init
initIconCell:
initTextCell:
copyFromZone:
free

Determining component sizes calcCellSize:
calcCellSize:inRect:
calcDrawInfo:
getDrawRect:
getIconRect:
getTitleRect:

Setting the Cell's type setType:
type

Setting the Cell's state setState:
incrementState
state

Enabling and disabling the Cell setEnabled:
isEnabled

Setting the icon setIcon:
icon

Setting the Cell's value setDoubleValue:
doubleValue
setFloatValue:
floatValue
setIntValue:
intValue
setStringValue:
setStringValueNoCopy:
setStringValueNoCopy:shouldFree:
stringValue

Interacting with other Cells takeDoubleValueFrom:
takeFloatValueFrom:
takeIntValueFrom:
takeStringValueFrom:

Modifying text attributes setAlignment:
alignment
setFont:
font
setEditable:
isEditable
setSelectable:
isSelectable
setScrollable:

Formatting data setFloatingPointFormat:left:right:

Modifying graphic attributes setBezeled:
isBezeled
setBordered:
isBordered
isOpaque

Setting parameters setParameter:to:
getParameter:

Displaying controlView
drawInside:inView:
drawSelf:inView:
highlight:inView:lit:
isHighlighted

Target and action setAction:
action
setTarget:
target
setContinuous:
isContinuous
sendActionOn:

Assigning a tag setTag:
tag

Handling keyboard alternatives keyEquivalent

Tracking the mouse+ prefersTrackingUntilMouseUp
mouseDownFlags
getPeriodicDelay:andInterval:
trackMouse:inRect:ofView:
startTrackingAt:inView:
continueTracking:at:inView:
stopTracking:at:inView:mouseIsUp:

Managing the cursor resetCursorRect:inView:

Archiving read:
write:
awake

trackMouse:inRect:ofView:

(SEL)action

NX_CENTERED, or NX_RIGHTALIGNED.

setAlignment:

awake

Used during unarchiving to initialize static variables for the Cell class. Returns self.

read:

calcCellSize:(NXSize *)theSize

Returns by reference the minimum width and height required for displaying the Cell. This method
inRect: with the rectangle argument set to a rectangle with very large width and height. Override t
the proper way to calculate the minimum width and height required for displaying the Cell. Return

calcCellSize:inRect:

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

Returns by reference the minimum width and height required for displaying the Cell in the given re
possible to fit, the width and/or height could be bigger than the ones of the provided rectangle. Th
by trying to size the Cell so that it fits in the rectangle argument (for example, by wrapping the tex
made between extending the width or height of aRect to fit text, the height will be extended. Retu

calcCellSize:

calcDrawInfo:(const NXRect *)aRect

Does nothing and returns self. Objects using Cells generally maintain a flag that informs them if a
been modified in such a way that the location or size of the Cell should be recomputed. If so, calcS
invoked before displaying the Cell that method invokes Cell's calcDrawInfo: for each Cell.

calcSize (Matrix)

(BOOL)continueTracking:(const NXPoint *)lastPoint
at:(const NXPoint *)currentPoint
inView:aView

Determines whether or not the Cell should keep tracking the mouse based on the positions provide
can keep tracking, NO if should not. This method is invoked by trackMouse:inRect:ofView: as the
around inside the Cell. lastPoint and currentPoint should be in aView's coordinate system. By def
returns YES when the Cell is continuous (that is, when it should continually send action messages
pressed or dragged). This method is often overridden to provide more sophisticated tracking behav

trackMouse:inRect:ofView:, startTrackingAt:inView:, stopTracking:at:inView:mouseIsUp:

copyFromZone:(NXZone *)zone

Allocates, initializes, and returns a copy of the receiving Cell. The copy is allocated from zone and contents as the receiver. When you subclass Cell, override this method to send the message

copy (Object)

(double)doubleValue

Returns the receiving text Cell's value as a double-precision floating point number, by converting double using the standard C function atof(). Returns 0.0 if the Cell isn't a text Cell.

setDoubleValue:, floatValue, intValue, stringValue, type

drawInside:(const NXRect *)cellFrame inView:aView

Draws the ªinsideº of the Cell. For the base Cell class, it's the same as drawSelf:inView: except th bezel or border if there is one. cellFrame should be the frame of the Cell (that is, the same as the c drawSelf:inView:), not the rectangle returned by getDrawRect:. The PostScript focus must be lock method is invoked. If the Cell's highlight flag is YES, then the Cell is highlighted (by swapping li throughout cellFrame see the description of the Display PostScript operator compositerect for a de highlighting). Returns self.

drawInside:inView: is usually invoked from the Control class's drawCellInside: method and is use drawing to be done in order to update the value displayed by the Cell when the contents is changed important in more complex Cells such as ButtonCell and SliderCell.

All subclasses of Cell which override drawSelf:inView: must override drawInside:inView:. drawI never invoke drawSelf:inView:, but drawSelf:inView: canÐand often doesÐinvoke drawInside:in

drawSelf:inView:, lockFocus (View), highlight:inView:lit:, isHighlighted, compositerect (Displ

drawSelf:(const NXRect *)cellFrame inView:aView

Displays the contents of a Cell in a given rectangle of a given view. Your code must lock the focu invoking this method. It draws the border or bezel (if any), then invokes drawInside:inView:. A te in the rectangle by using a global Text object. An icon Cell displays its icon centered in the rectan rectangle, or by setting the icon origin on the rectangle origin if it doesn't fit. Nothing is displayed NX_NULLCELL. Override this method if you want a display that is specific to your own subclass

drawInside:inView:, lockFocus (View)

edit:(const NXRect *)aRect
inView:aView
editor:textObject
delegate:anObject

endEditing:textObject

Ends editing begun with edit:inView:editor:delegate:event: or select:inView:editor:delegate:start:le
method is invoked by the textDidEnd:endChar: method of the object you are using as the delegate
(most often a Matrix or TextField). This method should remove the Text object from the view hier
delegate to nil. Returns self.

edit:inView:editor:delegate:event:, select:inView:editor:delegate:start:length:, textDidEnd:endC
delegate method)

(int)entryType

Returns the type of data allowed in the Cell. See setEntryType: for the list of valid types.

setEntryType:

(float)floatValue

Returns the receiving text Cell's value as a single-precision floating point number, by converting it
double using the C function atof() and then casting the result to a float. Returns 0.0 if the receiver

setFloatValue:, doubleValue, intValue, stringValue, type

font

Returns the Font used to display text in the Cell. Returns nil if the receiver isn't a text Cell.

setFont:, type

free

Frees the memory used by the Cell and returns nil. If the Cell's contents was set by copy (the defa
is also freed.

getDrawRect:(NXRect *)theRect

Given the bounds of the Cell in theRect, this method changes it to be the rectangle into which the C
ªinsidesº (everything but a bezel or border), and returns it by reference. In other words, this metho
rectangle which is touched by drawInside:inView:. However, your code should not use the rectang
method as the argument to drawInside:inView:. Returns self.

getIconRect:, getTitleRect:, drawInside:inView:

(int)getParameter:(int)aParameter

Returns the value of one of the frequently accessed flags for a Cell. See setParameter:to: for a list corresponding methods. Since the parameters are also accessible through methods such as isEnabl you shouldn't need to use this method often.

setParameter:to:

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

Returns by reference two values: the amount of time (in seconds) that a continuous button will pau periodically send action messages to the target object, and the interval (also in seconds) at which th Periodic messaging behavior is controlled by Cell's sendActionOn: and setContinuous: methods. (the action message only on mouse up events.) Override this method to return your own values. Re

setContinuous:, sendActionOn:

getTitleRect:(NXRect *)theRect

Returns self, and, by reference in theRect, the rectangle into which the text will be drawn. If this C text, theRect is untouched. Your code should not use the rectangle returned by this method as the drawInside:inView:. Returns self.

getDrawRect:, getIconRect:, drawInside:inView:

highlight:(const NXRect *)cellFrame
inView:aView
lit:(BOOL)flag

If the Cell's highlight status is different from flag, sets the Cell's highlight status to flag and, if flag rectangle cellFrame in aView. Your code must lock focus on aView before invoking this method. composites with NX_HIGHLIGHT inside the bounds of cellFrame. Override this method if you w highlighting behavior in a Cell subclass. Returns self.

Note that the highlighting that the base Cell class does will not appear when printed (although subc TextFieldCell, SelectionCell, and ButtonCell can print themselves highlighted). This is because th transparent, and there is no concept of transparency in printed output.

isHighlighted, drawSelf:inView:, drawInside:inView:

(const char *)icon

Returns the name of the icon currently used by the Cell, if any, or NULL if the receiver isn't an ico

setIcon:, title

incrementState

init

Initializes and returns the receiver, a new Cell instance, as type NX_NULLCELL. This method is initializer for cells without either text or an icon.

initIconCell:, initTextCell:, setIcon:, setText:

initIconCell:(const char *)iconName

Initializes and returns the receiver, a new icon Cell instance (that is, its type is NX_ICONCELL). NXImage with the name iconName. If iconName is NULL or an image for iconName is not found initialized with a default icon, ªNXsquare16º. This method is the designated initializer for Cells th the Cell later has text assigned, its type will automatically change.

icon, setIcon:, initTextCell:, setText:, init, findImageFor: (NXImage), name (NXImage)

initTextCell:(const char *)aString

Initializes and returns the receiver, a new text Cell instance, (that is, its type is NX_TEXTCELL). to aString, or ªCellº if aString is NULL. This method is the designated initializer for text Cells.

title, setTitle:, initIconCell:, setIcon:, init

(int)intValue

Returns the receiving text Cell's value as an integer, by converting its string contents to an int usin Returns 0 if the receiver isn't a text Cell.

setIntValue:, doubleValue, floatValue, stringValue, type:

(BOOL)isBezeled

Returns YES if the Cell draws itself with a bezeled border, NO otherwise. The default is NO.

setBezeled:, isBordered

(BOOL)isBordered

Returns YES if the Cell draws itself surrounded by a 1-pixel black frame, NO otherwise. The defa

setBordered:, isBezeled

(BOOL)isContinuous

(BOOL)isEditable

Returns YES if text in the Cell is editable (and therefore also selectable), NO otherwise. The defau

setEditable:, isSelectable

(BOOL)isEnabled

Returns YES if the Cell is enabled, NO otherwise. The default is YES. A Cell's enabled status is handling and display: It affects the behavior of methods for mouse tracking and text editing, by al changes to the Cell within those methods, and only allows the Cell to highlight or set a cursor recta You can still affect many Cell attributes programmatically (setState:, for example, will still work).

setEnabled:, trackMouse:inRect:ofView:

(BOOL)isEntryAcceptable:(const char *)aString

Tests whether aString matches the Cell's entry type, as set by the setEntryType: method. Returns acceptable by the receiving Cell, NO otherwise. For example, a text Cell of type NX_INTTYPE a represent integers, but not floating point numbers or words. If aString is NULL or empty, this met

This method is invoked by Form, Matrix, and other Controls to see if a new text string is acceptabl method doesn't check for overflow. It can be overridden to enforce specific restrictions on what th Cell.

setEntryType:

(BOOL)isHighlighted

Returns YES if the Cell is highlighted, NO otherwise.

highlight:inView:lit:

(BOOL)isOpaque

Returns YES if the Cell is opaque (that is, if it draws over every pixel in its frame), NO otherwise. opaque if and only if it draws a bezel. Subclasses that draw differently should override this based their drawing.

setBezeled:

(BOOL)isScrollable

Returns YES if typing past an end of the text in the Cell will cause the Cell to scroll to follow the t return value is NO.

setScrollable:

Returns 0, as Cell provides no support for key equivalents. Subclasses can implement key equival
this method to return the key equivalent for the receiver.

setKeyEquivalent: (ButtonCell), keyEquivalent (ButtonCell)

(int)mouseDownFlags

Returns the flags (for example, NX_SHIFTMASK) that were set when the mouse went down to st
session. This method is only valid during tracking. It doesn't work if the target of the Cell initiate
loop as part of its action method (as does PopUpList).

sendActionOn:

read:(NXTypedStream *)stream

Reads the Cell from the typed stream stream.

write:, awake

resetCursorRect:(const NXRect *)cellFrame inView:aView

If the receiver is a textCell, then a cursor rectangle is added to aView (with addCursorRect:cursor:
cursor to change to an I-beam when it passes over the Cell. Override this method to change the cu
to provide a different cursor for a text Cell.

addCursorRect:cursor: (View, Control)

select:(const NXRect *)aRect
inView:aView
editor:aTextObject
delegate:anObject
start:(int)selStart
length:(int)selLength

Uses aTextObj to select text in the Cell identified by selStart and selLength, which will be highligh
though the user had dragged the cursor over it. This method is similar to edit:inView:editor:delega
can be invoked in any situation, not only on a mouse-down event.

edit:inView:editor:delegate:event:

(int)sendActionOn:(int)mask

Resets flags to determine when the action is sent to the target while tracking. Can be any logical c

NX_MOUSEUPMASK
NX_MOUSEDOWNMASK
NX_MOUSEDRAGGEDMASK

setAction:(SEL)aSelector

Does nothing. This method is overridden by Action Cell and its subclasses, which actually implen and action method. It is also overriden by NXBrowserCell to provide access to its NXBrowser's a self.

action, setTarget:

setAlignment:(int)mode

Sets the alignment of text in the Cell. mode should be one of three constants: NX_LEFTALIGNE or NX_RIGHTALIGNED. Returns self.

alignment, setWrap:

setBezeled:(BOOL)flag

If flag is YES, then the Cell draws itself surrounded by a bezel if NO, it doesn't. setBordered: and mutually exclusive. Returns self.

isBezeled, setBordered:

setBordered:(BOOL)flag

If flag is YES, then the Cell draws itself surrounded by a 1-pixel black frame if NO, it doesn't. set setBezeled: are mutually exclusive. Returns self.

isBordered, setBezeled:

setContinuous:(BOOL)flag

Sets whether a Cell continuously sends its action message to the target object when tracking. Norm set the continuous (cflags2.continuous) or the mouse-dragged flag (cflags2.actOnMouseDragged), setting is appropriate to the subclass implementing it. In the base Cell class, this method sets the c settings usually have meaning only for ActionCell and its subclasses which implement the instance that provide target/action functionality. Some Control subclasses, specifically Matrix, send a defau target when a Cell doesn't provide a target or action.

isContinuous, sendActionOn:

setDoubleValue:(double)aDouble

Sets the contents of the Cell to the string value representing the double-precision floating point nur the entry type of the Cell. Does nothing if the receiver isn't a text Cell. Returns self.

doubleValue, setFloatValue:, setIntValue:, setStringValue:, entryType, type

setEnabled:(BOOL)flag

Sets the enabled status of the Cell. A Cell's enabled status is used primarily in event handling and behavior of methods for mouse tracking and text editing, by allowing or disallowing changes to the methods, and only allows the Cell to highlight or set a cursor rectangle if it's enabled. Many Cell altered programmatically (setState:, for example, will still work). Returns self.

isEnabled

setEntryType:(int)aType

This method sets the data format allowed in the Cell. aType is one of these seven constants, allow corresponding numeric string values to be entered:

NX_ANYTYPENo restrictions
NX_INTTYPEInteger values
NX_FLOATTYPESingle-precision floating point values
NX_DOUBLETYPEDouble-precision floating point values
NX_POSINTTYPEPositive integer values
NX_POSFLOATTYPEPositive single-precision floating point values
NX_POSDOUBLETYPEPositive double-precision floating point values

If the receiver isn't a text Cell, it's converted to type NX_TEXTCELL, in which case its font is set font at 12.0 point, and its string value is set to ªCellº (even for text Cells that display numbers).

The entry type is checked by the isEntryAcceptable: method. That method is used by Controls tha (such as Matrix and TextField) to validate that what the user has typed is correct. If you want to ha accept some specific type of data (other than those listed above), override the isEntryAcceptable: m validity of the data the user has entered.

entryType, isEntryAcceptable:, setFloatingPointFormat:left:right:

setFloatingPointFormat:(BOOL)autoRange
left:(unsigned int)leftDigits
right:(unsigned int)rightDigits

Sets whether floating-point numbers are autoranged, and sets the sizes of the fields to the left and r point. leftDigits specifies the maximum number of digits to the left of the decimal point, and right number of digits to the right (the fractional digit places will be padded with zeros to fill this width) number is too large to fit its integer part in leftDigits digits, as many places as are needed on the le removed from rightDigits when the number is displayed.

If autoRange is YES, leftDigits and rightDigits are simply added to form a maximum total field wi for the decimal point). The fractional part will be padded with zeros on the right to fill this width, possible (up to removing the decimal point and displaying the number as an integer). The integer p never truncatedÐthat is, it is displayed in full no matter what the field width limit is.

leftDigits must be between 0 and 10. rightDigits must be between 0 and 14. If leftDigits is 0, then formatting applies. If rightDigits is 0, then the decimal and the fractional part of the floating-point (that is, the floating-point number is printed as if it were an integer). If the entry type of the Cell is NX_FLOATTYPE, NX_POSFLOATTYPE, NX_DOUBLETYPE, or NX_POSDOUBLETYPE, i NX_FLOATTYPE. Returns self.

setEntryType:

setFont:fontObject

Sets the Font to be used when displaying text in the Cell. Does nothing if the receiver isn't a text C

font

setIcon:(const char *)iconName

Sets the Cell's icon to iconName (an NXImage object with that name). iconName is stored as the NXImage is stored as its support. If the Cell isn't an icon cell, it's converted if the Cell was a text freed if necessary. If iconName is NULL or an empty string, or if an image can't be found for icon icon set to the standard system bitmap ªNXsquare16º.

If you specify a name for which an image can't be found, no change is made. Your code can verify properly changed by comparing the values returned by the type or icon methods before and after in Returns self.

icon, findImageNamed (NXImage), initIconCell:

setIntValue:(int)anInt

Sets the contents of the Cell to the string value representing the integer anInt. Does nothing if the Cell. This method ignores the entry type of the Cell. Returns self.

intValue, setDoubleValue:, setFloatValue:, setStringValue:, type, entryType

setParameter:(int)aParameter to:(int)value

Sets the value of one of the Cell's parameters to value, and returns self. You don't normally use th parameters can be set using specific methods such as setEditable:. In this method, the parameters i aParameter, a symbolic constant defined in the header file appkit/Cell.h. The following table lists corresponding methods for setting and getting the value of the related parameters:

getParameter:

setScrollable:(BOOL)flag

Sets whether the Cell will scroll to follow typing while being edited. Returns self.

isScrollable, edit:inView:editor:delegate:event:

setState:(int)value

Sets the state of the Cell to 0 if value is 0, to 1 otherwise. Returns self.

state, incrementState

setStringValue:(const char *)aString

Copies aString as the receiver's contents. If the receiver isn't a text Cell, this method converts it to
font to the user's system font at 12 points. Returns self.

If the receiver was an icon Cell, the NXImage for that icon is not freed your code should retrieve it
after sending this message.

If floating point formatting has been set (with setFloatingPointParameters:left:right:) and the entry
floating point number type, then the string is tested to determine whether it represents a floating po
string is displayed according to that floating point format.

setStringValueNoCopy:, setStringValueNoCopy:shouldFree:, stringValue, setDoubleValue:, se
setIntValue:, setFloatingPointFormat:left:right:

setStringValueNoCopy:(const char *)aString

Similar to setStringValue: but doesn't make a copy of aString. The Cell records that it doesn't hav
contents when it receives a free message. Note that if a string is set this way, floating-point format
(since a shared string can't be altered). Returns self.

setStringValue:, setStringValueNoCopy:shouldFree:, stringValue

setStringValueNoCopy:(char *)aString shouldFree:(BOOL)flag

Similar to setStringValueNoCopy:, but the sender can specify in flag if the contents should be free
receives a free message. Note that if a string is set this way, floating-point formatting isn't applied

If the contents was already the same string as aString (the same pointer, not the same string value),
can't be set set to YES. That is, you can't set a string as non-freeable and later change it to be free
method with that same string you can, however, change it from freeable to nonfreeable.

setStringValue:, setStringValueNoCopy:, stringValue

setTag:(int)anInt

Does nothing. This method is overridden by ActionCell and its subclasses to support Controls wit
(Matrix and Form). Override this method to provide a way to identify Cells. Returns self.

tag, findCellWithTag: (Matrix, Menu classes)

setTarget:anObject

override this method you must include this line first.

setType:(int)aType

Sets the type of the Cell. aType should be NX_TEXTCELL, NX_ICONCELL, or NX_NULLCEL
NX_TEXTCELL and the receiver isn't currently a text Cell, then the font is set to the user's syster
string value is set to ªCellº. If aType is NX_ICONCELL and the receiver isn't an icon Cell, then t
default, ªNXsquare16º.

type, init, initIconCell:, initTextCell:, setIcon:, setText:

setWrap:(BOOL)flag

If flag is YES, text will be wrapped to word breaks. If flag is NO, it will be truncated. The default
has effect only when displaying text, not when editing, and only applies to Cells whose alignment
NX_LEFTALIGNED (centered and right-aligned text always wraps to word breaks).

setAlignment:

(BOOL)startTrackingAt:(const NXPoint *)startPoint inView:aView

This method is invoked from trackMouse:inRect:ofView: the first time the mouse appears in the C
tracked. Override to provide implementation-specific tracking behavior. This method should retu
track based on this starting point, and only if the Cell is continuous otherwise it should return NO.

trackMouse:inRect:ofView:, continueTracking:at:inView:, stopTracking:at:inView:mouseIsUp:,
mouseDownFlags

(int)state

Returns the state of the Cell (0 or 1). The default is 0.

setState:, incrementState

stopTracking:(const NXPoint *)lastPoint
at:(const NXPoint *)stopPoint
inView:aView
mouseIsUp:(BOOL)flag

Returns the contents of the Cell as a string.

setStringValue:, doubleValue, floatValue, intValue

(int)tag

Returns 1. This method is overridden by ActionCell and its subclasses to support multiple-Cell co
Form). Override this method if you want to use tags to identifiy Cells. Returns self.

setTag:, findCellWithTag: (Matrix, Menu classes)

takeDoubleValueFrom:sender

Sets the Cell's double-precision floating point value to the value returned by sender's doubleValue
be of a class that implements the doubleValue method. Returns self.

This method can be used in action messages between Cells. It permits one Cell (the sender) to affe
Cell (the receiver). For example, a TextFieldCell can be made the target of a SliderCell, which wi
takeDoubleValueFrom: action message. The TextFieldCell will get the return value of the SliderC
method, turn it into a text string, and display it.

takeDoubleValueFrom: (Control), setDoubleValue:

takeFloatValueFrom:sender

Sets the Cell's single-precision floating-point value to the value returned by sender's floatValue m
of a class that implements the floatValue method. Returns self.

This method is similar to takeDoubleValueFrom: except it works with floats rather than doubles.

takeFloatValueFrom: (Control), setFloatValue:

takeIntValueFrom:sender

Sets the Cell's integer value to the value returned by sender's intValue method. sender must be of
the intValue method. Returns self.

This method is similar to takeDoubleValueFrom: except it works with ints rather than doubles.

takeIntValueFrom: (Control), setIntValue:

takeStringValueFrom:sender

Sets the Cell's string value to the value returned by sender's stringValue method. sender must be
implements the stringValue method. Returns self.

This method is similar to takeDoubleValueFrom: except it works with strings rather than doubles.

takeStringValueFrom: (Control), setStringValue:

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

Invoked by a Control to initiate the tracking behavior of a Cell. It's generally not overridden since implementation invokes other Cell methods that can be overridden to handle specific events in a dr Returns YES if the mouse goes up in cellFrame, NO otherwise.

This method first invokes startTrackingAt:inView:. If that method returns YES, then as mouse-dra intercepted, continueTracking:at:inView: is invoked, and, finally, when the mouse leaves the boun button goes up, stopTracking:at:inView:mouseIsUp: is invoked (if cellFrame is NULL, then the bo infinitely large). You usually override one or more of these methods to respond to specific mouse

If the other tracking methods are insufficient for your needs, override this method directly. It's thi responsibility to invoke aView's sendAction:to: method when appropriate (before, during, or after YES if and only if the mouse goes up within the Cell during tracking. If the Cell's action is sent o then startTrackingAt:inView: is invoked before the action is sent and the mouse is tracked until it g bounds. If the Cell sends its action periodically, then the action is sent periodically to the target ev moving (although continueTracking:at:inView: is only invoked when the mouse changes position) sent on a mouse dragged event, then continueTracking:at:inView: is invoked before the action is se Cell is incremented (with incrementState) before the action is sent and after stopTracking:at:inVie mouse goes up.

startTrackingAt:inView:, continueTracking:at:inView:, stopTracking:at::inView:mouseIsUp:

(int)type

Returns the type of the Cell, which can be either NX_NULLCELL, NX_ICONCELL or NX_TEX

setType:

write:(NXTypedStream *)stream

Writes the Cell to the typed stream stream. Returns self.

read: