int)position`  
`sender:sender` Notification when a formatter will change a value
- **(BOOL)formatterWillChangeValueFor:identifier**  
`at:(unsigned int)position`  
`to:aValue`  
`sender:sender` Notification when a formatter will change a value,  
with the proposed new value

## DBFormatterViewEditing

**Category Of:** Object

### Accepting changes

- **(BOOL)formatterEndedEditing:sender**  
`endChar:(unsigned short)whyEnd` Invoked by editor when user ends editing;  
returns YES unless editing in progress

---

## DBProperties

Adopted By: DBExpression

### Identifying a Property

- |  |   |
|--|---|
| – (const char*) <b>name</b>  | Name of the property                                      |
| – (BOOL) <b>setName:(const char *)aName</b>                        | Set the name of a custom property; YES if successful      |
| – (id <DBEntities>) <b>entity</b>                                  | The entity to which the property belongs                  |
| – (BOOL) <b>matchesProperty:(id &lt;DBProperties&gt;)aProperty</b> | YES if receiver identifies same thing as <i>aProperty</i> |

### Querying a Property

- |  |  |
|--|--|
| – (id <DBTypes>) <b>propertyType</b>                               | The type, as an object that responds to DBTypes protocol         |
| – (BOOL) <b>isSingular</b>   | YES if receive is an attribute or a one-to-one relationship      |
| – (BOOL) <b>isReadOnly</b>   | YES if the property's data is read only                          |
| – (BOOL) <b>isKey</b>  | YES if the property is a key property in the entity              |
| – (BOOL) <b>matchesProperty:(id &lt;DBProperties&gt;)aProperty</b> | YES if receiver has same name in same entity as <i>aProperty</i> |

---

## DBTableDataSources

(informal protocol)

Category Of: Object

### Reporting Table Size

- |                                     |  |
|-------------------------------------|--|
| – (unsigned int) <b>rowCount</b>    | Number of rows in the data source table    |
| – (unsigned int) <b>columnCount</b> | Number of columns in the data source table |

## Getting/Setting Data

<b>– getValueFor:<i>rowIdentifier</i></b> <i>:columnIdentifier</i> <i>into:aValue</i>	Puts source value identified by properties <i>rowIdentifier</i> and <i>columnIdentifier</i> into <i>aValue</i>
<b>– getValueFor:<i>identifier</i></b> <i>at:(unsigned int) aPosition</i> <i>into:aValue</i>	Puts value for property <i>identifier</i> at record <i>aPosition</i> into <i>aValue</i>
<b>– setValueFor:<i>rowIdentifier</i></b> <i>:columnIdentifier</i> <i>from:aValue</i>	Sets source value identified by properties <i>rowIdentifier</i> and <i>columnIdentifier</i> to <i>aValue</i>
<b>– setValueFor:<i>identifier</i></b> <i>at:(unsigned int) aPosition</i> <i>from:aValue</i>	Sets value for property <i>identifier</i> at record <i>aPosition</i> to <i>aValue</i>

---

## DBTableValues (informal protocol)

**Category Of:** Object

### Getting a Value from the Table

<b>– objectValue</b>	The item's value as an object
<b>– (int)intValue</b>	The item's value as an int
<b>– (double)doubleValue</b>	The item's value as an double
<b>– (float)floatValue</b>	The item's value as an float
<b>– (const char *)stringValue</b>	The item's value as an string

### Setting a Value in the Table

<b>– setObjectValue:(int)anObj</b>	Sets the item's value to <i>anObj</i>
<b>– setIntValue:(int)anInt</b>	Sets the item's value to <i>anInt</i>
<b>– setDoubleValue:(double)aDouble</b>	Sets the item's value to <i>aDouble</i>
<b>– setFloatValue:(float)aFloat</b>	Sets the item's value to <i>aFloat</i>
<b>– setStringValue:(const char *) aString</b>	Sets the item's value to <i>aString</i>

## DBTableVectors

**Category Of:** Object

### Controlling/Reporting Formatter

- formatter
- **setFormatter:***newFormatter*

The formatter responsible for formatting the vector  
Makes *newFormatter* the vector's formatter

### Controlling/Reporting Data Link

- identifier
- **setIdentifier:***aDataAttribute*

The identifier for the property that the vector displays  
Makes *aDataAttribute* the vector's identifier

### Controlling/Reporting Editing

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

YES if the vector's display is editable  
Permit/prohibit editing of the vector's display

### Controlling/reporting Size

- **(BOOL)isResizable**
- **setResizable:(BOOL)***flag*
- **(BOOL)isAutosizable**
- **setAutosizable:(BOOL)***flag*
- **(NXCoord)size**
- **(NXCoord)sizeTo:(NXCoord)***newSize*
- **(NXCoord)minSize**
- **setMinSize:(NXCoord)** *newMinSize*
- **(NXCoord)maxSize**
- **setMaxSize:(NXCoord)** *newMaxSize*

YES if the vector's display can be resized  
Permit/prohibit resizing the vector's display  
YES if automatically resized when the view is resized  
Permit/prohibit automatic resizing  
Width and height of an item in the vector's display  
Sets width and height of an item in the vector's display  
The minimum limit on resizing  
Sets minimum limit on resizing  
The maximum limit on resizing  
Sets maximum limit on resizing

### Controlling/Reporting Title

- **(const char \*)title**
- **setTitle:(const char \*)** *title*
- **titleFont**
- **setTitleFont:***fontObj*

The title (if row or column headings are shown)  
Makes *title* the vector's title  
The font for displaying the title  
Makes *fontObj* the font for displaying the title

- (int)titleAlignment
  - setTitleAlignment:(int)*align*
- A constant: NX\_LEFT/CENTERED/RIGHTALIGNED  
Sets the title alignment to *align*

## Controlling/Reporting Content Alignment

- (int)contentAlignment
  - setContentAlignment:(int) *align*
- A constant: NX\_LEFT/CENTERED/RIGHTALIGNED  
Sets the content alignment to *align*

---

## DBTransactions

**Adopted By:** DBBasicAdaptor : Object

### Basic Transaction Commands

- (BOOL)beginTransaction
  - (BOOL)commitTransaction
  - (BOOL)rollbackTransaction
- Begins a transaction; YES if successful  
Commits a transaction; YES if successful  
Rolls back a transaction; YES if successful

---

## DBTypes

**Adopted By:** none

### Querying for Type

- (const char \*)objcType
  - (const char \*)databaseType
  - (const char \*)objcClassName
- "id" /"char \*" /"int" /"float" /"double" as appropriate  
What the adaptor returns for the data's type, for example  
"@" (for id) /"\*" (for char \*) /"i" /"f" /"d"  
Name of the stored data's class (if any)

### Comparing Types

- (BOOL)isEntity
  - (BOOL)matchesType:(id <DBTypes>)anObject
- YES if the object conforms to DBEntities protocol  
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)0x7fffffff)

## **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<sub>x</sub> src<sub>y</sub> width height srcgstate dest<sub>x</sub> dest<sub>y</sub> op composite</i>	-	composite rectangle in source graphics state with image in current window
<i>dest<sub>x</sub> dest<sub>y</sub> width height op compositerect</i>	-	composite rectangle of current color and coverage with image in current graphics state
<i>src<sub>x</sub> src<sub>y</sub> width height srcgstate dest<sub>x</sub> dest<sub>y</sub> delta dissolve</i>	-	dissolve between area of window referred to by <i>srcgstate</i> and equal area of window referred to by current graphics state
<i>window currentwindowalpha state</i>	-	return information about how window's alpha values are stored
<i>x y width height proc<sub>0</sub> [... proc<sub>n-1</sub>] string bool readimage</i>	-	read image's pixel values and pass to corresponding procedures
<i>x y width height matrix sizeimage pixelswide pixelshigh bits/sample matrix multiproc ncolors</i>	<i>get various parameters required for readimage to read the image</i>	
<i>pixelswide pixelshigh bits/sample matrix proc<sub>0</sub> [... proc<sub>n</sub>] multiproc ncolors alphaimage</i>	-	render data and alpha information supplied by one or more procedures

## Instance Drawing Operators (NeXTSTEP)

<i>- newinstance -</i>	remove instance drawing from current window
<i>bool setinstance -</i>	turn instance-drawing mode on or off
<i>x y width height hideinstance -</i>	remove instance drawing from rectangle

## Mouse and Cursor Operators (NeXTSTEP)

<i>eventnum stilldown bool</i>	test whether left/only mouse button is still down from mouse-down <i>eventnum</i>
<i>eventnum rightstilldown bool</i>	test whether right mouse button is still down from mouse-down <i>eventnum</i>
<i>window currentmouse x y</i>	return mouse location in base coordinates *
<i>- buttondown bool</i>	test whether left/only mouse button is down

<i>x</i>	<b>rightbuttondown</b>	<i>bool</i>	test whether right mouse button is down
<i>x y</i>	<b>setmouse</b>	—	set mouse and cursor location
<i>dx dy</i>	<b>adjustcursor</b>	—	adjust cursor location by ( <i>dx</i> , <i>dy</i> )
<i>x y mx my</i>	<b>setcursor</b>	—	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> ).
	<b>hidcursor</b>	—	remove cursor from screen
	<b>showcursor</b>	—	restore cursor to screen
	<b>obscurecursor</b>	—	remove cursor from screen until mouse moves
	<b>revealcursor</b>	—	restore cursor if still obscured
<i>bool context</i>	<b>setwaitcursorenabled</b>	—	enable or disable wait cursor
<i>context</i>	<b>currentwaitcursorenabled</b>	<i>bool</i>	return whether wait cursor is enabled
<i>x y width height leftbool rightbool</i>	<b>settrackingrect</b>	—	set tracking rectangle in window referred to by <i>gstate</i>
<i>trectnum gstate</i>	<b>cleartrackingrect</b>	—	clear tracking rectangle in <i>gstate</i>

## Event Operators (NeXTSTEP)

<i>window</i>	<b>seventmask</b>	—	set window's Server-level event mask *
<i>window</i>	<b>currenteventmask</b>	<i>mask</i>	return window's current Server-level event mask *
<i>type x y time flags window</i>	<b>posteventbycontext</b>	<i>bool</i>	post event to specified context
<i>subType misc0 misc1 context</i>			
<i>bool window</i>	<b>setsendexposed</b>	—	set whether window-changed events are generated for exposed window areas *
	<b>setflushexposures</b>	—	set whether window-exposed and screen-changed subevents are flushed *
<i>bool context</i>	<b>setwaitcursorenabled</b>	—	enable or disable wait cursor operation
<i>context</i>	<b>currentwaitcursorenabled</b>	<i>bool</i>	return status of wait cursor in <i>context</i>
<i>context</i>	<b>setactiveapp</b>	—	establish application having <i>context</i> as the active application*
	<b>currentactiveapp</b>	<i>context</i>	return context of active application *

## Frame Buffer Operators (NeXTSTEP)

	<b>– countframebuffers</b>	<i>count</i>	return number of frame buffers in use
<i>fbnum</i>	<b>currentframebuffertransfer</b>	<i>redproc greenproc blueproc grayproc</i>	return current transfer function for <i>fbnum</i>
<i>index string</i>	<b>framebuffer</b>	<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>	<b>setframebuffertransfer</b>	<b>–</b>	set the transfer function for <i>fbnum</i>

## Window Management Operators (NeXTSTEP)

<i>x y width height type</i>	<b>window</b>	<i>window</i>	create window and return its number *	
	<b>window</b>	<b>termwindow</b>	remove window from screen list; cause eventual freeing *	
<i>type window</i>	<b>setwindowtype</b>	<b>–</b>	set <i>window</i> 's type to <i>type</i>	
<i>window</i>	<b>windowdevice</b>	<b>–</b>	set device of current graphics state to <i>window</i>	
<i>window</i>	<b>windowdeviceround</b>	<b>–</b>	set device to <i>window</i> and round coordinate system to integer pixels	
	<b>– currentwindow</b>	<i>window</i>	return window number of current device	
	<b>– setexposurecolor</b>	<b>–</b>	set exposure color for nonretained window of current graphics state	
	<b>– flushgraphics</b>	<b>–</b>	flush drawing in buffered window to screen	
<i>place otherwindow window</i>	<b>orderwindow</b>	<b>–</b>	order Above or Below <i>otherwindow</i> (0 for all) or Out of screen list *	
	<b>level window</b>	<b>setwindowlevel</b>	<b>–</b>	set window tier for <i>window</i> to <i>level</i>
	<i>window</i>	<b>currentwindowlevel</b>	<i>level</i>	return window tier for <i>window</i>
	<b>– frontwindow</b>	<i>window</i>	<b>–</b>	return frontmost window *
<i>x y width height window</i>	<b>placewindow</b>	<b>–</b>	reposition and resize with intersecting pixels unchanged *	
	<i>x y window</i>	<b>movewindow</b>	<b>–</b>	move lower left to screen coordinates ( <i>x, y</i> ) *
	<i>window</i>	<b>currentwindowbounds</b>	<i>x y width height</i>	return window's location and size in screen coordinates
<i>x y place otherwindow</i>	<b>findwindow</b>	<i>x' y' window bool</i>	locate window under screen coordinates ( <i>x, y</i> ) and give base coordinates (or return <i>false</i> )	

<i>bool</i>	<i>window</i>	<b>setautofill</b>	—	set whether exposure color fills window's exposed areas automatically
<i>dict</i>	<i>window</i>	<b>setwindowdict</b>	—	set window's dictionary *
	<i>window</i>	<b>currentwindowdict</b>	<i>dict</i>	return window's dictionary *
	<i>context</i>	<b>countscreenlist</b>	<i>count</i>	return number of windows in screen list that belong to <i>context</i>
	<i>context</i>	<b>countwindowlist</b>	<i>count</i>	return number of windows that belong to <i>context</i>
<i>array</i>	<i>context</i>	<b>screenlist</b>	<i>subarray</i>	return window numbers of all windows in screen list that belong to <i>context</i>
<i>array</i>	<i>context</i>	<b>windowlist</b>	<i>subarray</i>	return window numbers of all windows that belong to <i>context</i>
<i>context</i>	<i>window</i>	<b>setowner</b>	—	set owning PostScript context of <i>window</i> to <i>context</i>
	<i>window</i>	<b>currentowner</b>	<i>context</i>	return PostScript context that owns <i>window</i>
	<i>window</i>	<b>currentdeviceinfo</b>	<i>min max bool</i>	return window's sampling density and whether device is color
	<i>depth</i>	<b>setdefaultdepthlimit</b>	—	set depth limit for new windows *
	—	<b>currentdefaultdepthlimit</b>	<i>depth</i>	return depth limit for new windows *
	<i>window</i>	<b>currentwindowdepth</b>	<i>depth</i>	return <i>window</i> 's depth *
<i>depth</i>	<i>window</i>	<b>setwindowdepthlimit</b>	—	set <i>window</i> 's depth limit *
	<i>window</i>	<b>currentwindowdepthlimit</b>	<i>depth</i>	return <i>window</i> 's depth limit *
<i>dumplevel</i>	<i>window</i>	<b>dumpwindow</b>	—	report position and number of bytes of backing store for <i>window</i> *
<i>dumplevel</i>	<i>context</i>	<b>dumpwindows</b>	—	report position and number of bytes of backing store for all windows owned by <i>context</i> *

## statusdict Operators (NeXTSTEP)

—	<b>ostype</b>	<i>int</i>	return category of operating system (1=standalone, 3=UNIX® variant)
—	<b>osname</b>	<i>string</i>	return name of operating system

## Operand Stack Manipulation Operators

<i>any</i>	<b>pop</b>	<b>-</b>	discard top element
<i>any<sub>1</sub> any<sub>2</sub></i>	<b>exch</b>	<i>any<sub>2</sub> any<sub>1</sub></i>	exchange top two elements
<i>any</i>	<b>dup</b>	<i>any any</i>	duplicate top element
<i>any<sub>1</sub> ... any<sub>n</sub> n</i>	<b>copy</b>	<i>any<sub>1</sub> ... any<sub>n</sub> any<sub>1</sub> ... any<sub>n</sub></i>	duplicate top <i>n</i> elements
<i>any<sub>n</sub> ... any<sub>0</sub> n</i>	<b>index</b>	<i>any<sub>n</sub> ... any<sub>0</sub> any<sub>n</sub></i>	duplicate arbitrary element
<i>a<sub>n-1</sub> ... a<sub>0</sub> n j</i>	<b>roll</b>	<i>a<sub>(j-1) mod n</sub> ... a<sub>0</sub> a<sub>n-1</sub> ... a<sub>j mod n</sub></i>	roll <i>n</i> elements up <i>j</i> times
<b> –</b> <i>any<sub>1</sub> ... any<sub>n</sub></i>	<b>clear</b>	<b> –</b>	discard all elements
<b> –</b> <i>any<sub>1</sub> ... any<sub>n</sub></i>	<b>count</b>	<b> –</b> <i>any<sub>1</sub> ... any<sub>n</sub> n</i>	count elements on stack
<b>–</b>	<b>mark</b>	<b>mark</b>	push mark on stack
<i>mark obj<sub>1</sub> ... obj<sub>n</sub></i>	<b>cleartomark</b>	<b>–</b>	discard elements down through mark
<i>mark obj<sub>1</sub> ... obj<sub>n</sub></i>	<b>counttomark</b>	<i>mark obj<sub>1</sub> ... obj<sub>n</sub> n</i>	count elements down to mark

## Arithmetic and Math Operators

<i>num<sub>1</sub> num<sub>2</sub></i>	<b>add</b>	<i>sum</i>	<i>num<sub>1</sub></i> plus <i>num<sub>2</sub></i>
<i>num<sub>1</sub> num<sub>2</sub></i>	<b>div</b>	<i>quotient</i>	<i>num<sub>1</sub></i> divided by <i>num<sub>2</sub></i>
<i>int<sub>1</sub> int<sub>2</sub></i>	<b>idiv</b>	<i>quotient</i>	integer divide
<i>int<sub>1</sub> int<sub>2</sub></i>	<b>mod</b>	<i>remainder</i>	<i>int<sub>1</sub></i> mod <i>int<sub>2</sub></i>
<i>num<sub>1</sub> num<sub>2</sub></i>	<b>mul</b>	<i>product</i>	<i>num<sub>1</sub></i> times <i>num<sub>2</sub></i>
<i>num<sub>1</sub> num<sub>2</sub></i>	<b>sub</b>	<i>difference</i>	<i>num<sub>1</sub></i> minus <i>num<sub>2</sub></i>
<i>num<sub>1</sub></i>	<b>abs</b>	<i>num<sub>2</sub></i>	absolute value of <i>num<sub>1</sub></i>
<i>num<sub>1</sub></i>	<b>neg</b>	<i>num<sub>2</sub></i>	negative of <i>num<sub>1</sub></i>
<i>num<sub>1</sub></i>	<b>ceiling</b>	<i>num<sub>2</sub></i>	ceiling of <i>num<sub>1</sub></i>
<i>num<sub>1</sub></i>	<b>floor</b>	<i>num<sub>2</sub></i>	floor of <i>num<sub>1</sub></i>
<i>num<sub>1</sub></i>	<b>round</b>	<i>num<sub>2</sub></i>	round <i>num<sub>1</sub></i> to nearest integer
<i>num<sub>1</sub></i>	<b>truncate</b>	<i>num<sub>2</sub></i>	remove fractional part of <i>num<sub>1</sub></i>
<i>num</i>	<b>sqrt</b>	<i>real</i>	square root of <i>num</i>
<i>num den</i>	<b>atan</b>	<i>angle</i>	arctangent of <i>num/den</i> in degrees
<i>angle</i>	<b>cos</b>	<i>real</i>	cosine of <i>angle</i> (degrees)

<i>angle</i>	<b>sin</b>	<i>real</i>	sine of <i>angle</i> (degrees)
<i>base exponent</i>	<b>exp</b>	<i>real</i>	raise <i>base</i> to <i>exponent</i> power
<i>num</i>	<b>ln</b>	<i>real</i>	natural logarithm (base e)
<i>num</i>	<b>log</b>	<i>real</i>	logarithm (base 10)
–	<b>rand</b>	<i>int</i>	generate pseudo-random integer
<i>int</i>	<b>srand</b>	–	set random number seed
–	<b>rrand</b>	<i>int</i>	return random number seed

## Array Operators

<i>int</i>	<b>array</b>	<i>array</i>	create array of length <i>int</i>
–	[	<i>mark</i>	start array construction
<i>mark obj<sub>0</sub> ... obj<sub>n-1</sub></i>	]	<i>array</i>	end array construction
<i>array</i>	<b>length</b>	<i>int</i>	number of elements in <i>array</i>
<i>array index</i>	<b>get</b>	<i>any</i>	get array element indexed by <i>index</i>
<i>array index any</i>	<b>put</b>	–	put <i>any</i> into array at <i>index</i>
<i>array index count</i>	<b>getinterval</b>	<i>subarray</i>	subarray of array starting at <i>index</i> for <i>count</i> elements
<i>array<sub>1</sub> index array<sub>2</sub></i>	<b>putinterval</b>	–	replace subarray of <i>array<sub>1</sub></i> starting at <i>index</i> by <i>array<sub>2</sub></i>
<i>array</i>	<b>aload</b>	<i>a<sub>0</sub> ... a<sub>n-1</sub> array</i>	push all elements of array on stack
<i>any<sub>0</sub> ... any<sub>n-1</sub> array</i>	<b>astore</b>	<i>array</i>	pop elements from stack into array
<i>array<sub>1</sub> array<sub>2</sub></i>	<b>copy</b>	<i>subarray<sub>2</sub></i>	copy elements of <i>array<sub>1</sub></i> to initial subarray of <i>array<sub>2</sub></i>
<i>array proc</i>	<b>forall</b>	–	execute <i>proc</i> for each element of array
<i>any<sub>0</sub> ... any<sub>n-1</sub> n</i>	<b>packedarray</b>	<i>packedarray</i>	create packed array consisting of specified <i>n</i> elements
–	<b>currentpacking</b>	<i>bool</i>	return array packing mode
<i>bool</i>	<b>setpacking</b>	–	set current array packing mode for ‘{...}’ syntax (true = packedarray)
<i>packedarray</i>	<b>length</b>	<i>int</i>	number of elements in <i>packedarray</i>
<i>packedarray index</i>	<b>get</b>	<i>any</i>	get <i>packedarray</i> element indexed by <i>index</i>
<i>packedarray index count</i>	<b>getinterval</b>	<i>subarray</i>	subarray of <i>packedarray</i> starting at <i>index</i> for <i>count</i> elements

<i>packedarray</i>	<b>aload</b>	$a_0 \dots a_{n-1}$	<i>packedarray</i>	push all elements of <i>packedarray</i> on stack
<i>packedarray</i> <sub>1</sub>	<b>array</b>	<b>copy</b>	<i>subarray</i> <sub>2</sub>	copy elements of <i>packedarray</i> <sub>1</sub> to initial subarray of <i>array</i> <sub>2</sub>
<i>packedarray</i>	<b>proc</b>	<b>forall</b>	–	execute <i>proc</i> for each element of <i>packedarray</i>

## Dictionary Operators

<i>int</i>	<b>dict</b>	<i>dict</i>	create dictionary with capacity for <i>int</i> elements
<i>dict</i>	<b>length</b>	<i>int</i>	number of key-value pairs in <i>dict</i>
<i>dict</i>	<b>maxlength</b>	<i>int</i>	capacity of <i>dict</i>
<i>dict</i>	<b>begin</b>	–	push dict on <i>dict</i> stack
–	<b>end</b>	–	pop <i>dict</i> stack
<i>key value</i>	<b>def</b>	–	associate <i>key</i> and <i>value</i> in current dict
<i>key</i>	<b>load</b>	<i>value</i>	search dict stack for <i>key</i> and return associated value
<i>key value</i>	<b>store</b>	–	replace topmost definition of <i>key</i>
<i>dict key</i>	<b>get</b>	<i>any</i>	get value associated with <i>key</i> in <i>dict</i>
<i>dict key value</i>	<b>put</b>	–	associate <i>key</i> with <i>value</i> in <i>dict</i>
–	<b>cleardictstack</b>	–	return dictionary stack to initial state
<i>dict key</i>	<b>known</b>	<i>bool</i>	test whether <i>key</i> is in <i>dict</i>
<i>key</i>	<b>where</b>	<i>dict true</i> or <i>false</i>	find dict in which <i>key</i> is defined
<i>dict</i> <sub>1</sub>	<i>dict</i> <sub>2</sub>	<b>copy</b>	<i>dict</i> <sub>2</sub>
<i>dict proc</i>	<b>forall</b>	–	execute <i>proc</i> for each element of <i>dict</i>
–	<b>errordict</b>	<i>dict</i>	push <b>errordict</b> on operand stack
–	<b>systemdict</b>	<i>dict</i>	push <b>systemdict</b> on operand stack
–	<b>userdict</b>	<i>dict</i>	push <b>userdict</b> on operand stack
–	<b>currentdict</b>	<i>dict</i>	push current dict on operand stack
–	<b>countdictstack</b>	<i>int</i>	count elements on dict stack
<i>array</i>	<b>dictstack</b>	<i>subarray</i>	copy dict stack into <i>array</i>

## String Operators

<i>int</i>	<b>string</b>	<i>string</i>	create string of length <i>int</i>
<i>string</i>	<b>length</b>	<i>int</i>	number of elements in string
<i>string index</i>	<b>get</b>	<i>int</i>	get string element indexed by <i>index</i>
<i>string index int</i>	<b>put</b>	<i>–</i>	put <i>int</i> into string at <i>index</i>
<i>string index count</i>	<b>getinterval</b>	<i>substring</i>	substring of string starting at <i>index</i> for <i>count</i> elements
<i>string<sub>1</sub> index string<sub>2</sub></i>	<b>putinterval</b>	<i>–</i>	replace substring of <i>string<sub>1</sub></i> starting at <i>index</i> by <i>string<sub>2</sub></i>
<i>string<sub>1</sub> string<sub>2</sub></i>	<b>copy</b>	<i>substring<sub>2</sub></i>	copy elements of <i>string<sub>1</sub></i> to initial substring of <i>string<sub>2</sub></i>
<i>string proc</i>	<b>forall</b>	<i>–</i>	execute <i>proc</i> for each element of string
<i>string seek</i>	<b>anchorsearch</b>	<i>post match true</i> or <i>string false</i>	determine if <i>seek</i> is initial substring of string
<i>string seek</i>	<b>search</b>	<i>post match pre true</i> or <i>string false</i>	search for <i>seek</i> in string
<i>string</i>	<b>token</b>	<i>post token true</i> or <i>false</i>	read token from start of <i>string</i>

## Relational, Boolean, and Bitwise Operators

<i>any<sub>1</sub> any<sub>2</sub></i>	<b>eq</b>	<i>bool</i>	test equal
<i>any<sub>1</sub> any<sub>2</sub></i>	<b>ne</b>	<i>bool</i>	test not equal
<i>num<sub>1</sub> str<sub>1</sub> num<sub>2</sub> str<sub>2</sub></i>	<b>ge</b>	<i>bool</i>	test greater or equal
<i>num<sub>1</sub> str<sub>1</sub> num<sub>2</sub> str<sub>2</sub></i>	<b>gt</b>	<i>bool</i>	test greater than
<i>num<sub>1</sub> str<sub>1</sub> num<sub>2</sub> str<sub>2</sub></i>	<b>le</b>	<i>bool</i>	test less or equal
<i>num<sub>1</sub> str<sub>1</sub> num<sub>2</sub> str<sub>2</sub></i>	<b>lt</b>	<i>bool</i>	test less than
<i>bool<sub>1</sub> int<sub>1</sub> bool<sub>2</sub> int<sub>2</sub></i>	<b>and</b>	<i>bool<sub>3</sub> int<sub>3</sub></i>	logical/bitwise and
<i>bool<sub>1</sub> int<sub>1</sub></i>	<b>not</b>	<i>bool<sub>2</sub> int<sub>2</sub></i>	logical/bitwise not
<i>bool<sub>1</sub> int<sub>1</sub> bool<sub>2</sub> int<sub>2</sub></i>	<b>or</b>	<i>bool<sub>3</sub> int<sub>3</sub></i>	logical/bitwise inclusive or
<i>bool<sub>1</sub> int<sub>1</sub> bool<sub>2</sub> int<sub>2</sub></i>	<b>xor</b>	<i>bool<sub>3</sub> int<sub>3</sub></i>	logical/bitwise exclusive or
<i>–</i>	<b>true</b>	<i>true</i>	push boolean value <i>true</i>
<i>–</i>	<b>false</b>	<i>false</i>	push boolean value <i>false</i>
<i>int<sub>1</sub> shift</i>	<b>bitshift</b>	<i>int<sub>2</sub></i>	bitwise shift of <i>int<sub>1</sub></i> (positive is left)

## Rectangle Operators (Display)

<i>x y width height</i>	<b>rectfill</b>	-	
<i>numarray</i>	<b>rectfill</b>	-	
<i>numstring</i>	<b>rectfill</b>	-	fill path consisting of one or more rectangles
<i>x y width height</i>	<b>rectstroke</b>	-	
<i>x y width height matrix</i>	<b>rectstroke</b>	-	
<i>numarray</i>	<b>rectstroke</b>	-	
<i>numarray matrix</i>	<b>rectstroke</b>	-	
<i>numstring</i>	<b>rectstroke</b>	-	
<i>numstring matrix</i>	<b>rectstroke</b>	-	stroke path consisting of one or more rectangles
<i>x y width height</i>	<b>rectclip</b>	-	
<i>numarray numstring</i>	<b>rectclip</b>	-	intersect inside of current clipping path with supplied path

## Control Operators

<i>any</i>	<b>exec</b>	-	execute arbitrary object
<i>bool proc</i>	<b>if</b>	-	execute <i>proc</i> if <i>bool</i> is true
<i>bool proc<sub>1</sub> proc<sub>2</sub></i>	<b>ifelse</b>	-	execute <i>proc<sub>1</sub></i> if <i>bool</i> is true, <i>proc<sub>2</sub></i> if <i>bool</i> is false
<i>init incr limit proc</i>	<b>for</b>	-	execute <i>proc</i> with values from <i>init</i> by steps of <i>incr</i> to <i>limit</i>
<i>int proc</i>	<b>repeat</b>	-	execute <i>proc int</i> times
<i>proc</i>	<b>loop</b>	-	execute <i>proc</i> an indefinite number of times
-	<b>exit</b>	-	exit innermost active loop
-	<b>stop</b>	-	terminate <b>stopped</b> context
<i>any</i>	<b>stopped</b>	<i>bool</i>	establish context for catching <b>stop</b>
-	<b>countexecstack</b>	<i>int</i>	count elements on exec stack
<i>array</i>	<b>execstack</b>	<i>subarray</i>	copy exec stack into <i>array</i>
-	<b>quit</b>	-	terminate interpreter
-	<b>start</b>	-	executed at interpreter startup

## Type, Attribute, and Conversion Operators

<i>any</i>	<b>type</b>	<i>name</i>	return name identifying <i>any</i> 's type
<i>any</i>	<b>cvlit</b>	<i>any</i>	make object be literal

<i>any</i>	<b>cvx</b>	<i>any</i>	make object be executable
<i>any</i>	<b>xcheck</b>	<i>bool</i>	test executable attribute
<i>array packedarray file string</i>	<b>executeonly</b>	<i>array packedarray file string</i>	reduce access to execute-only
<i>array packedarray dict file string</i>	<b>noaccess</b>	<i>array packedarray dict file string</i>	disallow any access
<i>array packedarray dict file string</i>	<b>readonly</b>	<i>array packedarray dict file string</i>	reduce access to read-only
<i>array packedarray dict file string</i>	<b>rcheck</b>	<i>bool</i>	test read access
<i>array packedarray dict file string</i>	<b>wcheck</b>	<i>bool</i>	test write access
<i>numstring</i>	<b>cvi</b>	<i>int</i>	convert to integer
<i>string</i>	<b>cvn</b>	<i>name</i>	convert to name
<i>numstring</i>	<b>cvr</b>	<i>real</i>	convert to real
<i>num radix string</i>	<b>cvrs</b>	<i>substring</i>	convert to string with <i>radix</i>
<i>any string</i>	<b>cvs</b>	<i>substring</i>	convert to string

## File Operators

<i>string<sub>1</sub> string<sub>2</sub></i>	<b>file</b>	<i>file</i>	open file identified by <i>string<sub>1</sub></i> with access <i>string<sub>2</sub></i>
<i>string</i>	<b>deletefile</b>	<i>-</i>	remove specified file from device
<i>string<sub>1</sub> string<sub>2</sub></i>	<b>renamefile</b>	<i>-</i>	change file name from <i>string<sub>1</sub></i> to <i>string<sub>2</sub></i>
<i>pattern proc scratch</i>	<b>filenameforall</b>	<i>-</i>	process each file whose name matches <i>pattern</i> with <i>proc</i>
<i>file</i>	<b>closefile</b>	<i>-</i>	close <i>file</i>
<i>file</i>	<b>read</b>	<i>int true</i> or <i>false</i>	read one character from <i>file</i>
<i>file int</i>	<b>write</b>	<i>-</i>	write one character to <i>file</i>
<i>file string</i>	<b>readhexstring</b>	<i>substring bool</i>	read hex from <i>file</i> into <i>string</i>
<i>file string</i>	<b>writehexstring</b>	<i>-</i>	write <i>string</i> to <i>file</i> as hex
<i>file string</i>	<b>readstring</b>	<i>substring bool</i>	read string from <i>file</i>
<i>file string</i>	<b>writestring</b>	<i>-</i>	write characters of <i>string</i> to <i>file</i>
<i>file string</i>	<b>readline</b>	<i>substring bool</i>	read line from <i>file</i> into <i>string</i>

<i>file</i>	<b>token</b>	<i>token true</i> or <i>false</i>	read token from <i>file</i>
<i>file int</i>	<b>setfileposition</b>	–	position next read or write in <i>file</i> to <i>int</i>
<i>file</i>	<b>fileposition</b>	<i>int</i>	return current position in already open file
<i>file</i>	<b>bytesavailable</b>	<i>int</i>	number of bytes available to read
–	<b>flush</b>	–	send buffered data to standard output file
<i>file</i>	<b>flushfile</b>	–	send buffered data or read to EOF
<i>file</i>	<b>resetfile</b>	–	discard buffered characters
<i>file</i>	<b>status</b>	<i>bool</i>	return status of file
<i>string</i>	<b>status</b>	<i>pages bytes referenced created true</i> or <i>false</i>	(Display) return status of file
<i>string</i>	<b>run</b>	–	execute contents of named file
–	<b>currentfile</b>	<i>file</i>	return file currently being executed
<i>string</i>	<b>print</b>	–	write characters of <i>string</i> to standard output file
<i>any</i>	=	–	write text representation of <i>any</i> to standard output file
<b>!–</b> <i>any<sub>1</sub> ... any<sub>n</sub></i>		<b>stack</b>	<b>!–</b> <i>any<sub>1</sub> ... any<sub>n</sub></i>
<i>any</i>	==	–	print stack nondestructively using =
<b>!–</b> <i>any<sub>1</sub> ... any<sub>n</sub></i>		<b>pstack</b>	<b>!–</b> <i>any<sub>1</sub> ... any<sub>n</sub></i>
–	<b>prompt</b>	–	write syntactic representation of <i>any</i> to standard output file
<i>bool</i>	<b>echo</b>	–	print stack nondestructively using ==
–			executed when ready for interactive input
			turn on/off echoing

## Structured Output Operators (Display)

<i>int</i>	<b>setobjectformat</b>	–	set format for object sequences written by <b>printobject</b> and <b>writeobject</b>
–	<b>currentobjectformat</b>	<i>int</i>	return current object format used by <b>printobject</b> and <b>writeobject</b>
<i>obj tag</i>	<b>printobject</b>	–	write binary object sequence to standard output file
<i>file obj tag</i>	<b>writeobject</b>	–	write binary object sequence to <i>file</i>

## Virtual Memory Operators (Display)

–	<b>save</b>	<i>save</i>	create VM snapshot
<i>save</i>	<b>restore</b>	–	restore VM snapshot
<i>dict</i>	<i>key</i>	<b>undef</b>	– remove <i>key</i> and associated value from dict
<i>bool</i>	<b>setshared</b>	–	set private or shared VM allocation mode
–	<b>currentshared</b>	<i>bool</i>	return current value of VM allocation mode
<i>any</i>	<b>scheck</b>	<i>bool</i>	return whether <i>any</i> is sharable
<i>int</i>	<b>vmreclaim</b>	–	control garbage collection mode
–	<b>vmstatus</b>	<i>level used maximum</i>	report VM status

## Context Operators (Display)

<i>mark obj<sub>1</sub> ... obj<sub>n</sub> proc</i>	<b>fork</b>	<i>context</i>	create new context within private VM of current context
<i>context</i>	<b>join</b>	<i>mark obj<sub>1</sub> ... obj<sub>n</sub></i>	when context ceases execution, push its operand stack onto current context's operand stack
<i>context</i>	<b>detach</b>	–	cause context to terminate when it's done executing
–	<b>currentcontext</b>	<i>context</i>	return integer identifying current context
–	<b>lock</b>	<i>lock</i>	create lock
–	<b>condition</b>	<i>condition</i>	create condition object
<i>lock proc</i>	<b>monitor</b>	–	acquire lock, execute <i>proc</i> , and then release lock
<i>lock condition</i>	<b>wait</b>	–	release lock, wait for condition, and reacquire lock
<i>condition</i>	<b>notify</b>	–	resume execution of contexts waiting for condition
–	<b>yield</b>	–	suspend current context until other contexts sharing same VM have executed

## Miscellaneous Operators

<i>proc</i>	<b>bind</b>	<i>proc</i>	replace operator names in <i>proc</i> by operators
<i>index name</i>	<b>defineusername</b>	–	(Display) associate <i>index</i> with <i>name</i> in user name table

–	<b>null null</b>	push null on operand stack
–	<b>usertime int</b>	(Display) return PostScript interpreter execution time
–	<b>realtime int</b>	(Display) return value of clock that counts in real time
–	<b>version string</b>	interpreter version
–	<b>nextrelease string</b>	(NeXTSTEP) NeXTSTEP version information
<i>bool</i>	<b>setwriteblock –</b>	(NeXTSTEP) set whether Window Server blocks
–	<b>currentwriteblock bool</b>	(NeXTSTEP) return whether Server blocks
–	<b>currentuser uid gid</b>	(NeXTSTEP) return user id and group id of currently logged-in user
–	<b>currentrusage ctime utime stime msgsend msgrcv nsignals nvcs w nivcs w</b>	(NeXTSTEP) report Window Server's resource usage
<i>soundname priority</i>	<b>playsound –</b>	(NeXTSTEP) play <i>soundname</i> at given priority level

## Graphics State Operators

–	<b>gsave –</b>	save graphics state
–	<b>grestore –</b>	restore graphics state
–	<b>grestoreall –</b>	restore to bottommost graphics state
–	<b>initgraphics –</b>	(standard) reset graphics state parameters (NeXTSTEP) also reset alpha and instancing
<i>num</i>	<b>setlinewidth –</b>	set line width
–	<b>currentlinewidth num</b>	return current line width
<i>int</i>	<b>setlinecap –</b>	set shape of line ends for stroke (0=butt, 1=round, 2=square)
–	<b>currentlinecap int</b>	return current line cap
<i>int</i>	<b>setlinejoin –</b>	set shape of corners for stroke (0=miter, 1=round, 2=bevel)
–	<b>currentlinejoin int</b>	return current line join
<i>num</i>	<b>setmiterlimit –</b>	set miter length limit
–	<b>currentmiterlimit num</b>	return current miter limit

<i>array offset</i>	<b>setdash</b> –	set dash pattern for stroking
	– <b>currentdash</b> <i>array offset</i>	return current dash pattern
<i>num</i>	<b>setflat</b> –	set flatness tolerance
	– <b>currentflat</b> <i>num</i>	return current flatness
<i>patternname</i>	<b>setpattern</b> –	(NeXTSTEP) set pattern for drawing
<i>num</i>	<b>setgray</b> –	set color to gray value from 0 (black) to 1 (white)
	– <b>currentgray</b> <i>num</i>	return current gray
<i>hue sat brt</i>	<b>sethsbcolor</b> –	set color given hue, saturation, brightness
	– <b>currenthsbcolor</b> <i>hue sat brt</i>	return current color hue, saturation, brightness
<i>red green blue</i>	<b>setrgbcolor</b> –	set color given red, green, blue
	– <b>currentrgbcolor</b> <i>red green blue</i>	return current color red, green, blue
<i>coverage</i>	<b>setalpha</b> –	(NeXTSTEP) set current coverage
	– <b>currentalpha</b> <i>coverage</i>	(NeXTSTEP) return current coverage setting
<i>cyan magenta yellow black</i>	<b>setcmykcolor</b> –	set current color parameter in graphics state
	– <b>currentcmykcolor</b> <i>cyan magenta yellow black</i>	return current color parameter in graphics state
<i>redproc greenproc</i>		
<i>blueproc grayproc</i>	<b>setcolortransfer</b> –	set current transfer function parameters for specified colors
	– <b>currentcolortransfer</b> <i>redproc greenproc blueproc grayproc</i>	return current transfer function parameters for specified colors
<i>proc</i>	<b>setblackgeneration</b> –	set current black generation function parameter in graphics state
	– <b>currentblackgeneration</b> <i>proc</i>	return current black generation function parameter in graphics state
<i>proc</i>	<b>setundercolorremoval</b> –	set current undercolor removal function parameter in graphics state
	– <b>currentundercolorremoval</b> <i>proc</i>	return current undercolor removal function parameter in graphics state

<i>redfrequency redangle redproc</i>		
<i>greenfrequency greenangle</i>		
<i>greenproc bluefrequency</i>		
<i>blueangle blueproc grayfrequency</i>	<b>setcolorscreen</b>	–
<i>grayangle grayproc</i>		set all twelve current halftone screen parameters in graphics state
	– <b>currentcolorscreen</b>	<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</i>		
<i>proc<sub>0</sub> [... proc<sub>n</sub>] multiproc ncolors</i>	<b>colorimage</b>	–
		render sampled image with 1, 3, or 4 color values
<i>freq angle proc</i>	<b>setscreen</b>	–
<i>freq angle halftone</i>	<b>setscreen</b>	–
	– <b>currentscreen</b>	<i>freq angle proc</i>
	– <b>currentscreen</b>	<i>60 0 halftone</i>
<i>proc</i>	<b>settransfer</b>	–
	– <b>currenttransfer</b>	<i>proc</i>
		set gray transfer function
		return current transfer function

### Graphics State Object Operators (Display)

– <b>gstate</b>	<i>gstate</i>	create graphics state object
<i>gstate</i>	<b>setgstate</b>	– replace current graphics state by value of <i>gstate</i>
<i>gstate</i>	<b>currentgstate</b>	<i>gstate</i>

### Halftone Definition Operators (Display)

<i>dict</i>	<b>sethalftone</b>	–	establish <i>dict</i> as current halftone dictionary
	– <b>currenthalftone</b>	<i>dict</i>	return current halftone dictionary
<i>x y</i>	<b>sethalftonephase</b>	–	(Display) set current halftone phase parameters
	– <b>currenthalftonephase</b>	<i>x y</i>	(Display) return current halftone phase parameters

## Coordinate System and Matrix Operators

<i>x</i> <i>y</i>	<b>basetocurrent</b>	<i>x'</i> <i>y'</i>	(NeXTSTEP) convert from base to current coordinate system
<i>x</i> <i>y</i>	<b>basetoscreen</b>	<i>x'</i> <i>y'</i>	(NeXTSTEP) convert from base to screen coordinate system
<i>x</i> <i>y</i>	<b>currenttobase</b>	<i>x'</i> <i>y'</i>	(NeXTSTEP) convert from current to base coordinate system
<i>x</i> <i>y</i>	<b>currenttoscreen</b>	<i>x'</i> <i>y'</i>	(NeXTSTEP) convert from current to screen coordinate system
<i>x</i> <i>y</i>	<b>screentobase</b>	<i>x'</i> <i>y'</i>	(NeXTSTEP) convert from screen to base coordinate system
<i>x</i> <i>y</i>	<b>screentocurrent</b>	<i>x'</i> <i>y'</i>	(NeXTSTEP) convert from screen to current coordinate system
–	<b>matrix</b>	<i>matrix</i>	create identity matrix
–	<b>initmatrix</b>	–	set CTM to device default
<i>matrix</i>	<b>identmatrix</b>	<i>matrix</i>	fill matrix with identity transform
<i>matrix</i>	<b>defaultmatrix</b>	<i>matrix</i>	fill matrix with device default matrix
<i>matrix</i>	<b>currentmatrix</b>	<i>matrix</i>	fill matrix with CTM
<i>matrix</i>	<b>setmatrix</b>	–	replace CTM by matrix
<i>t<sub>x</sub></i> <i>t<sub>y</sub></i>	<b>translate</b>	–	translate user space by ( <i>t<sub>x</sub></i> , <i>t<sub>y</sub></i> )
<i>t<sub>x</sub></i> <i>t<sub>y</sub></i> <i>matrix</i>	<b>translate</b>	<i>matrix</i>	define translation by ( <i>t<sub>x</sub></i> , <i>t<sub>y</sub></i> )
<i>s<sub>x</sub></i> <i>s<sub>y</sub></i>	<b>scale</b>	–	scale user space by <i>s<sub>x</sub></i> and <i>s<sub>y</sub></i>
<i>s<sub>x</sub></i> <i>s<sub>y</sub></i> <i>matrix</i>	<b>scale</b>	<i>matrix</i>	define scaling by <i>s<sub>x</sub></i> and <i>s<sub>y</sub></i>
<i>angle</i>	<b>rotate</b>	–	rotate user space by <i>angle</i> degrees
<i>angle</i> <i>matrix</i>	<b>rotate</b>	<i>matrix</i>	define rotation by <i>angle</i> degrees
<i>matrix</i>	<b>concat</b>	–	replace CTM by <i>matrix</i> × CTM
<i>matrix<sub>1</sub></i> <i>matrix<sub>2</sub></i> <i>matrix<sub>3</sub></i>	<b>concatmatrix</b>	<i>matrix<sub>3</sub></i>	fill <i>matrix<sub>3</sub></i> with <i>matrix<sub>1</sub></i> × <i>matrix<sub>2</sub></i>
<i>x</i> <i>y</i>	<b>transform</b>	<i>x'</i> <i>y'</i>	transform ( <i>x</i> , <i>y</i> ) by CTM
<i>x</i> <i>y</i> <i>matrix</i>	<b>transform</b>	<i>x'</i> <i>y'</i>	transform ( <i>x</i> , <i>y</i> ) by matrix
<i>dx</i> <i>dy</i>	<b>dtransform</b>	<i>dx'</i> <i>dy'</i>	transform distance ( <i>dx</i> , <i>dy</i> ) by CTM
<i>dx</i> <i>dy</i> <i>matrix</i>	<b>dtransform</b>	<i>dx'</i> <i>dy'</i>	transform distance ( <i>dx</i> , <i>dy</i> ) by matrix
<i>x'</i> <i>y'</i>	<b>itransform</b>	<i>x</i> <i>y</i>	inverse transform ( <i>x'</i> , <i>y'</i> ) by CTM
<i>x'</i> <i>y'</i> <i>matrix</i>	<b>itransform</b>	<i>x</i> <i>y</i>	inverse transform ( <i>x'</i> , <i>y'</i> ) by matrix
<i>dx'</i> <i>dy'</i>	<b>idtransform</b>	<i>dx</i> <i>dy</i>	inverse transform distance ( <i>dx'</i> , <i>dy'</i> ) by CTM
<i>dx'</i> <i>dy'</i> <i>matrix</i>	<b>idtransform</b>	<i>dx</i> <i>dy</i>	inverse transform distance ( <i>dx'</i> , <i>dy'</i> ) by matrix
<i>matrix<sub>1</sub></i> <i>matrix<sub>2</sub></i>	<b>invertmatrix</b>	<i>matrix<sub>2</sub></i>	fill <i>matrix<sub>2</sub></i> with inverse of <i>matrix<sub>1</sub></i>

## Path Construction Operators

<code>- newpath -</code>	initialize current path to be empty
<code>- currentpoint <i>x y</i></code>	return current point coordinate
<code><i>x y</i> moveto -</code>	set current point to ( <i>x, y</i> )
<code><i>dx dy</i> rmoveto -</code>	relative <b>moveto</b>
<code><i>x y</i> lineto -</code>	append straight line to ( <i>x, y</i> )
<code><i>dx dy</i> rlineto -</code>	relative <b>lineto</b>
<code><i>x y r ang<sub>1</sub> ang<sub>2</sub></i> arc -</code>	append counterclockwise arc
<code><i>x y r ang<sub>1</sub> ang<sub>2</sub></i> arcn -</code>	append clockwise arc
<code><i>x<sub>1</sub> y<sub>1</sub> x<sub>2</sub> y<sub>2</sub> r</i> arcto <i>xt<sub>1</sub> yt<sub>1</sub> xt<sub>2</sub> yt<sub>2</sub></i></code>	append tangent arc
<code><i>x<sub>1</sub> y<sub>1</sub> x<sub>2</sub> y<sub>2</sub> x<sub>3</sub> y<sub>3</sub></i> curveto -</code>	append Bezier cubic section
<code><i>dx<sub>1</sub> dy<sub>1</sub> dx<sub>2</sub> dy<sub>2</sub> dx<sub>3</sub> dy<sub>3</sub></i> rcurveto -</code>	relative <b>curveto</b>
<code>- closepath -</code>	connect subpath back to its starting point
<code>- flattenpath -</code>	convert curves to sequences of straight lines
<code>- reversepath -</code>	reverse direction of current path
<code>- strokepath -</code>	compute outline of stroked path
<code><i>string bool</i> charpath -</code>	append character outline to current path
<code>- clippath -</code>	set current path to clipping path
<code>- pathbbox <i>ll<sub>x</sub> ll<sub>y</sub> ur<sub>x</sub> ur<sub>y</sub></i></code>	return bounding box of current path
<code><i>move line curve close</i> pathforall -</code>	enumerate current path
<code>- initclip -</code>	set clipping path to device default
<code>- clip -</code>	establish new clipping path
<code>- eoclip -</code>	clip using even-odd inside rule

## User Path Operators (Display)

<code><i>ll<sub>x</sub> ll<sub>y</sub> ur<sub>x</sub> ur<sub>y</sub></i> setbbox -</code>	establish bounding box for current path
<code><i>x<sub>1</sub> y<sub>1</sub> x<sub>2</sub> y<sub>2</sub> r</i> arct -</code>	append arc of circle to current path
<code><i>userpath</i> uappend -</code>	append <i>userpath</i> to current path
<code><i>bool</i> upath <i>userpath</i></code>	create <i>userpath</i> as copy of current path

<i>userpath</i>	<b>ufill</b>	–	fill <i>userpath</i> as if by using fill operator
<i>userpath</i>	<b>ueofill</b>	–	fill <i>userpath</i> as if by using eofill operator
<i>userpath</i>	<b>ustroke</b>	–	stroke <i>userpath</i> as if by using stroke operator
<i>userpath matrix</i>	<b>ustroke</b>	–	
<i>userpath</i>	<b>ustrokepath</b>	–	
<i>userpath matrix</i>	<b>ustrokepath</b>	–	replace current path with <i>userpath</i> and stroke result
–	<b>ucache</b>	–	store enclosing user path if not already stored
–	<b>ucachestatus</b>	<i>mark bsize bmax rsize rmax blimit</i>	report status of user path cache
<i>mark blimit</i>	<b>setucacheparams</b>	–	set user path cache parameters

## View Clip Operators (Display)

–	<b>viewclip</b>	–	replace view clipping path with copy of current path
–	<b>eoviewclip</b>	–	replace view clipping path with current path using even-odd inside rule
<i>x y width height</i>	<b>rectviewclip</b>	–	
<i>numarray numstring</i>	<b>rectviewclip</b>	–	replace view clipping path with specified path
–	<b>viewclippath</b>	–	replace path with copy of current view clipping path
–	<b>initviewclip</b>	–	replace view clipping path with one equal to imageable area

## Painting Operators

–	<b>erasepage</b>	–	(NeXTSTEP) erase entire window to opaque white (standard) paint current page white
–	<b>fill</b>	–	fill current path with current color
–	<b>eofill</b>	–	fill using even-odd rule
–	<b>stroke</b>	–	draw line along current path
<i>width height bits/sample matrix proc</i>	<b>image</b>	–	render sampled image onto current page
<i>width height invert matrix proc</i>	<b>imagemask</b>	–	render mask onto current page

## Window System Support Operators (Display)

– **wtranslation** *xy*

return translation from window origin to device space origin

*xy infill bool*  
*userpath infill bool*

return *true* if pixel at (*x*, *y*) (or any pixels in *userpath*) would be painted by fill of current path

*xy ineofill bool*  
*userpath ineofill bool*

return *true* if pixel at (*x*, *y*) (or any pixels in *userpath*) would be painted by eofill of current path

*xy userpath inufill bool*  
*userpath<sub>1</sub> userpath<sub>2</sub> inufill bool*

return *true* if pixel at (*x*, *y*) (or any pixels in *userpath<sub>1</sub>*) would be painted by ufill of current path

*xy userpath inueofill bool*  
*userpath<sub>1</sub> userpath<sub>2</sub> inueofill bool*

return *true* if pixel at (*x*, *y*) (or any pixels in *userpath<sub>1</sub>*) would be painted by ueofill of current path

*xy instroke bool*  
*userpath instroke bool*

return *true* if pixel at (*x*, *y*) (or any pixels in *userpath*) would be painted by stroke of current path

*xy userpath inustroke bool*  
*xy userpath matrix inustroke bool*  
*userpath<sub>1</sub> userpath<sub>2</sub> inustroke bool*  
*userpath<sub>1</sub> userpath<sub>2</sub> matrix inustroke bool*

return *true* if pixel at (*x*, *y*) (or any pixels in *userpath<sub>1</sub>*) would be painted by ustroke of current path

– **deviceinfo** *dict*

return *dict* containing static information about current device

## Device Setup and Output Operators

**proc** *window* **setshowpageprocedure** –

set the procedure that's executed during showpage

*window* **currentshowpageprocedure** *proc*

return the procedure that's executed during showpage

– **showpage** –

output and reset current page

<i>width height bbox matrix</i>	<b>copypage</b> –	output current page
<i>hostname portname pixelencoding</i>	<b>nulldevice</b> –	install no-output device
	<b>machportdevice</b> –	(NeXTSTEP) set up PostScript device for generic rendering service

## Scan Conversion Operators (Display)

<i>bool</i>	<b>setstrokeadjust</b> –	turn automatic stroke adjustment on or off
	<b>currentstrokeadjust</b> <i>bool</i>	return current state of automatic stroke adjustment

## Character and Font Operators

<i>key font</i>	<b>definefont</b> <i>font</i>	register <i>font</i> as font dictionary
<i>key</i>	<b>undefinefont</b> –	(Display) remove <i>key</i> from FontDictionary dictionary
<i>key</i>	<b>findfont</b> <i>font</i>	return font dict identified by <i>key</i>
<i>font scale</i>	<b>scalefont</b> <i>font'</i>	scale <i>font</i> by <i>scale</i> to produce new <i>font'</i>
<i>font matrix</i>	<b>makefont</b> <i>font'</i>	transform <i>font</i> by <i>matrix</i> to produce new <i>font'</i>
<i>font</i>	<b>setfont</b> –	set font dictionary
	<b>currentfont</b> <i>font</i>	return current font dictionary
<i>string</i>	<b>show</b> –	print characters of <i>string</i> on page
<i>a<sub>x</sub> a<sub>y</sub> string</i>	<b>ashow</b> –	add ( <i>a<sub>x</sub>, a<sub>y</sub></i> ) to width of each char while showing <i>string</i>
<i>text numarray numstring</i>	<b>xyshow</b> –	(Display) print characters according to x, y displacements in <i>numarray</i> or <i>numstring</i>
<i>text numarray numstring</i>	<b>xshow</b> –	(Display) print characters according to x displacements in <i>numarray</i> or <i>numstring</i>
<i>text numarray numstring</i>	<b>yshow</b> –	(Display) print characters according to y displacements in <i>numarray</i> or <i>numstring</i>
<i>c<sub>x</sub> c<sub>y</sub> char string</i>	<b>widthshow</b> –	add ( <i>c<sub>x</sub>, c<sub>y</sub></i> ) to width of <i>char</i> while showing <i>string</i>
<i>c<sub>x</sub> c<sub>y</sub> char a<sub>x</sub> a<sub>y</sub> string</i>	<b>awidthshow</b> –	combine effects of <b>ashow</b> and <b>widthshow</b>
<i>proc string</i>	<b>kshow</b> –	execute <i>proc</i> between characters shown from <i>string</i>

<i>string</i>	<b>stringwidth</b>	<i>w<sub>x</sub> w<sub>y</sub></i>	width of <i>string</i> in current font
–	<b>FontDirectory</b>	<i>dict</i>	dictionary of font dictionaries
–	<b>SharedFontDirectory</b>	<i>dict</i>	(Display) dictionary of font dictionaries
–	<b>NextStepEncoding</b>	<i>array</i>	NeXTSTEP font encoding vector
–	<b>StandardEncoding</b>	<i>array</i>	standard font encoding vector
<i>key scale matrix</i>	<b>selectfont</b>	–	(Display) establish font specified by <i>key</i> as current font

## Font Cache Operators

–	<b>cachestatus</b>	<i>bsize bmax msize mmax csize cmax blimit</i>	return cache status and parameters
<i>w<sub>x</sub> w<sub>y</sub> ll<sub>x</sub> ll<sub>y</sub> ur<sub>x</sub> ur<sub>y</sub></i>	<b>setcachedevice</b>	–	declare cached character metrics
<i>w<sub>x</sub> w<sub>y</sub></i>	<b>setcharwidth</b>	–	declare uncached character metrics
<i>num</i>	<b>setcachelimit</b>	–	set max bytes in cached character
<i>mark size lower upper</i>	<b>setcacheparams</b>	–	set character cache parameters
–	<b>currentcacheparams</b>	<i>mark lower upper</i>	return current font cache parameters

## User Object Encoding Operators (Display)

<i>index any</i>	<b>defineuserobject</b>	–	associate <i>index</i> with an object in the user object array
<i>index</i>	<b>undefineuserobject</b>	–	remove associate between <i>index</i> and the object it referred to
<i>index</i>	<b>execuserobject</b>	–	execute object referred to by <i>index</i>

## Errors

<b>dictfull</b>	no more room in dictionary
<b>dictstackoverflow</b>	too many begins
<b>dictstackunderflow</b>	too many ends
<b>execstackoverflow</b>	exec nesting too deep
<b>handleerror</b>	called to report error information
<b>interrupt</b>	external interrupt request (e.g., Control-C)

<b>invalidaccess</b>	attempt to violate access attribute
<b>invalidcontext</b>	invalid use of context synchronization facilities
<b>invalidexit</b>	exit not in loop
<b>invalidfileaccess</b>	unacceptable access string
<b>invalidfont</b>	invalid font name or dict
<b>invalidrestore</b>	improper <b>restore</b>
<b>invalidid</b>	(Display) invalid identifying number as operand
<b>ioerror</b>	input/output error occurred
<b>limitcheck</b>	implementation limit exceeded
<b>nocurrentpoint</b>	current point is undefined
<b>rangecheck</b>	operand out of bounds
<b>stackoverflow</b>	operand stack overflow
<b>stackunderflow</b>	operand stack underflow
<b>syntaxerror</b>	syntax error in PS program
<b>timeout</b>	time limit exceeded
<b>typecheck</b>	operand of wrong type
<b>undefined</b>	name not known
<b>undefinedfilename</b>	file not found
<b>undefinedresult</b>	over/underflow or meaningless result
<b>unmatchedmark</b>	expected mark not on stack
<b>unregistered</b>	internal error
<b>VMerror</b>	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      PSarect(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 *pbsize, 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 destx,
                  float desty, int op)†
void    PScompositerect(float destx, float desty, 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    PSshow(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 *ewidth)
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    PScurrenttrgbcolor(float *pr, float *pg, float *pb)
void    PScurrentusage(float *pnnow, 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	<b>PScurrentwindowlevel</b> (userobject <i>window</i> , int * <i>plevel</i> ) <sup>†</sup>
void	<b>PScurrentwriteblock</b> (int * <i>pflag</i> ) <sup>†</sup>
void	<b>PScurvetoto</b> (float <i>x<sub>1</sub></i> , float <i>y<sub>1</sub></i> , float <i>x<sub>2</sub></i> , float <i>y<sub>2</sub></i> , float <i>x<sub>3</sub></i> , float <i>y<sub>3</sub></i> )
void	<b>PScvi</b> (void)
void	<b>PScvlit</b> (void)
void	<b>PScvn</b> (void)
void	<b>PScvr</b> (void)
void	<b>PScvrs</b> (void)
void	<b>PScvs</b> (void)
void	<b>PScvx</b> (void)
void	<b>PSdef</b> (void)
void	<b>PSdefaultmatrix</b> (void)
void	<b>PSdefinefont</b> (void)
void	<b>PSdefineresource</b> (char * <i>category</i> )
void	<b>PSdefineusername</b> (int <i>index</i> , char * <i>name</i> )
void	<b>PSdefineuserobject</b> (void) <b>Warning:</b> Use <b>DPSDefineUserObject()</b> instead.
void	<b>PSdeletefile</b> (char * <i>filename</i> )
void	<b>PSdetach</b> (void)
void	<b>PSdeviceinfo</b> (void)
void	<b>PSdict</b> (int <i>length</i> )
void	<b>PSdictstack</b> (void)
void	<b>PSdissolve</b> (float <i>src<sub>x</sub></i> , float <i>src<sub>y</sub></i> , float <i>width</i> , float <i>height</i> , userobject <i>srcGstate</i> , float <i>dest<sub>x</sub></i> , float <i>dest<sub>y</sub></i> , float <i>delta</i> ) <sup>†</sup>
void	<b>PSdiv</b> (void)
void	<b>PSdtransform</b> (float <i>x</i> , float <i>y</i> , float * <i>px</i> , float * <i>py</i> )
void	<b>PSdumpwindow</b> (int <i>level</i> , userobject <i>window</i> ) <b>Warning:</b> Don't use this function if you're using the Application Kit.
void	<b>PSdumpwindows</b> (int <i>level</i> , userobject <i>context</i> ) <b>Warning:</b> Don't use this function if you're using the Application Kit.
void	<b>PSdup</b> (void)
void	<b>PSecho</b> (boolean <i>flag</i> )
void	<b>PSend</b> (void)
void	<b>PSeoclip</b> (void)

```

void    PSoeffill(void)
void    PSeoviewclip(void)
void    PSeq(void)
void    PSequals(void)
void    PSequalsquals(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 *ewidth, int *eheight, int *edepth)†
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    PSineofill(float x, float y, boolean *pflag)
void    PSinfill(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    PSstroke(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 *pllx, float *plly, float *purx, float *pury)
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    PSrcurvetoto(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    PSrand(void)
void    PSrun(char *name)
void    PSsave(void)
void    PSscale(float sx, float sy)
void    PSscalefont(float size)
void    PScheck(boolean *pflag)
void    PSscreenlist(int context, int count, int windows[])†
void    PSscreenbase(float x, float y, float *px, float *py)†
void    PScurrent(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 llx, float lly, float urx, float ury)
void    PSsetblackgeneration(void)
void    PSsetcachedevice(float wx, float wy, float llx, float lly, float urx, float ury)
void    PSsetcachelimit(float num)
void    PSsetcacheparams(void)
void    PSsetcharwidth(float wx, float wy)
void    PSsetcmykcolor(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    PSshareddict(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    PSrand(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    PSermwindow(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    PSstroke(char nums[], int n, char ops[], int l)
void    PSstrokepath(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*)<sup>†</sup>

void **PSwindowdeviceround**(userobject *window*)<sup>†</sup>

void **PSwindowlist**(int *context*, int *count*, int *windows*[])<sup>†</sup>

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	<b>DPSCreateContext</b> (const char * <i>hostName</i> , const char * <i>serverName</i> , DPSTextProc <i>textProc</i> , DPSErrorProc <i>errorProc</i> ) <sup>†</sup>
DPSContext	<b>DPSCreateContextWithTimeoutFromZone</b> (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> ) <sup>†</sup>
DPSContext	<b>DPSCreateNonsecureContext</b> (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> ) <sup>†</sup>
DPSContext	<b>DPSCreateStreamContext</b> (NXStream * <i>stream</i> , int <i>debugging</i> , DPSProgramEncoding <i>progEnc</i> , DPSNameEncoding <i>nameEnc</i> , DPSErrorProc <i>errorProc</i> ) <sup>†</sup>
void	<b>DPSDestroyContext</b> (DPSContext <i>context</i> )

### Create a child context

int	<b>DPSChainContext</b> (DPSContext <i>parent</i> , DPSContext <i>child</i> )
void	<b>DPSUnchainContext</b> (DPSContext <i>context</i> )

### Access the current context

void	<b>DPSSetContext</b> (DPSContext <i>context</i> )
DPSContext	<b>DPSGetCurrentContext</b> (void)

### Control a context

int	<b>DPSynchronizeContext</b> (DPSContext <i>context</i> , int <i>enableFlag</i> ) <sup>†</sup>
void	<b>DPSWaitContext</b> (DPSContext <i>context</i> )
void	<b>DPSAsynchronousWaitContext</b> (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	<b>DPSInterruptContext()</b>
void	<b>DPSResetContext()</b>

### **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	<b>DPSGetEvent(DPSContext context, NXEvent *anEvent, int mask, double timeout,                   int threshold)</b> <sup>†</sup>
int	<b>DPS.PeekEvent(DPSContext context, NXEvent *anEvent, int mask, double timeout,                   int threshold)</b> <sup>†</sup>
void	<b>DPSDiscardEvents(DPSContext context, int mask)</b> <sup>†</sup>

## Coalesce events

int	<b>DPSSetTracking(int flag)</b> <sup>†</sup>
-----	--

## Set the event-filter function

DPSEventFilterFunc	<b>DPSSetEventFunc(DPSContext context, DPSEventFilterFunc func)</b> <sup>†</sup>
--------------------	--

## Create an event

int	<b>DPSPostEvent(NXEvent *anEvent, int atStart)</b> <sup>†</sup>
-----	---

## Create a timed entry

DPSTimedEntry	<b>DPSAddTimedEntry(double period, DPSTimedEntryProc handler,                   void *userData, int priority)</b> <sup>†</sup>
void	<b>DPSRemoveTimedEntry(DPSTimedEntry teNumber)</b> <sup>†</sup>

## Initiate a count down for the wait cursor

void	<b>DPSStartWaitCursorTimer(void)</b> <sup>†</sup>
------	---

## Allow dead key processing

void	<b>DPSSetDeadKeysEnabled(DPSContext context, int flag)</b> <sup>†</sup>
------	---

## Generate an event mask for an event type

int	<b>NX_EVENTCODEMASK(int type)</b>
-----	-----------------------------------

# File and Port Monitoring

## Monitor a file descriptor

```
void          DPSAddFD(int fd, DPSFDProc handler, void *userData, int priority)†  
void          DPSRemoveFD(int fd)†
```

## Monitor a Mach port

```
void          DPSAddPort(port_t newPort, DPSPortProc handler, int maxSize, void *userData,  
                         int priority)†  
void          DPSRemovePort(port_t port)†
```

## Set the notify port call-back function

```
void          DPSAddNotifyPortProc(DPSPortProc handler, void *userData)†  
void          DPSRemoveNotifyPortProc(DPSPortProc handler)†
```

# Text-Handling

## Set the text call-back functions

```
DPSTextProc   DPSSSetTextProc(DPSContext context, DPSTextProc tp)  
DPSTextProc   DPSSSetTextBackstop(DPSTextProc textProc)  
DPSTextProc   DPSGetCurrentTextBackstop(void)
```

# Debugging and Error-Handling

## Trace data and events

```
int           DPSTraceContext(DPSContext context, int flag)†  
void          DPSTraceEvents(DPSContext context, int flag)†
```

## Handle errors

DPSErrorProc	<b>DPSSetErrorProc(DPSContext <i>context</i>, DPSErrorProc <i>ep</i>)</b>
void	<b>DPSDefaultErrorProc(DPSContext <i>context</i>, DPSErrorCode <i>errorCode</i>, long unsigned int <i>arg</i><sub>1</sub>, long unsigned int <i>arg</i><sub>2</sub>)</b>
void	<b>DPSSetErrorBackstop(DPSErrorProc <i>errorProc</i>)</b>
DPSErrorProc	<b>DPSGetCurrentErrorBackstop(void)</b>
void	<b>DPSPrintError(FILE *<i>fp</i>, const DPSBinObjSeqRec <i>error</i>)<sup>†</sup></b>
void	<b>DPSPrintErrorToStream(NXStream *<i>stream</i>, const DPSBinObjSeqRec <i>error</i>)<sup>†</sup></b>

## Functions Used by pswrap

### Wait for return values from the Window Server

void **DPSAwaitReturnValues(DPSContext *context*)**

### Write strings in binary object sequence

void **DPSWriteStringChars(DPSContext *context*, const char \**buf*, unsigned int *count*)**

### Write PostScript code in a specified format

void **DPSWriteTypedObjectArray(DPSContext *context*, DPSDefinedType *type*, const void \**array*, unsigned int *length*)**

### Begin a new binary object sequence

void **DPSBinObjSeqWrite(DPSContext *context*, const void \**buf*, unsigned int *count*)**

### Define information expected from the PostScript interpreter

void **DPSSetResultTable(DPSContext *context*, DPSResults *table*, unsigned int *length*)**

### Update a context's name map from the client library's name map

void **DPSUpdateNameMap(DPSContext *context*)**

# Types and Constants

## Defined Types

### DPSContextRec

```
typedef struct _t_DPSContextRec {  
    char *priv;  
    DPSSpace space;  
    DPSProgramEncoding programEncoding;  
    DPSNameEncoding nameEncoding;  
    struct _t_DPSProcsRec const * procs;  
    void (*textProc)();  
    void (*errorProc)();  
    DPSResults resultTable;  
    unsigned int resultTableLength;  
    struct _t_DPSContextRec *chainParent, *chainChild;  
    DPSContextType type;  
} DPSContextRec, *DPSContext;
```

### DPSContextType

```
typedef enum {  
    dps_machServer,  
    dps_fdServer,  
    dps_stream  
} DPSContextType;
```

### 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\_DINGBATSSET

## **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

<b>Event Types</b>	<b>Meaning</b>
NX_NULLEVENT	A non-event
NX_LMOUSEDOWN	Left mouse-down
NX_LMOUSEUP	Left mouse-up
NX_LMOUSEDRAGGED	left mouse-dragged
NX_MOUSEDOWN	Same as NX_LMOUSEDOWN
NX_MOUSEUP	Same as NX_LMOUSEUP
NX_MOUSEDRAGGED	Same as NX_LMOUSEDRAGGED
NX_RMOUSEDOWN	Right mouse-down
NX_RMOUSEUP	Right mouse-up
NX_RMOUSEDRAGGED	Right mouse-dragged
NX_MOUSEMOVED	Mouse-moved
NX_MOUSEENTERED	Mouse-entered
NX_MOUSEEXITED	Mouse-exited
NX_KEYDOWN	Key-down
NX_KEYUP	Key-up event
NX_FLAGSCHANGED	Flags-changed
NX_KITDEFINED	Appkit-defined
NX_SYSDEFINED	System-defined
NX_APPDEFINED	Application-defined
NX_TIMER	Timer used for tracking
NX_CURSORUPDATE	Cursor tracking
NX_JOURNALEVENT	Event used by journaling

<code>NX_FIRSTEVENT</code>	The smallest-valued event constant
<code>NX_LASTEVENT</code>	The greatest-valued event constant
<code>NX_ALLEVENTS</code>	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_CURSORUPDATEMASK`  
`NX_MOUSEDOWNMASK`  
`NX_MOUSEUPMASK`  
`NX_MOUSEDRAGGEDMASK`  
`NX_JOURNALEVENTMASK`

### **Forever**

`NX_FOREVER`

<b>Keyboard State Flags Masks</b>	<b>Meaning</b>
NX_ALPHASHIFTMASK	Shift lock
NX_SHIFTMASK	Shift key
NX_CONTROLMASK	Control key
NX_ALTERNATEMASK	Alt key
NX_COMMANDMASK	Command key
NX_NUMERICPADMASK	Number pad key
NX_HELPMASK	Help key
NX_NEXTCTRLKEYMASK	Control key
NX_NEXTLSHIFTKEYMASK	Left shift key
NX_NEXTRSHIFTKEYMASK	Right shift key
NX_NEXTLCMDKEYMASK	Left command key
NX_NEXTRCMDKEYMASK	Right command key
NX_NEXTLALTKEYMASK	Left alt key
NX_NEXTRALTKEYMASK	Right alt key

  

<b>Event Flags Masks</b>	<b>Meaning</b>
NX_STYLUSPROXIMITYMASK	Stylus is in proximity (for tablets)
NX_NONCOALSECSEDMASK	Event coalescing disabled

  

<b>Window Backing Types</b>
NX_RETAINED
NX_NONRETAINED
NX_BUFFERED

  

<b>Window Screen List Placement</b>
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 *) <i>aPort</i> withInPort:(NXPort *) <i>inPort</i>	Connects over the specified port
+ (NXProxy *)connectToPort:(NXPort *) <i>aPort</i> withInPort:(NXPort *) <i>inPort</i> fromZone:(NXZone *) <i>z</i>	Connects over the specified port

## Ascertaining Connections

+ connections:(List *) <i>l</i>	Returns all connections
---------------------------------	-------------------------

## Registering an Object

+ registerRoot: <i>anObject</i>	Establishes a root object
+ registerRoot: <i>anObject</i> fromZone:(NXZone *) <i>z</i>	Establishes a root object
+ registerRoot: <i>anObject</i> withName:(const char *) <i>n</i>	Establishes a named root object
+ registerRoot: <i>anObject</i> withName:(const char *) <i>n</i> fromZone:(NXZone *) <i>z</i>	Establishes a named root object

## Eliminating References

+ removeObject: <i>anObject</i>	Removes an object from all connections
---------------------------------	--

## Invalidation

+ unregisterForInvalidationNotification: <i>anObject</i>	Unregisters an object for notifications
---	---

## Statistics

+ (int)messagesReceived	Number of messages received over connection
-------------------------	---

## Timeouts

+ setDefaultTimeout:(int) <i>t</i>	Sets default timeout for all connections
+ (int)defaultTimeout	Returns default timeout for all connections
- setInTimeout:(int) <i>t</i>	Sets in timeout
- setOutTimeout:(int) <i>t</i>	Sets out timeout
- (int)inTimeout	Returns in timeout
- (int)outTimeout	Returns out timeout

## **Zone Usage**

+ <b>setDefaultZone:</b> (NXZone *) <i>zone</i>	Sets default zone for all connections
- (NXZone *) <b>defaultZone</b>	Returns default zone for all connections

## **Assigning a Delegate**

- <b>setDelegate:</b> <i>anObject</i>	Sets the connection's delegate
- <b>delegate</b>	Returns the connection's delegate

## **Returning Port Objects**

- (NXPort *) <b>inPort</b>	Returns the connection's in port
- (NXPort *) <b>outPort</b>	Returns the connection's out port

## **Getting and Setting the Root Object**

- <b>rootObject</b>	Returns the connection's root object
- <b>setRoot:</b> <i>anObject</i>	Sets the connection's root object

## **Imported and Exported Objects**

- (List *) <b>remoteObjects</b>	Returns the connection's remote proxies
- (List *) <b>localObjects</b>	Returns the connection's local proxies

## **Returning a Proxy**

- <b>getLocal:</b> <i>anId</i>	Returns an object's local proxy
- <b>newRemote:</b> (unsigned) <i>anObject</i> <b>withProtocol:</b> (Protocol *) <i>p</i>	Creates a remote proxy for an object

## **Running a Connection**

- <b>run</b>	Runs the connection and blocks
- <b>runWithTimeout:</b> (int) <i>t</i>	Runs the connection for a while
- <b>runInNewThread</b>	Runs the connection asynchronously
- <b>runFromAppKit</b>	Runs the connection from DPS client
- <b>runFromAppKitWithPriority:</b> (int) <i>priority</i>	Runs the connection from DPS client

## **Freeing an NXConnection Instance**

- <b>free</b>	Frees the connection
---------------	----------------------

---

## NXProxy

**Inherits From:** none

**Conforms To:** NXReference  
NXTransport

### Counting References

- |                       |                              |
|-----------------------|------------------------------|
| – <b>addReference</b> | Adds a reference             |
| – <b>free</b>         | Eliminates a reference       |
| – <b>references</b>   | Returns number of references |

### Returning the proxy's connection

- |                             |                                |
|-----------------------------|--------------------------------|
| – <b>connectionForProxy</b> | Returns the proxy's connection |
|-----------------------------|--------------------------------|

### Freeing an NXProxy instance

- |                    |   |
|--------------------|---|
| – <b>freeProxy</b> | Frees the proxy but not its real object |
|--------------------|---|

### Determining if an object is a proxy

- |                  |                                    |
|------------------|------------------------------------|
| – <b>isProxy</b> | Identifies the receiver as a proxy |
|------------------|------------------------------------|

### Specifying a protocol

- |  |  |
|--|--|
| – <b>setProtocolForProxy:(Protocol *)proto</b> | Sets the proxy's protocol for efficiency |
|--|--|

---

## Object Additions

### Making Objects Distributable

- |   |                                      |
|---|--------------------------------------|
| – <b>encodeRemotelyFor:</b><br>(NXConnection *) <i>connection</i> | Transports an object using a proxy   |
| <b>freeAfterEncoding:(BOOL *)</b> <i>flagp</i>                    |                                      |
| <b>isBycopy:(BOOL)</b> <i>isBycopy</i>                            |                                      |
| – (BOOL) <b>isProxy</b>   | Identifies the receiver as an object |

# Protocols

---

## NXDecoding

**Adopted By:** A private class that decodes data sent across a connection

### Methods

– <b>decodeBytes:</b> (void *) <i>bytes</i> <i>count:(int)count</i>	Decodes untyped data
– <b>decodeData:</b> (void *) <i>d</i> <i>ofType:(const char *)t</i>	Decodes typed data
– <b>decodeMachPort:</b> (port_t *) <i>pp</i>	Decodes a Mach port
– <b>decodeObject</b>	Decodes an object
– <b>decodeVM:</b> (void **) <i>bytes</i> <i>count:(int *)count</i>	Decodes virtual memory pages

---

## NXEncoding

**Adopted By:** A private class that encodes data across a connection

### Methods

– <b>encodeBytes:</b> (const void *) <i>bytes</i> <i>count:(int)count</i>	Encodes untyped data
– <b>encodeData:</b> (void *) <i>data</i> <i>ofType:(const char *)type</i>	Encodes typed data
– <b>encodeMachPort:</b> (port_t) <i>port</i>	Encodes a Mach port
– <b>encodeObject:</b> <i>anObject</i>	Encodes an object as a proxy
– <b>encodeObjectBycopy:</b> <i>anObject</i>	Encodes a copy of an object
– <b>encodeVM:</b> (const void *) <i>bytes</i> <i>count:(int)count</i>	Encodes virtual memory pages

---

## NXTransport

**Adopted By:** List (common classes)  
NXData (Mach Kit)  
NXPort (Mach Kit)  
NXProxy

### Methods

- |   |  |
|---|--|
| <ul style="list-style-type: none"><li>– <b>encodeRemotelyFor:</b><br/>(NXConnection *)<i>connection</i><br/>freeAfterEncoding:(BOOL *)<i>flagp</i><br/>isBycopy:(BOOL)<i>isBycopy</i></li><li>– <b>encodeUsing:(id &lt;NXEncoding&gt;)portal</b></li><li>– <b>decodeUsing:(id &lt;NXDecoding&gt;)portal</b></li></ul> | <p>Determines what to encode across a connection</p> <p>Encodes an object across a connection</p> <p>Decodes an object over a connection</p> |
|---|--|

# 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) <b>understandsType:</b> (const char *) <i>aType</i> | Returns YES if streams or files of <i>aType</i> are parsed |
| – <b>addSourceType:</b> (const char *) <i>aType</i>          | Sets <i>aType</i> as a type that will be parsed            |
| – <b>removeSourceType:</b> (const char *) <i>aType</i>       | Unsets <i>aType</i> as a type that will be parsed          |

## Managing Parse Options

<b>- setMinimumWeight:(unsigned int)<i>aInt</i></b>	Sets the minimum weight for a token to be included
<b>- (unsigned int)minimumWeight</b>	Returns the minimum weight for a token to be included
<b>- setPercentPassed:(unsigned int)<i>aInt</i></b>	Sets the percent of tokens dropped
<b>- (unsigned int)percentPassed</b>	Returns the percent of tokens dropped
<b>- setWeightingDomain:(IXWeightingDomain *)<i>aDomain</i></b>	Sets the weighting domain for calculating peculiarities
<b>- (IXWeightingDomain *)weightingDomain</b>	Returns the weighting domain
<b>- setWeightingType:(IXWeightingType)<i>aInt</i></b>	Sets the type of weighting to calculate
<b>- (IXWeightingType)weightingType</b>	Returns the type of weighting calculated

## Parsing Text

<b>- parseFile:(const char *)<i>filename</i>     ofType:(const char *)<i>aType</i></b>	Parses the contents of <i>filename</i> if possible, adding the information to the attribute-value list
<b>- parseStream:(NXStream *)<i>stream</i>     ofType:(const char *)<i>aType</i></b>	Parses the contents of <i>stream</i> if possible, adding the information to the attribute-value list
<b>- (NXStream *)analyzeFile:(const char *)<i>filename</i>     ofType:(const char *)<i>aType</i></b>	Parses the contents of <i>filename</i> if possible, adding the information and returning the analyzed stream
<b>- (NXStream *)analyzeStream:(NXStream *)<i>stream</i>     ofType:(const char *)<i>aType</i></b>	Parses the contents of <i>stream</i> if possible, adding the information and returning the analyzed stream
<b>- reset</b>	Clears all compiled information in the attribute-value list

---

## IXAttributeQuery

**Inherits From:** Object

### Initializing an IXAttributeQuery

<b>- initQueryString:(const char *)<i>aString</i>     andAttributeParser:(IXAttributeParser *)<i>aParser</i></b>	Initializes a new IXAttributeQuery with <i>aString</i> as the query expression and <i>aParser</i> 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 *) <b>punctuation</b>	Returns the set of characters used to delimit tokens
– <b>setStopWords:(const char *)stopWords</b>	Sets the words that are deleted from a stream
– (char *) <b>stopWords</b>	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*  
– (BOOL)getKey:(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

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*  
– writeRange:(void \*)*aRange*  
    *atOffset*:(unsigned int)*anOffset*  
    *forLength*:(unsigned int)*aLength*  
– (unsigned int)readValue:(void \*\*)*aValue*  
  
– (unsigned int)readRange:(void \*\*)*aRange*  
    *ofLength*:(unsigned int)*aLength*  
    *atOffset*:(unsigned int)*anOffset*  
– removeValue

Writes or inserts *aValue* at the cursor's position

Writes *aRange* into the value at the cursor's position;  
    if the cursor isn't exactly on a key, raises an exception

Reads the value at the cursor's position and returns its  
    length; slides forward if needed to get a value

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

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  
IXNameAnd FileAccess  
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

– <b>setFilename:</b> (const char *) <i>aName</i>	Sets the filename that the IXFileRecord refers to to <i>aName</i>
– (const char *) <b>filename</b>	Returns the filename that the IXFileRecord refers to
– <b>setFileType:</b> (const char *) <i>aType</i>	Sets the recorded type for the associated file to <i>aType</i>
– (const char *) <b>filetype</b>	Returns the recorded type for the associated file
– <b>setDescription:</b> (const char *) <i>aDescription</i>	Sets the description for the associated file to <i>aName</i>
– (const char *) <b>description</b>	Returns the description for the associated file
– <b>setFiledate:</b> (unsigned int) <i>aDate</i>	Sets the recorded creation date for the associated file to <i>aDate</i>
– (unsigned int) <b>filedate</b>	Returns the recorded creation date for the associated file

## Accessing UNIX File Information

– (const struct stat *) <b>statBuffer</b>	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 *) <b>installedLanguages</b>	Returns the languages for which readers are installed
+ (Class) <b>classForLanguage:</b> (const char *) <i>aLanguage</i>	Returns the IXLanguageReader subclass for <i>aLanguage</i>

### Getting Objects Associated with Languages

+ <b>readerForLanguage:</b> (const char *) <i>aLanguage</i>	Returns an IXLanguageReader for <i>aLanguage</i>
+ <b>domainForLanguage:</b> (const char *) <i>aLanguage</i>	Returns an IXWeightingDomain for <i>aLanguage</i>

### Getting the Target Language

+ (NXAtom) <b>targetLanguage</b>	Returns the target language of the subclass
– (NXAtom) <b>targetLanguage</b>	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)handleForObjectAt:(unsigned int)index` Returns the handle for the posting at *index*
- `(unsigned int)weightForObjectAt:(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

- **initCount:(unsigned int)*count***  
    **andPostings:(const IXPosting \*)*postings***
- **initWithPostingsIn:*anObject***

Initializes a new IXPostingSet to contain *count postings*

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

– <b>abortTransaction</b>	Aborts the current transaction
– <b>commitTransaction</b>	Commits the changes made in the current transaction
– <b>(BOOL)areTransactionsEnabled</b>	Returns YES if a transaction has ever been started
– <b>(unsigned int)nestingLevel</b>	Returns the nesting level of transactions pending
– <b>(unsigned int)changeCount</b>	Returns the number of <b>commitTransaction</b> and <b>abortTransaction</b> messages received by the instantiation

## Accessing the Contents

– <b>setContents:(vm_address_t)<i>someContents</i> andLength:(vm_size_t)<i>aLength</i></b>	Replaces the IXStore’s storage with that in <i>someContents</i>
– <b>getContents:(vm_address_t *)<i>theContents</i> andLength:(vm_size_t *)<i>aLength</i></b>	Returns in <i>theContents</i> and <i>aLength</i> the IXStore’s storage and its length

## Reducing Memory Consumption

– <b>compact</b>	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

– <b>(void *)openAtOffset:(unsigned int)<i>anOffset</i> forLength:(unsigned int)<i>aLength</i></b>	Opens a portion of the block for writing if possible; raises an exception otherwise
– <b>(void *)readAtOffset:(unsigned int)<i>anOffset</i> forLength:(unsigned int)<i>aLength</i></b>	Opens a portion of the block for reading if possible; raises an exception otherwise
– <b>(unsigned int)copyAtOffset:(unsigned int)<i>anOffset</i> forLength:(unsigned int)<i>aLength</i></b>	Copies a portion of the block, returning the copy’s identifier
– <b>close</b>	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

<code>- (BOOL)hasEntryNamed:(const char *)<i>aName</i></code>	Returns whether there's an object stored under <i>aName</i>
<code>- getBlock:(unsigned int *)<i>aHandle</i> ofEntryNamed:(const char *)<i>aName</i></code>	Gets the block identifier for the object stored under <i>aName</i>
<code>- getClass:(id *)<i>aClass</i> ofEntryNamed:(const char *)<i>aName</i></code>	Gets the class object for the object stored under <i>aName</i>
<code>- openEntryNamed:(const char *)<i>aName</i></code>	Activates and returns the object stored under <i>aName</i>
<code>- (const char **)<i>entries</i></code>	Returns the stored names

---

## IXStoreFile

**Inherits From:** IXStore : Object

### Initializing and Freeing Instances

<code>- init</code>	Initializes a new IXStoreFile in a temporary file
<code>- initWithFile:(const char *)<i>filename</i></code>	Initializes a new IXStoreFile with a file named <i>filename</i>
<code>- initFromFile:(const char *)<i>filename</i> forWriting:(BOOL)<i>flag</i></code>	Initializes an IXStoreFile from existing storage in <i>filename</i> , allowing writing back to the file if <i>flag</i> is YES
<code>- free</code>	Frees the IXStoreFile, but doesn't affect the storage file

### Limiting the File Mapping Size

<code>- setSizeLimit:(vm_size_t)<i>aLimit</i></code>	Sets the caching limit for the IXStoreFile to <i>aLimit</i>
<code>- (vm_size_t)sizeLimit</code>	Returns the caching limit for the IXStoreFile

### Getting File Information

<code>- (int)descriptor</code>	Returns the file descriptor for the storage file
<code>- (const char *)filename</code>	Returns the name of the storage file

# IXWeightingDomain

Inherits From: Object

## Initializing Instances

- **initFromDomain:**(NXStream \*)*stream*
- **initFromHistogram:**(NXStream \*)*stream*
- **initFromWFTable:**(NXStream \*)*stream*

Initializes a new IXWeightingDomain from domain format data in *stream*  
Initializes a new IXWeightingDomain from histogram format data in *stream*  
Initializes a new IXWeightingDomain from NeXTSTEP 2.0 WFTable format data in *stream*

## Saving Domain Information

- **writeDomain:**(NXStream \*)*stream*
- **writeHistogram:**(NXStream \*)*stream*
- **writeWFTable:**(NXStream \*)*stream*

Write domain format data to *stream*  
Writes histogram format data to *stream*  
Writes NeXTSTEP 2.0 WFTable format data to *stream*

## Counting Tokens

- (unsigned int)**totalTokens**
- (unsigned int)**uniqueTokens**

Returns the total number of tokens in the weighting domain  
Returns the total number of unique tokens

## Retrieving Information about Tokens

- (unsigned int)**countForToken:**(void \*)*aToken*  
    *ofLength:*(unsigned int)*aLength*
- (unsigned int)**rankForToken:**(void \*)*aToken*  
    *ofLength:*(unsigned int)*aLength*
- (float)**frequencyOfToken:**(void \*)*aToken*  
    *ofLength:*(unsigned int)*aLength*
- (float)**peculiarityOfToken:**(void \*)*aToken*  
    *ofLength:*(unsigned int)*aLength*  
    *andFrequency:*(float)*aFrequency*

Returns the number of occurrences of *aToken* in the domain  
Returns the ordinal rank of occurrences of *aToken* in the domain  
Returns the frequency of *aToken* in the domain  
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**
- **(const char \*)comparisonFormat**

Sets the data format of items compared by the receiver  
Returns the data format of items compared

---

## **IXCursorPositioning**

**Adopted By:** IXBTreeCursor

### **Absolute Positioning**

- **(BOOL)setKey:(void \*)aKey  
andLength:(unsigned int)aLength**
- **(BOOL)getKey:(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

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**
- **(BOOL)setNext**
- **(BOOL)setLast**
- **(BOOL)setPrevious**

Positions the receiver at the first key if there is one; returns  
YES if there is one, NO if not

Moves the receiver forward one key value and returns YES  
if there's a key there

Positions the receiver at the last key if there is one; returns  
YES if there is one, NO if not

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

– <b>setScansForModifiedFiles:(BOOL)<i>flag</i></b>	Sets whether the file finder scans for files whose modification times have changed
– <b>(BOOL)scansForModifiedFiles</b>	Returns whether the file finder scans for modified files

### Ignoring Files

– <b>setIgnoredTypes:(const char *)<i>types</i></b>	Sets to <i>types</i> the types of files that won't be indexed
– <b>(char *)ignoredTypes</b>	Returns the types of files that aren't indexed
– <b>setIgnoredNames:(const char *)<i>names</i></b>	Sets to <i>names</i> the literal, base names of files that won't be indexed
– <b>(char *)ignoredNames</b>	Returns the names of files that aren't indexed

## IXFileFinderQueryAndUpdate

**Adopted By:** IXFileFinder

### Getting the Target Directory

– <b>(const char *)rootPath</b>	Returns the base path for the file finder's index
---------------------------------	---

### Getting the Record Manager

– <b>recordManager</b>	Returns the object that stores the file finder's IXFileRecords
------------------------	--

### Performing Queries

– <b>(IXPostingList *)performQuery:(const char *)<i>aQuery</i></b>	Evaluates <i>aQuery</i> for <i>sender</i> returning in an IXPostingList the IXFileRecords that match
<b>atPath:(const char *)<i>path</i></b>	
<b>forSender:<i>sender</i></b>	

## Updating Indexes

– <b>updateIndexAtPath:(const char *)path forSender:sender</b>	Updates the indexes for files within <i>path</i> relative to the file finder's root path
– <b>(BOOL)isUpdating</b>	Returns whether the file finder is updating its indexes
– <b>suspendUpdating</b>	Suspends updating of indexes
– <b>resumeUpdating</b>	Resumes updating of indexes
– <b>clean</b>	Removes inaccurate or out of date information from indexes
– <b>reset</b>	Completely empties indexes

## Methods Implemented by the Sender of a Query or Update

– <b>fileFinder:(IXFileFinder *)aFinder didFindFile:(IXFileRecord *)aRecord</b>	Asynchronously notifies the sender of a <b>performQuery:atPath:forSender:</b> message that <i>aRecord</i> matches the query
– <b>fileFinder:(IXFileFinder *)aFinder didFindList:(IXPostingList *)aList</b>	Asynchronously notifies the sender of an <b>performQuery:atPath:forSender:</b> message that the IXFileRecords in <i>aList</i> match the query
– <b>fileFinder:(IXFileFinder *)aFinder willAddFile:(IXFileRecord *)aRecord</b>	Asynchronously notifies the sender of an <b>updateIndexAtPath:forSender:</b> message that <i>aRecord</i> is about to be added to the index

---

## IXLexemeExtraction

**Adopted By:** No NeXTSTEP classes.

### Lexing a Stream

– <b>(unsigned int)getLexeme:(char *)aString inLength:(unsigned int)aLength fromStream:(NXStream *)stream</b>	Puts the next lexeme from <i>stream</i> into <i>aString</i>
---	---

### Manipulating a Word/Lexeme

– <b>(unsigned int)foldCase:(char *)aString inLength:(unsigned int)aLength</b>	Reduces <i>aString</i> 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*
  - **initFromName:**(const char \*)*aName*  
    **inFile:**(const char \*)*filename*  
        **forWriting:**(BOOL)*flag*
  - **freeFromStore**
  - + **freeFromName:**(const char \*)*aName*  
    **andFile:**(const char \*)*filename*
- Initializes a new store client under *aName* in *filename*
- Initializes a store client from data previously stored under *aName* in *filename*; if *flag* is YES, changes can be written back to the file
- Removes the store client's storage from the IXStoreFile and frees the run-time object
- 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*  
    **withWeight:**(unsigned int)*aWeight*

– **removeHandle:**(unsigned int)*aHandle*

Adds a postings to the set of postings

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*

– (unsigned int)**getHandle:**(unsigned int \*)*aHandle*  
    **andWeight:**(unsigned int \*)*aWeight*

– (unsigned int)**setFirstHandle**

– (unsigned int)**setNextHandle**

Sets the selected posting to the one with *aHandle* and returns that handle, or 0 if *aHandle* isn't in the set

Gets the handle and weight of the selected posting

Sets the selected posting to the first in the set and returns its handle, or 0 if there are no postings

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*
- **reclaimRecord:**(unsigned int)*aHandle*

Discards the records identified by *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**
- **readRecord:**(unsigned int)*aHandle*  
    **fromZone:**(NXZone \*)*zone*

Returns the number of records in the archive

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  
offIvar:(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  
offIvar:(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  
offIvar:(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  
offIvar:(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  
offIvar:(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  
offIvar:(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  
offIvar:(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)doubleValue:(double *) <i>aValue</i><br>ofMessage:(SEL) <i>aSelector</i><br>forRecord:(unsigned int) <i>aHandle</i>  | Retrieves as a <b>double</b> the value obtained by sending<br><i>aSelector</i> to the object whose record is identified<br>by <i>aHandle</i>                        |
| – (BOOL)floatValue:(float *) <i>aValue</i><br>ofMessage:(SEL) <i>aSelector</i><br>forRecord:(unsigned int) <i>aHandle</i>  | Retrieves as a <b>float</b> the value obtained by sending<br><i>aSelector</i> to the object whose record is identified<br>by <i>aHandle</i>                         |
| – (BOOL)intValue:(int *) <i>aValue</i><br>ofMessage:(SEL) <i>aSelector</i><br>forRecord:(unsigned int) <i>aHandle</i>  | Retrieves as a <b>int</b> the value obtained by sending<br><i>aSelector</i> to the object whose record is identified<br>by <i>aHandle</i>                           |
| – (BOOL)objectValue:(Object **) <i>aValue</i><br>ofMessage:(SEL) <i>aSelector</i><br>forRecord:(unsigned int) <i>aHandle</i>   | Retrieves as an object the value obtained by sending<br><i>aSelector</i> to the object whose record is identified<br>by <i>aHandle</i>                              |
| – (BOOL)opaqueValue:(NXData **) <i>aValue</i><br>ofMessage:(SEL) <i>aSelector</i><br>forRecord:(unsigned int) <i>aHandle</i>   | Retrieves as untyped data the value obtained by sending<br><i>aSelector</i> to the object whose record is identified<br>by <i>aHandle</i>                           |
| – (BOOL)stringValue:(char **) <i>aValue</i><br>ofMessage:(SEL) <i>aSelector</i><br>forRecord:(unsigned int) <i>aHandle</i>   | Retrieves as a string the value obtained by sending<br><i>aSelector</i> to the object whose record is identified<br>by <i>aHandle</i>                               |
| – (BOOL)stringValue:(char **) <i>aValue</i><br>inLength:(unsigned int) <i>aLength</i><br>ofMessage:(SEL) <i>aSelector</i><br>forRecord:(unsigned int) <i>aHandle</i> | Retrieves as a string no longer than <i>aLength</i> the value<br>obtained by sending <i>aSelector</i> to the object whose<br>record is identified by <i>aHandle</i> |

# Functions

## Comparator Functions

Compare two sets of data as arrays of various types and return their ordering.

int	<b>IXCompareBytes</b> (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	<b>IXCompareUnsignedBytes</b> (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	<b>IXCompareShorts</b> (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	<b>IXCompareUnsignedShorts</b> (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	<b>IXCompareLongs</b> (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	<b>IXCompareUnsignedLongs</b> (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	<b>IXCompareFloats</b> (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	<b>IXCompareDoubles</b> (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	<b>IXCompareShort</b> (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	<b>IXCompareUnsignedShort</b> (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	<b>IXCompareLong</b> (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	<b>IXCompareUnsignedLong</b> (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	<b>IXCompareFloat</b> (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	<b>IXCompareDouble</b> (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	<b>IXCompareStringAndUnsigneds</b> (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	<b>IXCompareUnsignedAndStrings</b> (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	<b>IXCompareStrings</b> (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	<b>IXCompareMonocaseStrings</b> (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

<code>self</code>	Refers to the object the expression is being evaluated for
<code>true</code> or <code>yes</code>	Boolean values for TRUE
<code>false</code> or <code>no</code>	Boolean values for FALSE
<code>"aString"</code> or <code>'aString'</code>	Declares <i>aString</i> a string literal
<code>number</code>	Any legal integer or floating point number
<code>AttributeName</code>	An attribute reference

## Transform Operators

<code>quote(some text)</code>	Results in a string value equal to <i>some text</i>
<code>regex(a BSD regular expression)</code>	Results in a regular expression
<code>shell(an expression with shell wildcards)</code>	Results in a regular expression
<code>parse(some text)</code>	Parses the text argument into an attribute-value list

## Projection Operators

<code>project(AttributeName entity_or_object)</code>	Projects the attribute named <i>AttributeName</i> from the set of attributes represented by <i>entity_or_object</i>
--	---

## Boolean Operators

<code>or(a b)</code>	Results in true if either <i>a</i> or <i>b</i> is true, false if both are false
<code>and(a b)</code>	Results in true if both <i>a</i> and <i>b</i> are true, false if either is false
<code>not(a b)</code>	Equivalent to <code>and(a not(b))</code>
<code>not(a)</code>	The logical negation of <i>a</i>

## Set Operators

<code>or(s t)</code>	Results in the union of sets <i>s</i> and <i>t</i>
<code>or(s)</code>	“Any one or more of the items in set <i>s</i> ” (for use with search operators only)
<code>and(s t)</code>	Results in the intersection of sets <i>s</i> and <i>t</i>
<code>and(s)</code>	“All of the items in set <i>s</i> ” (for use with search operators only)
<code>not(s t)</code>	Results in a set containing those items in set <i>s</i> that aren’t in set <i>t</i>

## Relational Operators

<code>gt(a b)</code>	Results in <b>true</b> if $a$ is greater than $b$ , <b>false</b> otherwise
<code>ge(a b)</code>	Results in <b>true</b> if $a$ is greater than or equal to $b$ , <b>false</b> otherwise
<code>eq(a b)</code>	Results in <b>true</b> if $a$ is equal to $b$ , <b>false</b> otherwise
<code>ne(a b)</code>	Results in <b>true</b> if $a$ is not equal to $b$ , <b>false</b> otherwise
<code>lt(a b)</code>	Results in <b>true</b> if $a$ is less than $b$ , <b>false</b> otherwise
<code>le(a b)</code>	Results in <b>true</b> if $a$ is less than or equal to $b$ , <b>false</b> otherwise

## Arithmetic Operators

<code>add(a b)</code>	Results in the sum of $a$ and $b$
<code>sub(a b)</code>	Results in difference of $a$ and $b$
<code>mul(a b)</code>	Results in product of $a$ and $b$
<code>div(a b)</code>	Results in quotient of $a$ and $b$
<code>neg(a)</code>	Results in the arithmetic negative of $a$

## Search Operators

<code>whole(value)</code>	Searches for $value$ as an exact match in the Default attribute
<code>whole(Attribute Name value)</code>	Searches for $value$ as an exact match in the attribute named <i>AttributeName</i>
<code>whole(entity value)</code>	Searches for $value$ as an exact match in <i>entity</i> ; results in <b>true</b> if value is found, <b>false</b> if not
<code>whole(string value)</code>	Searches for $value$ as an exact match to <i>string</i> ; results in <b>true</b> if value is found, <b>false</b> if not (this is equivalent to <code>eq(string value)</code> )
<code>prefix()</code>	Searches for a value as a prefix match; arguments may be as for <code>whole()</code>
<code>within()</code>	Searches for a value as a match anywhere within an attribute or string; arguments may be as for <code>whole()</code>

## Pre-defined Attributes

<code>Default</code>	The default attribute defined by the query language
<code>Content</code>	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 <code>stat()</code> (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 <code>stat()</code> (number).
ModifyTime	The time the file's content was last modified, as returned by <code>stat()</code> (number).
ChangeTime	The time the file's status information was last changed, as returned by <code>stat()</code> (number).
UnixType	The inode type for the file, encoded as a number.

# Attribute Reader Format

## Declaring Attributes

{\zd *AttributeName*}  
{\zd *AttributeName*\zt*typeEncoding*}

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*<sup>th</sup> lexeme encountered

## Adding Information to Lexeme Markings

{\z *lexeme*\zan}  
{\z *lexeme*\zw*weight*}  
{\z *lexeme*\zccookie}

Associate *lexeme* with the *n*<sup>th</sup> attribute declared  
Marks *lexeme* as though it occurred *weight* times in the text  
Declare *cookie* as an opaque identifier associated with  
the *lexeme*



---

# 8      *Interface Builder*

## Classes

---

### **IBInspector**

**Inherits From:** Object

**Conforms To:** IBInspectors

#### Accessing Objects

- |                 |   |
|-----------------|---|
| – <b>object</b> | Returns the object that's being inspected                             |
| – <b>window</b> | Returns the window that contains the user interface for the inspector |

#### Managing Changes

- |                               |  |
|-------------------------------|--|
| – <b>touch:sender</b>         | Changes the image in the Inspector panel's close box to a broken "X" |
| – <b>textDidChange:sender</b> | 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 *) <b>getConnectionInspectorClassName</b>	Implement to return class name of Connection inspector
– (const char *) <b>getEditorInspectorClassName</b>	Implement to return class name of the object's editor
– (const char *) <b>getHelpInspectorClassName</b>	Implement to return class name of the Help inspector
– (NXImage *) <b>getIBImage</b>	Implement to return an image to represent the object in the File window.
– (const char *) <b>getInspectorClassName</b>	Implement to return class name of the Attributes inspector
– (const char *) <b>getSizeInspectorClassName</b>	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

– <b>getMinSize:(NXSize *)minSize maxSize:(NXSize *)maxSize from:(int)where</b>	Implement this method to control the dimensions of a View
– <b>placeView:(NXRect *)frameRect</b>	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

– <b>destination</b>	Implement to return the object that's the destination of the connection
– <b>establishConnection</b>	Implement to connect the source and destination objects
– <b>free</b>	Implement to release the storage for the connector object
– <b>nibInstantiate</b>	Implement to verify the identities of the connector's source and destination objects
– <b>read:(NXTypedStream *)stream</b>	Implement to unarchive the connector object from <i>stream</i>
– <b>renewObject:<i>old</i> to:<i>new</i></b>	Implement to update a connector by replacing its old source or destination object with a new object
– <b>source</b>	Implement to return the object that's the source of the connection
– <b>write:(NXTypedStream *)stream</b>	Implement to archive the connector object to <i>stream</i>

---

## IBDocuments

**Adopted By:** Interface Builder's document objects

### Managing the Document

– <b>touch</b>	Marks the document as edited
– <b>getDocumentPathIn:(char *)buffer</b>	Places the document's path in <i>buffer</i>

### Managing the Object Hierarchy

– <b>attachObject:<i>anObject</i> to:<i>parent</i></b>	Adds <i>anObject</i> to the hierarchy by attaching it to <i>parent</i>
– <b>attachObjects:(List *)objectList to:<i>parent</i></b>	Adds the objects in <i>objectList</i> to the hierarchy by attaching them to <i>parent</i>
– <b>deleteObject:<i>anObject</i></b>	Removes <i>anObject</i> from the object hierarchy

<code>- deleteObjects:(List *)objectList</code>	Removes objects in <i>objectList</i> from the object hierarchy
<code>- copyObject:<i>anObject</i>     <i>type</i>:(NXAtom)<i>type</i>     <i>inPasteboard</i>:(Pasteboard *)<i>aPasteboard</i></code>	Copies <i>anObject</i> to the specified pasteboard
<code>- copyObjects:(List *)<i>objectList</i>     <i>type</i>:(NXAtom)<i>type</i>     <i>inPasteboard</i>:(Pasteboard *)<i>aPasteboard</i></code>	Copies the objects in <i>objectList</i> to the specified pasteboard
<code>- (List *)<i>pasteType</i>:(NXAtom)<i>type</i>     <i>fromPasteboard</i>:(Pasteboard *)<i>pboard</i>     <i>parent</i>:<i>theParent</i></code>	Alerts the document object that objects were pasted
<code>- (BOOL)<i>objectIsMember</i>:<i>anObject</i></code>	Returns YES if <i>anObject</i> is a part of the object hierarchy
<code>- getObjects:(List *)<i>objectList</i></code>	Places the objects from the object hierarchy into <i>objectList</i>
<code>- getParentForObject:<i>anObject</i></code>	Returns the object above <i>anObject</i> in the object hierarchy

## Setting Object Names

<code>- (BOOL)setName:(const char *)<i>name</i>     <i>for</i>:<i>anObject</i></code>	Sets the name associated with the <i>anObject</i>
<code>- getNameIn:(char *)<i>name</i>     <i>for</i>:<i>anObject</i></code>	Places the name associated with <i>anObject</i> in the buffer <i>name</i>

## Managing Connectors

<code>- addConnector:<i>aConnector</i></code>	Adds a connector object to Interface Builder's list
<code>- removeConnector:<i>aConnector</i></code>	Removes <i>aConnector</i> from the list of connectors.
<code>- listConnectors:(List *)<i>aList</i>     <i>forSource</i>:<i>aSource</i></code>	Places in <i>aList</i> connector objects whose sources are <i>aSource</i>
<code>- listConnectors:(List *)<i>aList</i>     <i>forDestination</i>:<i>aDestination</i></code>	Places in <i>aList</i> connector objects whose destinations are <i>aDestination</i>
<code>- listConnectors:(List *)<i>aList</i>     <i>forSource</i>:<i>aSource</i>     <i>filterClass</i>:<i>filterClass</i></code>	Places in <i>aList</i> the connector objects of class <i>filterClass</i> whose sources are <i>aSource</i>
<code>- listConnectors:(List *)<i>aList</i>     <i>forDestination</i>:<i>aDestination</i>     <i>filterClass</i>:<i>filterClass</i></code>	Places in <i>aList</i> the connector objects of class <i>filterClass</i> whose destinations are <i>aDestination</i>

## Managing Editors

<code>- setSelectionFrom:<i>anEditor</i></code>	Registers <i>anEditor</i> as the editor that owns the selection
<code>- editorDidClose:<i>anEditor</i>     <i>for</i>:<i>anObject</i></code>	Informs the document object that <i>anEditor</i> is no longer 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 \*)getConnectInspectorClassName** 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 IBMENUBOARDTYPE;  
NXAtom IBMENUCELLPboardType;  
NXAtom IBVIEWPboardType;  
NXAtom IBWINDOWPboardType;
```

## Classes

### **NXConditionLock**

**Inherits From:** Object

**Conforms To:** NXLock

#### **Initializing an Instance**

- |                                  |                              |
|----------------------------------|------------------------------|
| – <b>init</b>                    | Initialize a condition lock  |
| – <b>initWith:(int)condition</b> | Initialize and set condition |

#### **Get the Condition of the Lock**

- |                    |                      |
|--------------------|----------------------|
| – <b>condition</b> | Return the condition |
|--------------------|----------------------|

#### **Acquire or Release the Lock**

- |                                    |                               |
|------------------------------------|-------------------------------|
| – <b>lock</b>                      | Grab the lock                 |
| – <b>lockWhen:(int)condition</b>   | Wait for the condition        |
| – <b>unlock</b>                    | Release the lock              |
| – <b>unlockWith:(int)condition</b> | 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 *) <i>nxport</i><br>withName:(const char *) <i>aName</i> | Advertises a port |
|--|-------------------|

### **Removing a Port**

- |   |                                      |
|---|--------------------------------------|
| + checkOutPortWithName:<br>(const char *) <i>name</i> | Withdraws a port from public display |
|---|--------------------------------------|

### **Getting Ports**

- |   |                                |
|---|--------------------------------|
| + (NXPort *) lookUpPortWithName:<br>(const char *) <i>name</i>  | Finds a port on the local host |
| + (NXPort *) lookUpPortWithName:<br>(const char *) <i>name</i><br>onHost:(const char *) <i>hostname</i> | Finds a port on a named host   |

---

## NXPort

**Inherits From:** NXInvalidationNotifier : Object

**Conforms To:** NXReference (NXInvalidationNotifier)  
NXTransport

### Transporting Ports

- **encodeRemotelyFor:**  
(NXConnection \*)*connection*  
freeAfterEncoding:(BOOL \*)*flagp*  
isBycopy:(BOOL)*isBycopy*
  - **encodeUsing:(id <NXEncoding>)portal**
  - **decodeUsing:(id <NXDecoding>)portal**
- Has the port copied across a connection
- Encodes the port across a connection
- Encodes the port across a connection

### Creating an NXPort

- + **new**
  - + **newFromMachPort:(port\_t) *p***
  - + **newFromMachPort:(port\_t) *p*  
dealloc:(BOOL) *flag***
- Creates a new NXPort object
- Wraps an existing port
- Wraps an existing port

### 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**
  - **hash**
- Returns the embedded mach port
- Returns a hashtable value for the object

---

## NXProtocolChecker

**Inherits From:** Object

### Initializing a Protocol Checker

- **initWithObject:*anObject***  
**forProtocol:(Protocol \*)*proto***

Initializes a new object

### Reimplemented Object Methods

- **forward:(SEL)*sel*:(void \*)*args***
- **(struct objc\_method\_description \*)  
descriptionForMethod:(SEL)*sel***
- **free**

Passes allowed messages to its delegate

Required for the run-time system

Frees the checker and/or its delegate

---

## NXRecursiveLock

**Inherits From:** Object

**Conforms To:** NXLock

### Acquire or Release a Lock

- **lock**
- **unlock**

Grab the lock

Release the lock

---

## **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\_PHASEDEPTH  
(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

<b>Event Count</b>	<b>Value</b>
MIDI_MAX_EVENT	100

<b>Event Size</b>	<b>Value</b>
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\_NOTEON  
MIDI\_NOTEOFF  
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<br>readTimeout:(int)rtime<br>writeTimeout:(int)wtime<br>canAbort:(BOOL)abort<br>mustWrite:(BOOL)write | Makes a connection to a domain, as specified |

- (ni\_status)**setTaggedConnection:**(const char \*)*tag*  
    *to:(char \*)hostName*
  - (ni\_status)**setTaggedConnection:**(const char \*)*tag*  
    *to:(char \*)hostName*  
    *readTimeout:(int)rtime*  
    *writeTimeout:(int)wtime*  
    *canAbort:(BOOL)abort*
  - **disconnectFromCurrent**
- Makes a connection to a domain, using host name and tag
- Makes a connection to a domain, as specified
- Terminates a connection to a domain

### Getting Data about or from the Current Domain

- (const char \*)**getFullPath**
  - (const char \*)**getMasterServer**
  - (const char \*)**getCurrentServer**
  - (const char \*)**getTag**
  - (const struct sockaddr\_in \*)**getServerIPAddress**
  - (void \*)**getDomainHandle**
  - (ni\_entrylist \*)**findDirectory:**(const char \*)*parentDirectory*  
    *withProperty:(const char \*)property*
- Gets the path name of the current domain
- Gets the host name of the current domain's master server
- Gets the host name of the current domain's current server
- Gets the tag of the current domain
- Gets the socket address of the current domain's current server
- Gets the NetInfo™ handle of the current domain
- 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**
  - + **allocWithoutPanelFromZone:**(NXZone \*)*zone*
  - **init**
- Returns an NIDomainPanel object
- Allocates an NIDomainPanel object that has no panel
- 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: <i>sender</i> <i>inDomain</i> :(void *) <i>domainID</i>	Brings up the panel
- (BOOL)runModal: <i>sender</i> <i>inDomain</i> :(void *) <i>domainID</i> <i>withUser</i> :(const char *) <i>userName</i> <i>withInstruction</i> :(const char *) <i>whatWarning</i> <i>allowChange</i> :(BOOL) <i>disableUser</i>	Brings up the panel, as specified
- (BOOL)runModalWithValidation: <i>sender</i> <i>inDomain</i> :(void *) <i>domainID</i> <i>withUser</i> :(const char *) <i>userName</i> <i>withInstruction</i> :(const char *) <i>whatWarning</i> <i>allowChange</i> :(BOOL) <i>enableUser</i>	Brings up the panel and specifies validation by the delegate

### Target and Action Methods

- ok: <i>sender</i>	Method invoked by the OK button
- cancel: <i>sender</i>	Method invoked by the Cancel button

### Getting Data

- (BOOL)isValidLogin: <i>sender</i>	Returns whether the login was successful
- (const char *)getPassword: <i>sender</i>	Returns the text in the password field
- (const char *)getUser: <i>sender</i>	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
- setTitle: (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 fillMatrix:matrix inColumn:(int)column Fills the indicated column with data
- browser:sender loadCell:cell atRow:(int)row inColumn:(int)column Fills the indicated cell with data

## Text-Related Methods

- text:textObj isEmpty:(BOOL)flag Detects empty text field
- (BOOL)textWillChange: textObject Detects that text is about to change

<b>- completeItemName</b>	Reserved for future use
<b>- completeDomain</b>	Completes the text in the upper text field

### Target and Action Methods

<b>- cellWasHitInBrowser:(id)sender</b>	Method invoked by the upper browser
<b>- cellWasHitInItemList:sender</b>	Method invoked by the lower browser

## NISavePanel

**Inherits From:** NIOpenPanel : NIDomainPanel : Object

### Creating a New NISavePanel Object

<b>+ new</b>	Returns an NISavePanel object
--------------	-------------------------------

### Displaying the Panel

<b>- (int)runModal</b>	Brings up the panel
<b>- (int)runModalWithString:(char *)initialValue</b>	Brings up the panel, with the string in the lower text field
<b>- (int)runModalWithUneditableString:(char *)initialValue</b>	Brings up the panel, with an uneditable string in the lower text field

### Getting Data from the Panel

<b>- (const char *)panelSizeDefaultName</b>	Returns a string indicating the panel's default size
<b>- (const char *)directory</b>	Returns the directory that's selected in the lower browser

### Target and Action Methods

<b>- cellWasHitInItemList:sender</b>	Method invoked by the lower browser
--------------------------------------	-------------------------------------

### Manipulating the Panel

<b>- setStartingDomainPath:(const char *)directory</b>	Sets the initial path in the upper browser
<b>- refreshLowerData:sender</b>	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 NIHierarchyOfDomains *domains, int level, const  
                                     char *toMatch, int selectedLevel);
```

# Types and Constants

## Symbolic Constants

### Connection Status

NI\_ALREADYCONNECTED  
NI\_NOTCONNECTED

Test Mode	Value
NI_USERTESTMODE	0
NI_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**
- **initType:(NXPhoneDeviceType)*deviceType***

Initializes the NXPhone object to be type NX\_ISDNDevice  
Initializes the NXPhone object to *deviceType*

#### Running the Connection to the Phone Server

- **run**
- **runFromAppKit**
- **addPort:(port\_t)*aPort* receiver:*anObject* method:(SEL)*aSelector***

Runs the connection to the Phone Server  
Makes the main event loop monitor the Phone Server  
Adds *aPort* as a source for remote input

#### Testing the Phone Line

- **(BOOL)isActive**

Returns whether the phone line is working

## Managing Channels

– (void) <b>addChannel:aChannel</b>	Registers <i>aChannel</i> as one that the application can use
– (void) <b>removeChannel:aChannel</b>	Disassociates <i>aChannel</i> from the NXPhone object
– (BOOL) <b>acquireChannel:aChannel</b>	Locks down <i>aChannel</i> for the application's exclusive use
– (void) <b>releaseChannel:aChannel</b>	Relinquishes exclusive rights over <i>aChannel</i>

---

## NXPhoneCall

**Inherits From:** Object

### Initializing an NXPhoneCall Object

– <b>init</b>	Initializes the NXPhoneCall to be type NX_VoiceCall
– <b>initType:(NXPhoneCallType)callType</b>	Initializes the NXPhoneCall object to <i>callType</i>
– (void) <b>setType:(NXPhoneCallType)callType</b>	Sets the NXPhoneCall object to be <i>callType</i>
– (NXPhoneCallType) <b>type</b>	Returns the call type, NX_VoiceCall or NX_DataCall

### Initiating a Call

– (void) <b>pickUp</b>	Takes the phone off-hook to initiate an outgoing call
– (void) <b>dialToneReceived</b>	Implemented by subclasses to respond to a dial tone
– (void) <b>dialDigits:(char *)digits</b>	Dials the phone number recorded in <i>digits</i>
– (void) <b>dialingComplete</b>	Implemented by subclasses to react when number is dialed
– (void) <b>remoteBusy</b>	Implemented by subclasses to respond to busy signal
– (void) <b>remoteRing</b>	Implemented by subclasses to respond to remote ring
– (void) <b>remotePickup</b>	Implemented by subclasses to respond to remote pickup

### Getting a Call

– (void) <b>ring</b>	Implemented by subclasses to respond to ringing phone
– (void) <b>pickUp</b>	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

- |  |   |
|--|---|
| – <code>init</code>  | Initializes the NXPhoneChannel to NX_AnyISDNChannel         |
| – <code>initType:(NXPhoneChannelType)channelType</code>      | Initializes the NXPhoneChannel object to <i>channelType</i> |
| – <code>(void)setType:(NXPhoneChannelType)channelType</code> | Sets the type of the NXPhoneChannel to <i>channelType</i>   |
| – <code>(NXPhoneChannelType)type</code>                      | Returns the channel type                                    |

### Tracking Calls

- |  |  |
|--|--|
| – <code>(void)addCall:<i>aCall</i></code>    | Adds <i>aCall</i> to the channel                         |
| – <code>(void)removeCall:<i>aCall</i></code> | Removes <i>aCall</i> from the channel                    |
| – <code>activeCall</code>                    | Returns the currently active NXPhoneCall for the channel |

### Setting Up an Incoming Call

- |  |  |
|--|--|
| – <code>allocateIncomingCallOfType:(NXPhoneCallType)<i>callType</i></code> | Implemented by subclasses to provide object to handle call |
| – <code>(BOOL)acceptCall:<i>newCall</i></code>                             | Implemented by subclasses to accept or refuse a call       |

### Responding to Errors

- |  |  |
|--|--|
| – <code>(void)channelError:(NXPhoneError)<i>cause</i></code> | Implemented by subclasses to handle errors due to <i>cause</i> |
|--|--|

# 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 owner:*anOwner*** Loads the nib file named “name.nib” and makes *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

- |               |  |
|---------------|--|
| – <b>view</b> | Returns the View that's loaded into the Preferences window |
|---------------|--|

### Notification of State Change

- |                              |   |
|------------------------------|---|
| – <b>didHide:sender</b>      | Received when the application hides itself              |
| – <b>didUnhide:sender</b>    | Received when the application unhides itself            |
| – <b>willSelect:sender</b>   | Received before the module is displayed                 |
| – <b>didSelect:sender</b>    | Received after the module is displayed                  |
| – <b>willUnselect:sender</b> | Received before the module is removed from display      |
| – <b>didUnselect:sender</b>  | 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



---

# 15 *The Run-Time System*

## Classes

---

### Protocol

**Inherits From:** Object

#### Getting the Protocol Name

– (const char \*)**name** Returns the name of the protocol

#### Testing for Incorporated Protocols

– (BOOL)**conformsTo:(Protocol \*)aProtocol** Returns whether the receiver incorporates *aProtocol*

#### Getting Method Descriptions

– (struct objc\_method\_description \*)**descriptionForInstanceMethod:(SEL)aSelector**  
Returns information about the *aSelector* instance method

– (struct objc\_method\_description \*)**descriptionForClassMethod:(SEL)aSelector**  
Returns information about the *aSelector* class method

# Functions

## Class Functions

### Create a new instance of a class:

id	<code>class_createInstance(Class aClass, unsigned int indexedIvarBytes)</code>
id	<code>class_createInstanceFromZone(Class aClass, unsigned int indexedIvarBytes, NXZone *zone)</code>

### Get the class template for an instance variable:

Ivar	<code>class_getInstanceVariable(Class aClass, const char *variableName)</code>
------	--

### Get, add, and remove methods:

Method	<code>class_getInstanceMethod(Class aClass, SEL aSelector)</code>
Method	<code>class_getClassMethod(Class aClass, SEL aSelector)</code>
void	<code>class_addMethods(Class aClass, struct objc_method_list *methodList)</code>
void	<code>class_removeMethods(Class aClass, struct objc_method_list *methodList)</code>

### Pose as the superclass:

Class	<code>class_poseAs(Class theImposter, Class theSuperclass)</code>
-------	---

### Set and get the class version:

void	<code>class_setVersion(Class aClass, int versionNumber)</code>
int	<code>class_getVersion(Class aClass)</code>

# System Functions

## Manage run-time structures:

id	objc_getClass(const char * <i>aClassName</i> )
id	objc_lookUpClass(const char * <i>aClassName</i> )
id	objc_getMetaClass(const char * <i>aClassName</i> )
NXHashTable *	objc_getClasses(void)
void	objc_addClass(Class <i>aClass</i> )
Module *	objc_getModules(void)

## Dynamically load and unload classes:

long	objc_loadModules(char * <i>files</i> [], NXStream * <i>stream</i> , void (* <i>callback</i> )(Class, Category), struct mach_header ** <i>header</i> , char * <i>debugFilename</i> )
long	objc_unloadModules(NXStream * <i>stream</i> , void (* <i>callback</i> )(Class, Category))

## Send messages at run time:

id	objc_msgSend(id <i>theReceiver</i> , SEL <i>theSelector</i> , ...)
id	objc_msgSendSuper(struct objc_super * <i>superContext</i> , SEL <i>theSelector</i> , ...)
id	objc_msgSendv(id <i>theReceiver</i> , SEL <i>theSelector</i> , unsigned int <i>argSize</i> , marg_list <i>argFrame</i> )

## Make the run-time system thread safe:

void	objc_setMultithreaded(BOOL <i>flag</i> )
------	--

## Object Functions

Manage object memory:

id	<code>object_dispose(Object *anObject)</code>
id	<code>object_copy(Object *anObject, unsigned int indexedIvarBytes)</code>
id	<code>object_copyFromZone(Object *anObject, unsigned int indexedIvarBytes, NXZone *zone)</code>
id	<code>object_realloc(Object *anObject, unsigned int numBytes)</code>
id	<code>object_reallocFromZone(Object *anObject, unsigned int numBytes, NXZone *zone)</code>

Return the class name:

const char *	<code>object_getClassName(id anObject)</code>
--------------	---

Set and get instance variables:

Ivar	<code>object_setInstanceVariable(id anObject, const char *variableName, void *value)</code>
Ivar	<code>objectGetInstanceVariable(id anObject, const char *variableName, void **value)</code>

Return a pointer to an object's extra memory:

void *	<code>object_getIndexedIvars(id anObject)</code>
--------	--

## Method Functions and Macros

Get information about a method:

unsigned int	<code>method_getNumberOfArguments(Method aMethod)</code>
unsigned int	<code>method_getSizeOfArguments(Method aMethod)</code>
unsigned int	<code>method_getArgumentInfo(Method aMethod, int index, const char **type, int *offset)</code>

Examine and alter method argument values:

<i>type-name</i>	<code>marg_getValue(marg_list argFrame, int offset, type-name)</code>
<i>type-name</i> *	<code>marg_getRef(marg_list argFrame, int offset, type-name)</code>
void	<code>margSetValue(marg_list argFrame, int offset, type-name, type-name value)</code>

## Selector Functions

**Match method names with method selectors:**

SEL	<code>sel_getUid(const char *<i>aName</i>)</code>
const char *	<code>sel_getName(SEL <i>aSelector</i>)</code>

**Determine whether a selector is valid:**

BOOL	<code>sel_isMapped(SEL <i>aSelector</i>)</code>
------	---

**Register a method name:**

SEL	<code>sel_registerName(const char *<i>aName</i>)</code>
-----	---

# 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>	<code>id</code>	'@'
<code>_C_CLASS</code>	Class	'#'
<code>_C_SEL</code>	<code>SEL</code>	:
<code>_C_VOID</code>	<code>void</code>	'v'
<code>_C_CHR</code>	<code>char</code>	'c'
<code>_C_UCHR</code>	<code>unsigned char</code>	'C'
<code>_C_SHT</code>	<code>short</code>	's'
<code>_C_USHT</code>	<code>unsigned short</code>	'S'
<code>_C_INT</code>	<code>int</code>	'i'
<code>_C_UINT</code>	<code>unsigned int</code>	'I'
<code>_C_LNG</code>	<code>long</code>	'l'
<code>_C_ULNG</code>	<code>unsigned long</code>	'L'
<code>_C_FLT</code>	<code>float</code>	'f'
<code>_C_DBL</code>	<code>double</code>	'd'
<code>_C_UNDEF</code>	an undefined type	'?'
<code>_C_PTR</code>	a pointer	'^'
<code>_C_CHARPTR</code>	<code>char *</code>	'*''
<code>_C_BFLD</code>	a bitfield	'b'
<code>_C_ARY_B</code>	begin an array	'[
<code>_C_ARY_E</code>	end an array	']'
<code>_C_UNION_B</code>	begin a union	'('
<code>_C_UNION_E</code>	end a union	')'
<code>_C_STRUCT_B</code>	begin a structure	'{'
<code>_C_STRUCT_E</code>	end a structure	'}'

## 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 syms;
};
```

```
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

**- initOnDevice:*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*
- soundStreamDidRecordData:(void \*)*data*  
*size:(unsigned int)numBytes* Invoked when the sound driver overruns  
*forBuffer:(int)tag* Delivers a buffer of recorded data

# NXSoundDevice

Inherits From: Object

## Initializing and Freeing an NXSoundDevice

- init
  - initOnHost:(const char \*)*hostName*
  - free
- Initializes a newly allocated NXSoundDevice  
Initializes a new NXSoundDevice on the given host  
Frees the NXSoundDevice

## Using a Separate Thread

- + (cthread\_t)replyThread
  - + setUseSeparateThread:(BOOL)*flag*
  - + (BOOL)isUsingSeparateThread
  - + setThreadThreshold:(int)*threshold*
  - + (int)threadThreshold
- Returns the thread in which driver messages are sent  
Sets whether a separate thread is used for driver messages  
Returns whether a separate thread is used for driver messages  
Sets the message-reception threshold  
Returns the message-reception threshold

## Examining Ports

- (port_t) <b>devicePort</b>	Returns the port used to communicate with the sound driver
+ (port_t) <b>replyPort</b>	Returns the port to which the sound driver sends messages
- (port_t) <b>streamOwnerPort</b>	Returns the port used to connect to the sound driver

## Identifying the Host Computer

- (const char *) <b>host</b>	Returns the name of the computer on which the NXSoundDevice was initialized
------------------------------	---

## Configuring the Object

- (NXSoundDeviceError) <b>setBufferCount:</b> (unsigned int) <i>count</i>	Sets the number of DMA buffers for the sound device
- (unsigned int) <b>bufferCount</b>	Returns the number of DMA buffers for the sound device
- (unsigned int) <b>bufferSize</b>	Returns the size in bytes of each DMA buffer
- (NXSoundDeviceError) <b>setReserved:</b> (BOOL) <i>flag</i>	Sets whether the underlying device is reserved
- (BOOL) <b>isReserved</b>	Returns whether the underlying device is reserved
- (NXSoundDeviceError) <b>setBufferSize:</b> (unsigned int) <i>bytes</i>	Sets the size in bytes of each DMA transfer buffer
+ <b>setTimeout:</b> (unsigned int) <i>milliseconds</i>	Sets the timeout period
+ (unsigned int) <b>timeout</b>	Returns the timeout period

## Finding Peak Amplitudes

- (NXSoundDeviceError) <b>getPeakLeft:</b> (float *) <i>leftAmp</i> <i>right:</i> (float *) <i>rightAmp</i>	Returns the most recent peak amplitudes by reference
- (NXSoundDeviceError) <b>setDetectPeaks:</b> (BOOL) <i>flag</i>	Sets whether the NXSoundDevice will detect peaks
- (BOOL) <b>isDetectingPeaks</b>	Returns whether the NXSoundDevice is detecting peaks
- (NXSoundDeviceError) <b>setPeakHistory:</b> (unsigned int) <i>bufferCount</i>	Sets the number of buffers over which peaks are detected
- (unsigned int) <b>peakHistory</b>	Returns the number of buffers over which peaks are detected

## Controlling Streams

- <b>abortStreams:</b> <i>sender</i>	Aborts all streams connected to this NXSoundDevice
- <b>pauseStreams:</b> <i>sender</i>	Pauses all streams connected to this NXSoundDevice
- <b>resumeStreams:</b> <i>sender</i>	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 *) <i>leftDB</i> <i>right:(float *)rightDB</i>	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 *) <i>hostName</i>	Returns the sound-out port on <i>hostName</i>
---	---

---

## NXSoundStream

**Inherits From:** Object

### Initializing and Freeing an NXSoundStream

– init	Initializes the NXSoundStream
– initOnDevice: <i>aDevice</i>	Initializes the NXSoundStream and connects it to <i>aDevice</i>
– free	Deactivates and frees the NXSoundStream.

### Setting the Device

– (NXSoundDeviceError)setDevice: <i>aDevice</i>	Connects the NXSoundStream to <i>aDevice</i>
– 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 \*)*name*  
    **fromBundle:**(NXBundle \*)*aBundle*
- + **addName:**(const char \*)*name*  
    **fromSection:**(const char \*)*sectionName*
- + **addName:**(const char \*)*name*  
    **fromSoundfile:**(const char \*)*filename*
- **initFromSection:**(const char \*)*sectionName*
- **initFromPasteboard:**(Pasteboard \*)*thePboard*
- **initFromSoundfile:**(const char \*)*filename*
- **free**

Creates a Sound object from the sound resource named *name* in the NXBundle *aBundle*

Creates a Sound object from the *sectionName* section of the sound segment in the application executable file

Creates a Sound object from *filename*

Creates a Sound object from the *sectionName* section of the sound segment in the application executable file

Creates a Sound object from the named pasteboard

Creates a Sound object from *filename*

Frees the Sound object

### Accessing the Sound Name Table

- + **addName:**(const char \*)*name* **sound:***aSound*
- + **findSoundFor:**(const char \*)*aName*
- + **removeSoundForName:**(const char \*)*name*

Assigns the name *name* to the Sound *aSound* and adds it to the name table

Finds and returns the named Sound object

Removes the named Sound from the name table

### Accessing the Sound's Name

- **setName:**(const char \*)*aName*
- (const char \*)*name*

Set's the Sound object's name

Return's the Sound object's name

### Reading and Writing Sound Data

- (int)**readSoundfile:**(const char \*)*filename*
- **readSoundFromStream:**(NXStream \*)*stream*
- (int)**writeSoundfile:**(const char \*)*filename*
- **writeSoundToStream:**(NXStream \*)*stream*
- (int)**writeToPasteboard:**(Pasteboard \*)*pboard*

Replaces the Sound's data with that in *filename*

Replaces the Sound's data with that read from *stream*

Writes the Sound's data to *filename*

Writes the Sound's data to *stream*

Writes the Sound's data to the named pasteboard

## Modifying Sound Data

- (int)convertToFormat:(int)*newFormat*  
    *samplingRate*:(double)*newRate*  
    *channelCount*:(int)*newChannelCount*
- (int)convertToFormat:(int)*newFormat*
- (int)setDataSize:(int)*newDataSize*  
    *dataFormat*:(int)*newDataFormat*  
    *samplingRate*:(double)*newSamplingRate*  
    *channelCount*:(int)*newChannelCount*  
    *infoSize*:(int)*newInfoSize*
- setSoundStruct:(SNDSoundStruct \*)*aStruct*  
    *soundStructSize*:(int)*size*

Converts the Sound's data to the specified format,  
sampling rate, and channel count

Convert's the Sound's data to the specified format  
Set's the Sound's data as specified

Set's the Sound's sound structure

## Querying the Sound Data

- (SNDSoundStruct \*)soundStruct
- (int)soundStructSize
- (unsigned char \*)data
- (int)dataFormat
- (int) dataSize
- (int) channelCount
- (double) samplingRate
- (int) sampleCount
- (double) duration
- (char \*)info
- (int) infoSize
- (BOOL) isEmpty
- (BOOL) compatibleWith:*aSound*

Returns the Sound's sound structure

Gives the size of the Sound's sound structure

Returns a pointer to the Sound's sound data

Returns the Sound's data format

Returns the size in bytes of the Sound's data

Returns the number of channels of sound

Returns the sound data's sampling rate

Returns the number of sample frames in the sound data

Returns the sound's length in seconds

Returns a pointer to the Sound's info string

Returns the length in bytes of the Sound's info string

Returns whether the Sound contains any sound data

Returns whether the Sound's format is compatible with  
that of *aSound*

## Recording and playing

- (int) pause
- pause:*sender*
- (BOOL) isPlayable
- (int) play
- play:*sender*
- (int) record
- record:*sender*
- (int) resume

Pauses the Sound's activity

Pauses the Sound's activity

Returns whether the Sound can be played

Plays the Sound

Plays the Sound

Records into the Sound

Records into the Sound

Resumes the Sound's activity

<b>– resume:<i>sender</i></b>	Resumes the Sound's activity
<b>– (int)stop</b>	Stops the Sound's activity
<b>– stop:<i>sender</i></b>	Stops the Sound's activity
<b>– (int)samplesProcessed</b>	Returns the number of sample frames played or recorded
<b>– (int)status</b>	Returns the Sound's activity code
<b>– soundBeingProcessed</b>	Returns self
<b>– (SNDSoundStruct *)soundStructBeingProcessed</b>	Returns the sound structure that's being played or recorded
<b>– (int)processingError</b>	Returns the last error code that was generated

## Editing Sound Data

<b>– (BOOL)isEditable</b>	Returns whether the Sound's data can be edited
<b>– (int)copySamples:<i>aSound</i> at:(int)<i>startSample</i> count:(int)<i>sampleCount</i></b>	Copies a range of samples from <i>aSound</i> into the receiver
<b>– (int)copySound:<i>aSound</i></b>	Replaces the Sound's data with that in <i>aSound</i>
<b>– (int)deleteSamples</b>	Removes the Sound's data
<b>– (int)deleteSamplesAt:(int)<i>startSample</i> count:(int)<i>sampleCount</i></b>	Removes a range of samples from the Sound's data
<b>– (int)insertSamples:<i>aSound</i> at:(int)<i>startSample</i></b>	Inserts <i>aSound</i> 's data into the Sound's data
<b>– (BOOL)needsCompacting</b>	Returns whether the Sound's data needs to be compacted
<b>– (int)compactSamples</b>	Compacts the Sound's data

## Archiving the Object

<b>– finishUnarchiving</b>	Invoked automatically after unarchiving
<b>– read:(NXTypedStream *)<i>stream</i></b>	Unarchives the Sound from <i>stream</i>
<b>– write:(NXTypedStream *)<i>stream</i></b>	Archives the Sound to <i>stream</i>

## Accessing the Delegate

<b>– setDelegate:<i>anObject</i></b>	Sets the Sound's delegate object
<b>– delegate</b>	Returns the Sound's delegate
<b>– tellDelegate:(SEL)<i>theMessage</i></b>	Sends <i>theMessage</i> to the delegate

## Accessing the Sound Hardware

<b>+ getVolume:(float *)<i>left</i> :(float *)<i>right</i></b>	Returns the left and right volume settings by reference
<b>+ setVolume:(float)<i>left</i> :(float)<i>right</i></b>	Sets the left and right volumes (0.0 to 1.0)

+ (BOOL)isMuted	Returns whether the internal speaker is muted
+ setMute:(BOOL) <i>aFlag</i>	Sets whether the internal speaker is muted

## Delegate Methods

- didPlay: <i>sender</i>	Sent to the delegate when the Sound stops playing
- didRecord: <i>sender</i>	Sent to the delegate when the Sound stops recording
- hadError: <i>sender</i>	Sent to the delegate if an error occurs during recording or playback
- willPlay: <i>sender</i>	Sent to the delegate when the Sound begins to play
- willRecord: <i>sender</i>	Sent to the delegate when the Sound begins to record

## SoundMeter

Inherits From: View : Responder : Object

### Initializing a SoundMeter

- initFrame:(const NXRect *) <i>frameRect</i>	Initializes the SoundMeter
---	----------------------------

### Graphic Attributes

- setBackgroundGray:(float) <i>aValue</i>	Sets the SoundMeter's background color
- (float)backgroundGray	Returns the SoundMeter's background color
- setForegroundGray:(float) <i>aValue</i>	Sets the SoundMeter's running bar color
- (float)foregroundGray	Returns the color of the running bar
- setBezeled:(BOOL) <i>aFlag</i>	Sets whether a bezeled border is drawn
- (BOOL)isBezeled	Returns whether the SoundMeter has a border
- setPeakGray:(float) <i>aValue</i>	Sets the SoundMeter's peak bubble color
- (float)peakGray	Returns the SoundMeter's peak bubble gray

### Metering Attributes

- setSound: <i>aSound</i>	Sets the SoundMeter's Sound object
- sound	Returns the Sound object that the SoundMeter is metering

<code>- setFloatValue:(float)<i>aValue</i></code>	Sets the current running value
<code>- setHoldTime:(float)<i>seconds</i></code>	Sets the SoundMeter's peak value hold time in seconds
<code>- (float)holdTime</code>	Returns the SoundMeter's peak hold time

### Retrieving Meter Values

<code>- (float)floatValue</code>	Returns the current running amplitude value
<code>- (float)maxValue</code>	Returns the maximum running value so far
<code>- (float)minValue</code>	Returns the minimum running value so far
<code>- (float)peakValue</code>	Returns the most recently detected peak value

### Operating the Object

<code>- run:<i>sender</i></code>	Starts the SoundMeter running
<code>- (BOOL)isRunning</code>	Returns whether the SoundMeter is currently running
<code>- stop:<i>sender</i></code>	Stops the SoundMeter's metering activity

### Drawing the Object

<code>- drawCurrentValue</code>	Draws the SoundMeter's running bar and peak bubble
<code>- drawSelf:(const NXRect *)<i>rects</i> :(int)<i>rectCount</i></code>	Draws all the components of the SoundMeter

### Archiving

<code>- read:(NXTypedStream *)<i>aStream</i></code>	Unarchives the SoundMeter by reading it from <i>aStream</i>
<code>- write:(NXTypedStream *)<i>aStream</i></code>	Archives the SoundMeter by writing it to <i>aStream</i>

## SoundView

**Inherits From:** View : Responder : Object

### Initializing and Freeing a SoundView

<code>- initFrame:(const NXRect *)<i>frameRect</i></code>	Initializes the SoundView
<code>- free</code>	Frees the SoundView

## Modifying the Object

– <b>scaleToFit</b>	Fits the sound data within the current frame
– <b>setBackgroundGray:(float)aGray</b>	Sets the SoundView's background gray
– <b>setBezeled:(BOOL)aFlag</b>	Sets whether the SoundView has a bezeled border
– <b>setContinuous:(BOOL)aFlag</b>	Sets the state of continuous action messages.
– <b>setDelegate:<i>anObject</i></b>	Sets the SoundView's delegate
– <b>setDisplayMode:(int)aMode</b>	Sets the SoundView's display mode
– <b>setEnabled:(BOOL)aFlag</b>	Enables or disables the SoundView
– <b>setForegroundGray:(float)aGray</b>	Sets the SoundView's foreground gray
– <b>setOptimizedForSpeed:(BOOL)<i>flag</i></b>	Sets whether the SoundView's display mechanism is optimized
– <b>setSound:<i>aSound</i></b>	Sets the SoundView's Sound object
– <b>sizeToFit</b>	Resizes the SoundView's frame to maintain a constant reduction factor

## Querying the Object

– <b>(float)backgroundGray</b>	Returns the SoundView's background gray
– <b>delegate</b>	Returns the SoundView's delegate
– <b>(int)displayMode</b>	Returns the SoundView's display mode
– <b>(float)foregroundGray</b>	Returns the SoundView's foreground gray
– <b>getSelection:(int *)<i>firstSample</i> size:(int *)<i>sampleCount</i></b>	Returns the selection by reference
– <b>(BOOL)isAutoScale</b>	Returns whether the SoundView is in autoscaling mode
– <b>(BOOL)isBezeled</b>	Returns whether the SoundView has a bezeled border
– <b>(BOOL)isContinuous</b>	Returns whether the SoundView responds to mouse-dragged events
– <b>(BOOL)isEnabled</b>	Returns whether the SoundView is enabled
– <b>(BOOL)isOptimizedForSpeed</b>	Returns whether the SoundView is optimized for speedy display
– <b>(float)reductionFactor</b>	Returns the SoundView's reduction factor
– <b>sound</b>	Returns the SoundView's Sound object.

## Selecting and Editing the Sound Data

– <b>copy:<i>sender</i></b>	Copies the current selection to the pasteboard
– <b>cut:<i>sender</i></b>	Deletes the current selection
– <b>delete:<i>sender</i></b>	Deletes the current selection
– <b>mouseDown:(NXEvent *)<i>theEvent</i></b>	Allows a selection to be defined

– <b>paste:sender</b>	Replaces the current selection
– <b>selectAll:sender</b>	Creates a selection over the SoundView's entire sound
– <b>setSelection:(int)firstSample size:(int)sampleCount</b>	Sets the selection
– <b>(BOOL)isEditable</b>	Returns whether the SoundView's data can be edited.
– <b>setEditable:(BOOL)aFlag</b>	Sets whether the SoundView can be edited

## Pasteboard and Services Support

– <b>pasteboard:<i>thePasteboard</i> provideData:(const char *)<i>pboardType</i></b>	Places the SoundView's sound on the given pasteboard
– <b>readSelectionFromPasteboard:<i>thePasteboard</i></b>	Replaces the SoundView's current selection
– <b>validRequestorForSendType:(NXAtom)<i>sendType</i> andReturnType:(NXAtom)<i>returnType</i></b>	You never invoke this method
– <b>writeSelectionToPasteboard:<i>thePasteboard</i> types:(NXAtom *)<i>pboardTypes</i></b>	Places a copy of the SoundView's current selection on the given pasteboard

## Modifying the Display Coordinates

– <b>setAutoscale:(BOOL)<i>aFlag</i></b>	Sets the SoundView's automatic scaling mode
– <b>setReductionFactor:(float)<i>reductionFactor</i></b>	Recomputes the size of the SoundView's frame

## Drawing the Object

– <b>drawSelf:(const NXRect *)<i>rects</i> :(int)<i>rectCount</i></b>	Displays the SoundView's sound data
– <b>drawSamplesFrom:(int)<i>first</i> to:(int)<i>last</i></b>	Redisplays the given range of samples
– <b>hideCursor</b>	Hides the SoundView's cursor
– <b>showCursor</b>	Displays the SoundView's cursor
– <b>sizeTo:(NXCoord)<i>width</i> :(NXCoord)<i>height</i></b>	Sets the width and height of the SoundView's frame

## Responding to Events

– <b>(BOOL)acceptsFirstResponder</b>	Returns YES
– <b>becomeFirstResponder</b>	Promotes the SoundView to first responder
– <b>resignFirstResponder</b>	Resigns the position of first responder

## Performing the Sound Data

– <b>(BOOL)isPlayable</b>	Returns whether the SoundView's sound can be played
– <b>play:<i>sender</i></b>	Play the current selection
– <b>record:<i>sender</i></b>	Replaces the SoundView's current selection

– <b>pause:sender</b>	Pauses the current playback or recording
– <b>resume:sender</b>	Resumes the current playback or recording
– <b>stop:sender</b>	Stops the SoundView’s current recording or playback
– <b>soundBeingProcessed</b>	Returns the Sound object that’s currently being played or recorded into

## Archiving the Object

– <b>read:(void *)stream</b>	Unarchives the SoundView by reading it from <i>stream</i>
– <b>write:(void *)stream</b>	Archives the SoundView by writing it to <i>stream</i>

## Accessing the Delegate

– <b>didPlay:sender</b>	Used to redirect delegate messages
– <b>didRecord:sender</b>	Used to redirect delegate messages
– <b>hadError:sender</b>	Used to redirect delegate messages
– <b>tellDelegate:(SEL)theMessage</b>	Sends <i>theMessage</i> to the SoundView’s delegate
– <b>willPlay:sender</b>	Used to redirect delegate messages
– <b>willRecord:sender</b>	Used to redirect delegate messages

## Delegate Methods

– <b>didPlay:sender</b>	Sent to the delegate just after the SoundView is played.
– <b>didRecord:sender</b>	Sent to the delegate just after the SoundView is recorded into.
– <b>hadError:sender</b>	Sent to the delegate if an error is encountered .
– <b>selectionChanged:sender</b>	Sent to the delegate when the SoundView’s selection changes.
– <b>soundDidChange:sender</b>	Sent to the delegate when the SoundView’s sound is edited
– <b>willFree:sender</b>	Sent to the delegate when the SoundView is freed.
– <b>willPlay:sender</b>	Sent to the delegate just before the SoundView’s sound is played.
– <b>willRecord:sender</b>	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)**

### **Speed up or slow down playback of ATC sound**

```
int          SNDDropATCSamples(int numSamples, int bySamples)
int          SNDInsertATCSamples(int numSamples, int bySamples)
```

### **Modify volume or equalization for ATC playback**

```
int          SNDSetATCGain(float level)
int          SNDGetATCGain(float *level)
int          SNDSetATCEqualizerGains(int numBands, float *gains)
int          SNDGetATCEqualizerGains(int numBands, float *gains)
int          SNDScaleATCEqualizerGains(int numBands, float *gainScalars)
```

### **Set or get ATC parameters**

```
int          SNDSetATCSquelchThresholds(int numBands, float *thresholds)
int          SNDGetATCSquelchThresholds(int numBands, float *thresholds)
int          SNDUseDefaultATCSquelchThresholds(void)
```

### **Set and get compression attributes used in recording**

```
int          SNDSetCompressionOptions(SNDSoundStruct *sound, int bitFaithful,
                                      int compressionAmount)
int          SNDGetCompressionOptions(SNDSoundStruct *sound, int *bitFaithful,
                                      int *compressionAmount)
```

## **Converting Sound Data**

### **Convert between logarithmic and linear units**

```
float        SNDConvertDecibelsToLinear(float dB)
float        SNDConvertLinearToDecibels(float linear)
```

### **Convert a sound's attributes**

```
int          SNDConvertSound(SNDSoundStruct *fromSound, SNDSoundStruct **toSound)
unsigned char SNDMulaw(short linearValue)
short        SNDiMulaw(unsigned char mulawValue)
```

## 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        *SNDSError(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    nddriver_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)
```

## **Set and get sound playback attributes**

```
kern_return_t    snddriver_set_device_parms(port_t devicePort, boolean_t speakerOn,
                                         boolean_t filterOn, boolean_t zeroFill)
kern_return_t    snddriver_get_device_parms(port_t devicePort, boolean_t *speakerOn,
                                         boolean_t *filterOn, boolean_t *zeroFill)
kern_return_t    snddriver_set_volume(port_t devicePort, int leftVolume, int rightVolume)
kern_return_t    snddriver_get_volume(port_t devicePort, int *leftVolume, int *rightVolume)
kern_return_t    snddriver_set_ramp(port_t devicePort, int rampOn)
```

## **Stream Setup and Access**

### **Configure stream transfer buffers**

```
kern_return_t    snddriver_set_sndout_bufcount(port_t devicePort, port_t sndoutPort, int count)
kern_return_t    snddriver_set_sndout_bufsize(port_t devicePort, port_t sndoutPort, int size)
kern_return_t    snddriver_stream_ndma(port_t streamPort, int regionTag, int count)
```

### **Control and query a stream**

```
kern_return_t    snddriver_stream_control(port_t streamPort, int regionTag, int control)
kern_return_t    snddriver_stream_nsamples(port_t streamPort, int *byteCount)
```

### **Configure a sound stream**

```
kern_return_t    snddriver_stream_setup(port_t devicePort, port_t ownerPort, int dataPath,
                                         int sampleCount, int sampleSize, int lowWater, int highWater,
                                         int *protocol, port_t *streamPort)
```

### **Send data to and retrieve data from a stream**

```
kern_return_t    snddriver_stream_start_writing(port_t streamPort, void *data, int sampleCount,
                                              int regionTag, boolean_t preempt,
                                              boolean_t deallocateWhenDone, boolean_t msgStarted,
                                              boolean_t msgCompleted, boolean_t msgAborted,
                                              boolean_t msgPaused, boolean_t msgResumed,
                                              boolean_t msgUnderrun, port_t replyPort)
kern_return_t    snddriver_stream_start_reading(port_t streamPort, char *filename,
                                              int sampleCount, int regionTag, boolean_t msgStarted,
                                              boolean_t msgCompleted, boolean_t msgAborted,
                                              boolean_t msgPaused, boolean_t msgResumed,
                                              boolean_t msgOverrun, port_t replyPort)
```

# 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

- |  |                                       |
|--|---------------------------------------|
| – <b>init</b>                            | Initializes with 0 frame size         |
| – <b>initFrame:(const NXRect *)fRect</b> | Initializes with specified frame size |
| – <b>free</b>                            | Frees the N3DCamera                   |

#### All Drawing

- |  |  |
|--|--|
| – <b>(BOOL)lockFocus</b>                       | YES if PostScript and RenderMan drawing lock on camera |
| – <b>unlockFocus</b>                           | Unlocks PostScript and RenderMan drawing               |
| – <b>drawSelf:(NXRect *)rects :(int)nRects</b> | Performs all RIB and PostScript drawing                |

## RenderMan Drawing

– <b>render</b>	Renders the camera and any content shapes
– <b>renderSelf:(RtToken)context</b>	Override to perform custom rendering in the camera
– <b>setFlushRIB:(BOOL)flag</b>	Sets the receiver to invoke <b>flushRIB</b> within <b>render</b>
– <b>doesFlushRIB</b>	Tests whether the receiver invokes <b>flushRIB</b> within <b>render</b>
– <b>flushRIB</b>	Waits until all RIB code has been rendered

## PostScript Drawing

– <b>drawPS:(NXRect *)rects :(int)nRects</b>	Override to perform custom PostScript drawing
--	---

## Background Color

– <b>setBackgroundColor:(NXColor)color</b>	Sets the NXColor filled behind all drawing
– <b>(NXColor)backgroundColor</b>	The NXColor filled behind all drawing
– <b>setDrawBackgroundColor:(BOOL)flag</b>	If <i>flag</i> is YES, fills color behind all drawing
– <b>(BOOL)doesDrawBackgroundColor</b>	YES if camera fills color behind all drawing

## PostScript Transformation Management

– <b> setFrame:(const NXRect *)fRect</b>	Sets frame for both PostScript and RenderMan coordinate systems
– <b>moveTo:(NXCoord)x :(NXCoord)y</b>	Moves both PostScript and RenderMan coordinate systems
– <b>moveBy:(NXCoord)deltaX :(NXCoord)deltaY</b>	Moves both PostScript and RenderMan coordinate systems
– <b>sizeTo:(NXCoord)width :(NXCoord)height</b>	Resizes both PostScript and RenderMan coordinate systems
– <b>sizeBy:(NXCoord)deltaWidth :(NXCoord)deltaHeight</b>	Resizes both PostScript and RenderMan coordinate systems
– <b>rotateTo:(NXCoord)angle</b>	Prevents rotation of PostScript coordinate system
– <b>rotateBy:(NXCoord)deltaAngle</b>	Prevents rotation of PostScript coordinate system

## Shape Hierarchy Management

– <b>setWorldShape:a3DShape</b>	Sets world shape
– <b>worldShape</b>	Returns world shape

## Global Light Management

– <b>addLight:aLight</b>	Adds an N3DLight to the camera's global light list
– <b>removeLight:aLight</b>	Removes an N3DLight to the camera's global light list
– <b>lightList</b>	Returns the camera's global light list

## Picking

- **selectShapesIn:(const NXRect \*)selectionRect** Returns a List of N3DShapes in *selectionRect*

## Projection Rectangle

- **setProjectionRectangle:(float)left  
:(float)right  
:(float)top  
:(float)bottom** Sets the 3D coordinate system projection rectangle
- **getProjectionRectangle:(float \*)left  
:(float \*)right  
:(float \*)top  
:(float \*)bottom** Returns the 3D coordinate system's projection rectangle

## 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  
toward:(RtPoint)toPoint  
roll:(float)aRollAngle** Sets the eye-to-viewpoint vector and roll
- **getEyeAt:(RtPoint \*)fromPoint  
toward:(RtPoint \*)toPoint  
roll:(float \*)aRollAngle** Gets the eye-to-viewpoint vector and roll
- **moveEyeBy:(float)sDistance  
:(float)tDistance  
:(float)uDistance** Moves camera in its own coordinate system
- **rotateEyeBy:(float)dElev :(float)dAzim  
about:(RtPoint)pivotPtr** Rotates the camera in its own coordinates

## Clipping Planes

- **setClipPlanesNear:(float)aNearPlane  
far:(float)aFarPlane**
- **getClipPlanesNear:(float \*)aNearPlane  
far:(float \*)aFarPlane**

Sets the camera's near and far clipping planes

Returns the camera's near and far clipping planes

## Field of View

- **setFieldOfViewByAngle:(float)viewAngle**
- **setFieldOfViewByFocalLength:(float)aLength  
(float)fieldOfView**

Sets the viewing angle of the camera

Converts a focal length into a viewing angle for the camera

Returns the viewing angle of the camera

## Pixel Aspect Ratio

- **setPixelAspectRatio:(float)theRatio**
- **(float)pixelAspectRatio**

Sets the pixel aspect ratio for the camera

Returns the pixel aspect ratio for the camera

## Converting Coordinates

- **convertPoints:(RtPoint \*)points  
count:(int)n  
fromSpace:aShape**
- **convertPoints:(NXPoint \*)mcoords  
count:(int)pointCount  
toWorld:(RtPoint \*)wcoords**

Converts an array of points to PostScript coordinates

Converts PostScript points to world coordinates

## Crop Windows

- **(int)numCropWindows**
- **cropInRects:(NXRect \*)theRects  
nRects:(int)rectCount**

Count of rectangle divisions for photoreal rendering

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  
methods

**didRenderStream:(NXStream \*)imageStream**

**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 *)came 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 <i>name</i>

### Archiving

- awake	Performs additional initialization after unarchiving
---------	--

N3DLight

**Inherits From:** N3DShape : Object

## Initializing

- init** Initializes the receiver as an N3D\_AmbientLight

## Setting Light Type

- |  |   |
|--|---|
| <b>- setType:(N3DLightType)<i>aType</i></b>  | Sets the receiver's light type                        |
| <b>- (N3DLightType)<i>type</i></b>   | Returns the receiver's light type                     |
| <b>- makeAmbientWithIntensity:(RtFloat)<i>intensity</i></b>  | Sets type N3D_AmbientLight with appropriate parameter |
| <b>- makePointFrom:(RtPoint)<i>from</i><br/>    <i>intensity:</i>(RtFloat)<i>intensity</i></b>   | Sets type N3D_PointLight with appropriate parameters  |
| <b>- makeDistantFrom:(RtPoint)<i>fromPoint</i><br/>    <i>to:</i>(RtPoint)<i>toPoint</i><br/>    <i>intensity:</i>(RtFloat)<i>i</i></b>  | Sets type N3D_PointLight with appropriate parameters  |
| <b>- makeSpotFrom:(RtPoint)<i>fromPoint</i><br/>    <i>to:</i>(RtPoint)<i>toPoint</i><br/>    <i>coneAngle:</i>(RtFloat)<i>coneAngle</i><br/>    <i>coneDelta:</i>(RtFloat)<i>deltaAngle</i><br/>    <i>beamDistribution:</i>(RtFloat)<i>distribution</i><br/>    <i>intensity:</i>(RtFloat)<i>intensity</i></b> | Sets type N3D_SpotLight with appropriate parameters   |

# Setting Light Parameters

- |   |  |
|---|--|
| <b>- setFrom:(RtPoint)fromPoint</b>   | Sets the from point                                    |
| <b>- setFrom:(RtPoint)fromPoint<br/>to:(RtPoint)toPoint</b>   | Sets the from and to points                            |
| <b>- getFrom:(RtPoint *)fromPoint<br/>to:(RtPoint *)toPoint</b>   | Returns the from and to points                         |
| <b>- setConeAngle:(RtFloat)coneAngle<br/>coneDelta:(RtFloat)coneDelta<br/>beamDistribution:(RtFloat)distribution</b>      | Sets the cone angle, cone delta, and beam distribution |
| <b>- getConAngle:(RtFloat *)coneAngle<br/>coneDelta:(RtFloat *)coneDelta<br/>beamDistribution:(RtFloat *)distribution</b> | Gets the cone angle, cone delta, and beam distribution |
| <b>- setIntensity:(RtFloat)intensity</b>  | Sets the intensity                                     |
| <b>- (RtFloat)intensity</b>   | Returns the intensity                                  |

## **Rendering**

- **renderSelf:(N3DCamera \*)theCamera**
- **renderGlobal:(N3DCamera \*)theCamera**

Renders the light as a local light  
Renders the light as a global light

## **Global Light Management**

- **setGlobal:(BOOL)flag**
- **(BOOL)isGlobal**

Override to add behavior on becoming/resigning global  
YES if light is global

## **Switching On and Off**

- **switchLight:(BOOL)flag**

If flag is YES, turns receiver on

## **Setting Color**

- **setColor:(NXColor)theColor**
- **(NXColor)color**

Sets the receiver's color  
Returns the receiver's color

## **Archiving**

- **read:(NXTypedStream \*)theStream**
- **write:(NXTypedStream \*)theStream**
- **awake**

Reads the receiver from the stream  
Writes the receiver to the stream  
Performs additional initialization after unarchiving

---

## **N3DMovieCamera**

**Inherits From:** N3DCamera : View : Responder : Object

### **Initializing**

- **initFrame:(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

– <b>setFrameNumber:(int)<i>aFrameNumber</i></b>	Sets the current frame number
– <b>(int)frameNumber</b>	Returns the current frame number
– <b>setStartFrame:(int)<i>start</i></b>	Sets counters for movie
<b>endFrame:(int)<i>end</i></b>	
<b>incrementFramesBy:(int)<i>skip</i></b>	
– <b>(int)startFrame</b>	Returns the first frame number
– <b>(int)endFrame</b>	Returns the last frame number
– <b>(int)frameIncrement</b>	Returns the frame increment

## Archiving

– <b>read:(NXTypedStream *)<i>theStream</i></b>	Reads the receiver from the stream
– <b>write:(NXTypedStream *)<i>theStream</i></b>	Writes the receiver to the stream
– <b>awake</b>	Performs additional initialization after unarchiving

---

## N3DRenderPanel

**Inherits From:** Panel : Window : Responder : Object

### Initializing the Class

+ <b>initialize</b>	Initializes the class with data from the defaults database
+ <b>new</b>	Creates, if necessary, and returns an N3DRenderPanel

### Setting Accessory View

– <b>accessoryView</b>	Returns the accessory view
– <b>setAccessoryView:<i>aView</i></b>	Sets the accessory view

### Running Modal

– <b>(int)runModal</b>	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

- **initFromFile:(const char \*)*ribfile*** Initializes the receiver from a file
- **initFromStream:(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**
- **getSize:(NXSize \*)theSize**

Returns the rectangle that bounds the image  
Returns the size of the image

## Background Color

- **(NXColor)backgroundColor**
- **setBackgroundColor:(NXColor)theColor**

Returns the background color  
Sets the background color

## Hidden Surface Removal Type

- **(N3DHider)hider**
- **setHider:(N3DHider)theHider**

Returns the hider type for rendering images  
Sets the hider type for rendering images

## Surface Type

- **setSurfaceType:(N3DSurfaceType)type**
- **(N3DSurfaceType)surfaceType**

Sets the surface type used for rendering images  
Returns the surface type used for rendering images

## Archiving

- **read:(NXTypedStream \*)theStream**
- **write:(NXTypedStream \*)theStream**

Reads the receiver from the stream  
Writes the receiver to the stream

---

# N3DRotator

**Inherits From:** Object

## Initializing

- **init**
- **initWithCamera:*myCamera***

Initializes the receiver  
Initializes the receiver and sets its camera

## Setting Parameters

- **setCamera:*myCamera***
- **setCenter:(const NXPoint \*)*center* andRadius:(float)*radius***

Sets the receiver's camera  
Sets the receiver's center point and radius

## Axes of Rotation

- **setRotationAxis:**(N3DAxis)*axis*
- (N3DAxis)**rotationAxis**

Sets the axes about which the receiver rotates  
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*
- **write:**(NXTypedStream \*)*theStream*

Reads the receiver from the stream  
Writes the receiver to the stream

---

## N3DShader

Inherits From: Object

### Initializing and Freeing

- **init**
- **initWithShader:**(const char \*)*aShader*
- **free**

Initializes the receiver with no shader file  
Initializes the receiver with a shader file  
Frees the receiver

### Shader Language Object File

- **setShader:**(const char \*)*aShader*
- (const char \*)**shader**

Sets the receiver's shader  
Returns the receiver's shader

### Shader Color

- **setColor:**(NXColor)*aColor*
- (NXColor)**color**
- **setUseColor:**(BOOL)*flag*
- (BOOL)**doesUseColor**

Sets the receiver's color  
Returns the receiver's color  
Sets the receiver to apply its color  
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 *) <i>theCamera</i>	Renders the shape and its descendants
– renderSelf:(N3DCamera *) <i>theCamera</i>	Override to implement actual rendering
– renderSelfAsBox:(N3DCamera *) <i>theCamera</i>	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: <i>aPeer</i>	Inserts <i>aPeer</i> between receiver and its next peer
– linkDescendant: <i>aDescendant</i>	Inserts <i>aDescendant</i> between receiver and its descendant
– linkAncestor: <i>anAncestor</i>	Sets receiver's ancestor, returns previous ancestor
– unlink	Unlinks the receiver, reconnects peers and descendants
– group: <i>anAncestor</i>	Makes the receiver a descendant of <i>anAncestor</i>
– ungroup	Removes receiver from hierarchy, promotes descendant

## Shader

- **setShader:***aShader*
- **shaderType:**(SLO\_TYPE)*type*

Sets an N3DShader for the shape

Returns the N3DShader of *type*

## Surface

- (N3DSurfaceType)**surfaceType**
- **setSurfaceType:**(N3DSurfaceType)*theSurface*  
andDescendants:(BOOL)*flag*

Returns the receiver's surface type

Sets receiver's surface type; if flag is YES, sets descendants surface types

## Bounding Box

- **getBoundingBox:**(RtBound \*)*boundingBox*
- **setDrawAsBox:**(BOOL)*flag*
- (BOOL)**doesDrawAsBox**
- **getBounds:**(NXRect \*)*boundingRect*  
inCamera:*theCamera*

Returns by reference the receiver's bounding box

Sets the receiver to draw only its bounding box

YES if the receiver draws only its bounding box

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*
- **convertPoints:**(RtPoint \*)*points*  
count:(int)*n*  
fromAncestor:(N3DShape \*)*theShape*
- **convertPoints:**(RtPoint \*)*points*  
count:(int)*n*  
toAncestor:(N3DShape \*)*theShape*

Converts points from the receiver's coordinate system to that of *camera*

Converts points from an ancestor's coordinate system to that of the receiver

Converts points from the receiver's coordinate system to that of an ancestor

## Selectability

- **setSelectable:**(BOOL)*flag*
- (BOOL)**isSelectable**

Sets whether the receiver can be selected

YES if the receiver can be selected

## Visibility

- **setVisible:**(BOOL)*flag*
- (BOOL)**isVisible**

Sets the receiver to render when **renderSelf:** is invoked

YES if receiver renders when **renderSelf:** is invoked

## Naming Shapes

- **setShapeName:**(const char \*)*aName*
- (const char \*)**shapeName**

Sets the name of the shape  
Returns the name of the shape

## Delegate for Rendering

- **setRenderDelegate:***aShape*
- **removeRenderDelegate**
- **renderDelegate**

Sets the rendering delegate for the receiver  
Removes and returns the render delegate  
Returns the render delegate

## Transformation Matrices

- **setTransformMatrix:**(RtMatrix)*newTransform*
- **getTransformMatrix:**(RtMatrix)*transform*
- **concatTransformMatrix:**(RtMatrix)*theMatrix*  
**premultiply:**(BOOL)*flag*
- **getCompositeTransformMatrix:**(RtMatrix)*theMatrix*  
**relativeToAncestor:**(N3DShape \*)*theAncestor*
- **getInverseCompositeTransformMatrix:**(RtMatrix)*theMatrix*  
**relativeToAncestor:**(N3DShape \*)*theAncestor*

Sets the receiver's transformation matrix  
Returns the receiver's transformation matrix  
Multiplies the transform matrix with *theMatrix*,  
premultiplying by *theMatrix* if flag is YES  
Returns the matrix representing the transformation  
between *theAncestor*'s space and the receiver's  
Returns the matrix representing the transformation  
between the receiver's space and *theAncestor*'s

## Rotation, Scaling, Translation

- **rotateAngle:**(float)*ang*  
**axis:**(RtPoint)*referencePoint*
- **preRotateAngle:**(float)*angle*  
**axis:**(RtPoint)*referencePoint*
- **scale:**(float)*xScaleFactor*  
:(float)*yScaleFactor*  
:(float)*zScaleFactor*
- **prescale:**(float)*xScaleFactor*  
:(float)*yScaleFactor*  
:(float)*zScaleFactor*
- **scaleUniformly:**(float)*scaleFactor*
- **prescaleUniformly:**(float)*scaleFactor*

Rotates the receiver about its origin and the point  
Premultiplies the rotation of the receiver about its  
origin and the point  
Scales the receiver  
Scales the receiver, premultiplying the transformation  
Scales the receiver uniformly in all axes  
Scales the receiver uniformly in all axes, premultiplying the  
transformation

- **translate:**(float)*xTranslation*  
:(float)*yTranslation*  
:(float)*zTranslation*
- **pretranslate:**(float)*xTranslation*  
:(float)*yTranslation*  
:(float)*zTranslation*

## Archiving

- **read:**(NXTypedStream \*)*theStream*
- **write:**(NXTypedStream \*)*theStream*
- **awake**

Translates the receiver

Translates the receiver, premultiplying the transformation

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 eval;
        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

– **initFrame:(const NXRect \*)frameRect**      Initializes the receiver

#### Freeing an NXLiveVideoView

– **free**      Freed 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 *) <i>theImage</i> fromRect:(NXRect *) <i>sourceRect</i> toRect:(NXRect *) <i>destRect</i>	Grabs the image in the <i>sourceRect</i> and places it in <i>theImage</i> in <i>destRect</i>
– (BOOL)doesGrabOnStop	YES if the receiver grabs an image when video stops
– setGrabOnStop:(BOOL) <i>flag</i>	Sets the receiver to grab an image when video stops

## Finding the Video Resource

+ (BOOL)doesRectSupportVideo:(const NXRect) <i>theRect</i> standard:(int *) <i>theStandard</i> size:(NXSize *) <i>theSize</i>	YES if the specified rectangle can display video
+ (BOOL)doesScreenSupportVideo:(const NXScreen *) <i>theScreen</i> standard:(int *) <i>theStandard</i> size:(NXSize *) <i>theSize</i>	YES if the specified screen can display video
+ (BOOL)doesWindowSupportVideo: <i>theWindow</i> standard:(int *) <i>theStandard</i> size:(NXSize *) <i>theSize</i>	YES if the specified window can display video
+ (const NXScreen *)videoScreen	The screen best suited for video display

## Getting the Video Rectangle

– getSourceVideoRect:(NXRect *) <i>sourceRect</i>	Returns the visible portion of the video view
---	---

## Selecting the Video Input Port

– selectInput:(int) <i>inputPortNumber</i>	Sets the video input port
– (int)numInputs	Returns the number of video ports

## Setting the Output Mode:

– setOutputMode:(int) <i>outputMode</i>	Sets the output mode of the video view
---	--

## Controlling Input Video Quality

– setInputBrightness:(float) <i>brightness</i>	Sets the input brightness value
– (float)inputBrightness	Returns the input brightness value
– setInputGamma:(float) <i>inputGamma</i>	Sets the input gamma value
– (float)inputGamma	Returns the input gamma value
– setInputHue:(float) <i>hue</i>	Sets the input hue
– (float)inputHue	Returns the input hue

– <b>setInputSaturation:</b> (float) <i>saturation</i>	Sets the input saturation
– <b>(float)inputSaturation</b>	Returns the input saturation
– <b>setInputSharpness:</b> (float) <i>sharpness</i>	Sets the input sharpness
– <b>(float)inputSharpness</b>	Returns the input sharpness
– <b>resetPictureDefaults</b>	Restores all input video settings to their defaults

## Controlling Output Video Quality

– <b>setOutputGamma:</b> (float) <i>outputGamma</i>	Sets the output gamma
– <b>(float)outputGamma</b>	Returns the output gamma

## Getting the Video Standard

– <b>getVideoStandard:</b> (int *) <i>standard</i>	Returns the video standard and size
– <b>NXSize *vidRectSize</b>	

## Setting Output Genlock

– <b>setOutputGenlocked:</b> (BOOL) <i>locked</i>	Locks video output to input signal
– <b>(BOOL)outputGenlocked</b>	YES if video output is genlocked to input signal

## Drawing

– <b>drawSelf:</b> (const NXRect *) <i>rects</i> :(int) <i>rectCount</i>	Overridden to assure video updates correctly
– <b>drawVideoBackground:</b> (const NXRect *) <i>rects</i> :(int) <i>rectCount</i>	Invoked by drawSelf:: to assure correct video display

## Setting a Delegate

– <b>delegate</b>	Returns the receiver's delegate
– <b>setDelegate:</b> <i>anObject</i>	Sets the receiver's delegate

## Archiving

– <b>read:</b> (NXTypedStream *) <i>stream</i>	Reads the receiver from the stream
– <b>write:</b> (NXTypedStream *) <i>stream</i>	Writes the receiver to the stream

## Delegate Method

– <b>videoDidActivate:</b> <i>sender</i>	Notifies the delegate when video activates
– <b>videoDidSuspend:</b> <i>sender</i>	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

# 19 *Workspace Manager*

## Class

---

### WMInspector

Inherits From: Object

#### Accessing the Inspector Object

+ new

Creates a new WMInspector if none exists, or returns the existing one

#### Accessing Panel Controls

– okButton

Returns the id of the Inspector's OK button

– revertButton

Returns the id of the Inspector's Revert button

– window

Returns the id of the window that contains the user interface for the inspector

#### Accessing Workspace Selection

– (unsigned)selectionCount

Returns the number of items selected in the File Viewer

– selectionPathsInto:(unsigned)pathString  
separator:(char)character

Returns the paths of the files selected in the File Viewer

## 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/cthreads.h**:

```
#include <mach/cthreads.h>
```

## Basic C-Thread Functions

### Control a thread:

cthread_t	<b>cthread_fork(any_t (*function)(), any_t arg)</b>
any_t	<b>cthread_join(cthread_t t)</b>
void	<b>cthread_detach(cthread_t t)</b>
void	<b>cthread_yield()</b>
kern_return_t	<b>cthread_abort(cthread_t t)</b>
void	<b>cthread_exit(any_t result)</b>

### Get access to a thread:

cthread_t	<b>cthread_self()</b>
thread_t	<b>cthread_thread(cthread_t t)</b>

**Associate a string with a thread:**

char *	<b>cthread_name(cthread_t <i>t</i>)</b>
void	<b>cthread_set_name(cthread_t <i>t</i>, char *<i>name</i>)</b>

**Associate data with a thread:**

any_t	<b>cthread_data(cthread_t <i>t</i>)</b>
void	<b>cthread_set_data(cthread_t <i>t</i>, any_t <i>data</i>)</b>

**Change the priority of a thread:**

kern_return_t	<b>cthread_max_priority(cthread_t <i>t</i>, processor_set_t <i>processor_set</i>, int <i>max_priority</i>)</b>
kern_return_t	<b>cthread_priority(cthread_t <i>t</i>, int <i>priority</i>, boolean_t <i>set_max</i>)</b>

**Get or set the UNIX™ error number of this thread:**

int	<b>cthread_errno()</b>
void	<b>cthread_set_errno_self(int <i>error</i>)</b>

**Get or set the maximum number of threads in this task:**

int	<b>cthread_limit()</b>
void	<b>cthread_set_limit(int <i>limit</i>)</b>
int	<b>cthread_count()</b>

## Mutex Functions

**Control a mutex:**

mutex_t	<b>mutex_alloc()</b>
void	<b>mutex_init(struct mutex *<i>m</i>)</b>
void	<b>mutex_clear(struct mutex *<i>m</i>)</b>
void	<b>mutex_free(mutex_t <i>m</i>)</b>

#### **Associate a string with a mutex:**

char *	<b>mutex_name(mutex_t <i>m</i>)</b>
void	<b>mutex_set_name(mutex_t <i>m</i>, char *<i>name</i>)</b>

#### **Synchronize a mutex:**

void	<b>mutex_lock(mutex_t <i>m</i>)</b>
int	<b>mutex_try_lock(mutex_t <i>m</i>)</b>
void	<b>mutex_unlock(mutex_t <i>m</i>)</b>

## **Condition Functions**

#### **Control a condition variable:**

condition_t	<b>condition_alloc()</b>
void	<b>condition_init(struct condition *<i>c</i>)</b>
void	<b>condition_clear(struct condition *<i>c</i>)</b>
void	<b>condition_free(condition_t <i>c</i>)</b>

#### **Associate a string with a condition variable:**

char *	<b>condition_name(condition_t <i>c</i>)</b>
void	<b>condition_set_name(condition_t <i>c</i>, char *<i>name</i>)</b>

#### **Synchronize a condition variable:**

void	<b>condition_broadcast(condition_t <i>c</i>)</b>
void	<b>condition_signal(condition_t <i>c</i>)</b>
void	<b>condition_wait(condition_t <i>c</i>, mutex_t <i>m</i>)</b>

# 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	<b>task_create(task_t parent_task, boolean_t inherit_memory, task_t *child_task)</b>
kern_return_t	<b>task_suspend(task_t target_task)</b>
kern_return_t	<b>task_resume(task_t target_task)</b>
kern_return_t	<b>task_terminate(task_t target_task)</b>

## Access a task or its threads:

kern_return_t	<b>task_threads(task_t target_task, thread_array_t *thread_list,</b> <b>                  unsigned int *thread_count)</b>
kern_return_t	<b>task_info(task_t target_task, int flavor, task_info_t task_info,</b> <b>                  unsigned int *task_info_count)</b>

#### **Access a task's special ports:**

kern_return_t	<code>task_get_special_port(task_t task, int which_port, port_t *special_port)</code>
kern_return_t	<code>task_set_special_port(task_t task, int which_port, port_t special_port)</code>
port_t	<code>task_notify()</code>
task_t	<code>task_self()</code>

#### **Translate between Mach task and UNIX process ID:**

kern_return_t	<code>task_by_unix_pid(task_t task, int pid, task_t *result_task)</code>
kern_return_t	<code>unix_pid(task_t target_task, int *pid)</code>

#### **Set a task's scheduling priority:**

kern_return_t	<code>task_priority(task_t task, int priority, boolean_t change_threads)</code>
---------------	---

#### **Multiprocessor functions (not useful on single-processor systems):**

kern_return_t	<code>task_assign(task_t task, processor_set_t new_processor_set, boolean_t assign_threads)</code>
kern_return_t	<code>task_assign_default(task_t task, boolean_t assign_threads)</code>
kern_return_t	<code>task_get_assignment(task_t task, processor_set_t *processor_set)</code>

## **Thread Functions**

#### **Control a thread:**

kern_return_t	<code>thread_create(task_t parent_task, thread_t *child_thread)</code>
kern_return_t	<code>thread_suspend(thread_t target_thread)</code>
kern_return_t	<code>thread_resume(thread_t target_thread)</code>
kern_return_t	<code>thread_terminate(thread_t target_thread)</code>
kern_return_t	<code>thread_abort(thread_t target_thread)</code>

#### **Get information about a thread:**

kern_return_t	<code>thread_info(thread_t target_thread, int flavor, thread_info_t thread_info, unsigned int *thread_info_count)</code>
---------------	--

#### **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 recv_size,  
                      msg_timeout_t send_timeout, msg_timeout_t recv_timeout)
```

## Virtual Memory Functions

### Control virtual memory:

```
kern_return_t    vm_allocate(vm_task_t target_task, vm_address_t *address, vm_size_t size,
                           boolean_t anywhere)
kern_return_t    vm_deallocate(vm_task_t target_task, vm_address_t address, vm_size_t size)
kern_return_t    vm_protect(vm_task_t target_task, vm_address_t address, vm_size_t size,
                           boolean_t set_maximum, vm_prot_t new_protection)
kern_return_t    vm_inherit(vm_task_t target_task, vm_address_t address, vm_size_t size,
                           vm_inherit_t new_inheritance)
```

### Access or modify the contents of virtual memory:

```
kern_return_t    vm_copy(vm_task_t target_task, vm_address_t source_address, vm_size_t size,
                           vm_address_t dest_address)
kern_return_t    vm_read(vm_task_t target_task, vm_address_t address, vm_size_t size,
                        pointer_t *data, unsigned int *data_count)
kern_return_t    vm_write(vm_task_t target_task, vm_address_t address, pointer_t data,
                        unsigned int data_count)
kern_return_t    map_fd(int fd, vm_offset_t offset, vm_offset_t *address, boolean_t find_space,
                        vm_size_t size)
```

### Get information about virtual memory:

```
kern_return_t    vm_region(vm_task_t target_task, vm_address_t *address, vm_size_t *size,
                           vm_prot_t *protection, vm_prot_t *max_protection,
                           vm_inherit_t *inheritance, boolean_t *shared,
                           port_t *object_name, vm_offset_t *offset)
kern_return_t    vm_statistics(vm_task_t target_task, vm_statistics_data_t *vm_stats)
```

## Host Functions

### Get this host's port:

```
host_t          host_self()
host_priv_t     host_priv_self()
```

#### **Get information about a host:**

```
kern_return_t    host_info(host_t host, int flavor, host_info_t host_info, unsigned int  
                           *host_info_count)  
kern_return_t    host_kernel_version(host_t host, kernel_version_t version)
```

#### **Get the name or privileged port of a processor set:**

```
kern_return_t    host_processor_set_priv(host_priv_t host_priv,  
                                         processor_set_t processor_set_name,  
                                         processor_set_t *processor_set)  
kern_return_t    host_processor_sets(host_t host, processor_set_name_array_t *processor_set_list,  
                                     unsigned int *processor_set_count)
```

#### **Get the ports of all processors on a host:**

```
kern_return_t    host_processors(host_priv_t host_priv, processor_array_t *processor_list,  
                                 unsigned int *processor_count)
```

## **Processor Functions**

#### **Get information about a processor:**

```
kern_return_t    processor_info(processor_t processor, int flavor, host_t *host,  
                                processor_info_t processor_info,  
                                unsigned int *processor_info_count)
```

#### **Get the name of the default processor set:**

```
kern_return_t    processor_set_default(host_t host, processor_set_t *default_set)
```

#### **Change the allowed scheduling policies of a processor set:**

```
kern_return_t    processor_set_policy_enable(processor_set_t processor_set, int policy)  
kern_return_t    processor_set_policy_disable(processor_set_t processor_set, int policy,  
                                             boolean_t change_threads)
```

### **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	<b>kern_loader_look_up</b> (port_t * <i>loader_port</i> )
kern_return_t	<b>kern_loader_server_com_port</b> (port_t <i>loader_port</i> , port_t <i>task_port</i> , server_name_t <i>server_name</i> , port_t * <i>server_com_port</i> )

## Get or display an error string:

void	<b>kern_loader_error</b> (const char * <i>string</i> , kern_return_t <i>errno</i> )
const char *	<b>kern_loader_error_string</b> (kern_return_t <i>errno</i> )

## Affect the runnability of a loadable kernel server:

kern_return_t	<b>kern_loader_add_server</b> (port_t <i>loader_port</i> , port_t <i>task_port</i> , server_reloc_t <i>server_reloc</i> )
kern_return_t	<b>kern_loader_delete_server</b> (port_t <i>loader_port</i> , port_t <i>task_port</i> , server_name_t <i>server_name</i> )
kern_return_t	<b>kern_loader_load_server</b> (port_t <i>loader_port</i> , server_name_t <i>server_name</i> )
kern_return_t	<b>kern_loader_unload_server</b> (port_t <i>loader_port</i> , port_t <i>task_port</i> , server_name_t <i>server_name</i> )

**Get information from or about a loaded kernel server:**

```
kern_return_t    kern_loader_get_log(port_t loader_port, port_t server_com_port, port_t  
                                reply_port)  
kern_return_t    kern_loader_log_level(port_t loader_port, port_t server_com_port, int log_level)  
kern_return_t    kern_loader_server_info(port_t loader_port, port_t task_port,  
                                         server_name_t server_name, server_state_t *server_state,  
                                         vm_address_t *load_address, vm_size_t *load_size,  
                                         server_reloc_t relocatable, server_reloc_t loadable,  
                                         port_name_array_t *port_list, unsigned int *port_list_count,  
                                         port_name_string_array_t *port_names,  
                                         unsigned int *port_names_count, boolean_array_t *advertised,  
                                         unsigned int *advertised_count)  
kern_return_t    kern_loader_server_task_port(port_t loader_port, port_t kernel_port,  
                                         server_name_t server_name, port_t *server_task_port)
```

**Get general information from the kernel-server loader:**

```
kern_return_t    kern_loader_server_list(port_t loader_port, server_name_array_t *server_names,  
                                         unsigned int *server_names_count)  
kern_return_t    kern_loader_status_port(port_t loader_port, port_t listen_port)
```

**Request or handle an asynchronous message from the kernel-server loader:**

```
kern_return_t    kern_loader_ping(port_t loader_port, port_t ping_port, int id)  
kern_return_t    kern_loader_reply_handler(msg_header_t *msg,  
                                         kern_loader_reply_t *kern_loader_reply)
```

**Shut down or reconfigure the kernel-server loader:**

```
kern_return_t    kern_loader_abort(port_t loader_port, port_t priv_port, boolean_t restart)
```



## 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 Development Tools and Techniques: Release 3**
- **NeXTSTEP Object-Oriented Programming and the Objective C Language: Release 3**
- **NeXTSTEP User Interface Guidelines: Release 3**
- **NeXTSTEP Operating System Software: 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*



ISBN 0-201-63253-5