Paul S. McCarthy and Eric Sunshine -- January 17, 1996

MiscTableScroll

Inherits From: ScrollView : View : Responder : Object

Declared In: MiscTableScroll.h

Class Description

This class provides a convenient and powerful user-interface object for displaying and manipulating tabular data. The appearance and behavior is similar to DBTableView, but it does not depend on obsolete packages (DBKit), nor on separately bundled packages (EOF). It does not support DBKit-style adaptors, but future versions might. Although it inherits from ScrollView, the programmatic interface is similar to the Matrix class.

User Interface Highlights

Scrollable Matrix.
Column and Row titles.
Columns and Rows can be resized.
Columns and Rows can be dragged (rearranged).
Expand-to-data columns.
Alt-Click works in Highlight mode.
Keyboard access as well as mouse access.
Automatically resorts rows when columns are rearranged.

Programmatic Interface Highlights

Each column and row can have its own size.
Each column can have its own cell-type.
Lazy-mode for large amounts of data.
Programmatic interface for multiple selection.
Built-in sort support.
Smart memory management.
Delegate methods for most features.
Simple indexed access to rows and columns.

Similarities between Rows and Columns -- Slots and Borders

Rows and columns are treated equally wherever it is practical and desirable to do so. Almost every action and option that is available for columns is also available for rows and vice versa. *Slot* is the generic term for a single column or row. *Border* is the generic term for row or column orientation. *Size* is the generic term for width or height. Most methods come in two flavors: a row/column specific flavor that uses "row" or "col" as part of the name; and a generic flavor that has a "border" argument and (when needed) a "slot" argument. Here are some

examples:

Generic	Specific
- border:setTitlesOn:	- setColTitlesOn:
	- setRowTitlesOn:
- border:setSlot:size:	- setCol:size:
	- setRow:size:

Differences between Rows and Columns

There are some differences between rows and columns. This object is designed to maximize the efficiency of displaying many rows of data. Hence, it is faster to add and remove rows than columns. You should set up all your columns in InterfaceBuilder, or while the table is empty, then add and remove rows afterwards. Rows are cached for use with the -renewRows: method. This makes it very fast to change the contents of the table on a row-oriented basis. Despite the row-oriented bias, column-oriented operations can be performed at any time; they will just be slower than the corresponding row-oriented operations. This behavior is intrinsic to the implementation, it cannot be changed.

Cell-prototypes are only used for columns. This behavior can only be changed by subclassing.

Selection in the body of a table performs selection on a row-wise basis. This behavior can be changed programmatically via the **-trackBy:** method.

There are numerous default settings which differ between columns and rows. Most of these options can be changed in InterfaceBuilder; all of them can be changed programmatically. Here is a summary of the defaults which differ between rows and columns:

Option	Column Default	Row Default
modifier drag	NO	YES
uniform size	NO	YES
user sizeable	YES	NO
user draggable	YES	NO
titles displayed	YES	NO
autosize-slots	YES	NO

min-size-constrained	YES	NO
title mode	Custom	Auto-Numbered

Selection stuff...

Slot Sizing

Uniform size is the simplest sizing mode. When you set the uniform size of a border to any non-zero value, all slots in that border will have the same (uniform) size. They will not be sizeable by the user, nor will they be able to have individually differing sizes. Setting the uniform size of a border to zero disables this feature, enabling slots to have individual sizes, and enabling user-sizing of slots in the border (subject to further conditions described below). By default, rows are uniform size, columns are not.

If uniform sizing is not set for a border, the following sizing information is maintained for each slot in the border:

target size
minimum size

maximum size
data size
adjusted size
user-sizeable-flag
expands-to-data-flag
autosize-flag

Target size is the desired size for a particular slot. Minimum and maximum sizes are the lower and upper bounds for the size of a slot. Data size is the size of the largest item in a slot for slots that are marked expands-to-data. Adjusted size is the final display size of the slot after all other factors have been taken into account. When a slot is marked as user-sizeable, the user will be able to resize the slot (subject to further conditions described below). When a slot is marked as expands-to-data, the display size will be increased as necessary to display the largest entry in that slot. Note that expands-to-data only affects the adjusted size. It does not affect the target size. Also note that expands-to-data will only increase the size of a slot, never decrease it. When a slot is marked as autosize, its adjusted size may be changed to meet global minimum and maximum size limits. The three flag values are mutually independent, but you should be careful, because user-sizeable and autosize do not mix well. They cause bizarre, counter-intuitive behavior on narrow tables in wide views.

Users resize columns by dragging the right-hand edge of the column's title cell to the desired width. Likewise, users resize rows by dragging the bottom edge of the row's title cell to the desired height. The cursor changes to a horizontal or vertical resize cursor whenever the cursor is over one of the resizing areas. When users resize a slot, they are setting the target size for the slot.

All of the following conditions must be met to enable the user to resize a particular slot:

(a) Uniform-size for the border must be disabled (-border:setUniformSize:0)
(b) The title cells must be displayed (-border:setTitlesOn:YES)
(c) The border in question must allow user-sizing of slots (-border:setSizeableSlots:YES)
(d) The slot in question must be user-sizeable (-border:setSlot:sizeable:YES)
(e) There must be some room to grow or shrink between the slot's current adjusted size and the slot's minimum and maximum sizes.

All of these conditions are met by default for new columns, unless you explicitly disable one of the global options for column sizing.

Slot Dragging and Indexing -- Visual vs. Physical

Dragging and sizing are independent of each other. You can have borders that are not sizeable in any way, but are still draggable, and vice versa. You can also have borders that are both draggable and sizeable, or neither draggable nor sizeable.

Users drag slots by dragging the title cells until the mouse is over the desired new location and "dropping" the slot in the new location. Dragging must be enabled for that border. By default, columns are dragged with an unmodified drag, and they are selected with a command-drag. By default, rows are selected with an unmodified drag, and they are dragged with a command-drag. By default, dragging is enabled for columns, but not for rows.

If slot-dragging is enabled for a border then an internal mapping vector is maintained which translates the original physical position of the slot to its current visual position. All programmer-interface methods and all delegate call-back methods use the original physical position of the slot so you can ignore the current visual ordering in your programs. If you need or want to examine the current visual ordering, you can do so with the

-border:slotPosition: and -border:slotAtPosition: methods.

Lazy vs. Eager...

Delegate stuff...

Prototype Cells

Each column maintains a prototype cell which is used when new rows are created. When new rows are created, a -copy message is sent to the prototype cell for each column, and the new copy of the prototype cell is put into the new row. This means that all prototype cells must implement the -copy method appropriately. They must make copies of all components that are freed when the cell itself is freed.

The prototype cell can be one of the built-in types (text, icon, or button), it can be supplied by the delegate, or you can set it programmatically. If you set a prototype cell programmatically, the MiscTableScroll object will take "ownership" of the prototype cell -- it will free the cell when it is finished with it. If the delegate provides the prototype cell, the delegate retains ownership -- the MiscTableScroll object will not free prototype cells provided

by the delegate.

Usage Tips

For simple, flexible and maintainable access to the columns of the table scroll, you should declare an **enum** which identifies the columns in the MiscTableScroll, like the following example from the ScrollDir program:

```
enum
   {
   ICON_SLOT,
   NAME_SLOT,
   SIZE_SLOT,
   MODIFIED_SLOT,
   PERMS_SLOT,
   OWNER_SLOT,
   GROUP_SLOT,
   HARDLINKS_SLOT,
   SOFTLINK_SLOT,
   MESSAGE_SLOT
```

```
};
```

Then you use the enumeration identifiers whenever you need to specify a column. Using an enumeration this way lets you add, remove and shuffle the slots just by updating the enum declaration, rather than searching through the code to find all the places that need to be fixed. It also makes your code more readable.

There are two standard patterns for putting the data into eager (non-lazy) MiscTableScroll objects: -renewRows: and -addRow.

-renewRows:
When you know the number of rows in advance, it is most efficient to use the -renewRows: method to tell the MiscTableScroll object the number of rows that you will need. Your code will usually be structured like the following:

```
id item;
int row;
int const NRows = [dataSource count];
```

```
[tableScroll renewRows: NRows];
for (row = 0;  row < NRows;  row++)
    {
    item = [dataSource itemAt:row];
    [tableScroll setRow:row tag:(int)item];
    [[tableScroll cellAt:row:NAME_SLOT]
        setStringValue:[item name]];
    [[[tableScroll cellAt:row:SIZE_SLOT]
        setIntValue:[item size]]
        setTag:[item size]];
    //... and so on ...
    }

if ([tableScroll autoSortRows])
   [tableScroll sortRows];
```

-addRow

When you do not know the final number of rows in advance, your code will usually be structured like the following:

```
int row = 0;
[tableScroll empty];

while ((item = [self getNextItem]) != 0)
   {
   [tableScroll addRow];
   [tableScroll setRow:row tag:(int)item];
   [[tableScroll cellAt:row:NAME_SLOT]
      setStringValue: [item name]];
   [[[tableScroll cellAt:row:SIZE_SLOT]
      setIntValue: [item size]]
      setTag: (int) [item size]];
   //... and so on ...
   row++;
   }

[tableScroll sizeToCells];

if ([tableScroll autoSortRows])
```

```
[tableScroll sortRows];
```

A common programming mistake is forgetting to call sizeToCells. You must call sizeToCells after you have finished adding rows so that the MiscTableScroll can update the frames of its various subviews. If you forget to call sizeToCells, the MiscTableScroll will appear to be empty when it is displayed.

Smart Memory Management

The MiscTableScroll class implements "smart" memory management. It does not allocate support structures until and unless they are needed. For example, since rows are uniform-size by default, the MiscTableScroll will not allocate the array of sizing-info structures until and unless you make the rows non-uniform size. In a complimentary fashion, if you make the columns uniform size, the MiscTableScroll object will free the sizing-info array for the columns. Similarly, custom titles must be stored in an array. However, no other title-mode requires this array, and the array will only exist for borders that have custom titles. Likewise, the visual-to-physical mapping vector that supports user-draggable slots is only created when the first slot is actually moved. Even if the draggable option is turned on, you will not incur the memory overhead until the option is used. The net result of all this is that you only pay for the features that you use.

On the other hand, you do pay for the features that you do use. These extra features exact a price in storage and cpu. You should be careful about using them for rows when you expect thousands of rows.

The MiscTableScroll class is designed to provide high-quality, consistent, flexible behavior to the user while supporting a wide range of load requirements -- from dozens of rows to hundreds of thousands of rows. Smart memory management is an important element in achieving that goal.

Instance Variables

Method Types

Creating and freeing instances	±€initFrame: ±€free
Delegates	±€setDelgate: ±€delegate ± setDataDelegate: ± dataDelegate
Transmitting action	±€setTarget: ±€target ± setDoubleTarget: ± doubleTarget ± setAction: ± action ± setDoubleAction: ± doubleAction ± sendAction:to:forAllCells:

	± sendAction:to: ± sendAction ± sendDoubleAction - sendActionIfEnabled - sendDoubleActionIfEnabled
Next-text chain	± setNextText: ±€nextText ± setPreviousText: ± previousText ± isSelectable - selectText:
Enabling and disabling	±€setEnabled: ±€isEnabled
Selection	± setSelectionMode: ±€selectionMode ± reflectSelection

± cellIsSelected::
± rowIsSelected:
± colIsSelected:
± selectedRow
± selectedCol
± selectedRowTags:
± selectedColTags:
± selectedRows:
± selectedCols:
± selectRow:
± selectCol:
± selectRowTags:
± selectColTags:
± selectRows:
± selectCols:
± selectAll:
± clearSelection

± border:setModifierDragSlots:
± setModifierDragCols:
± setModifierDragRows:
± modifierDragSlots:
± modifierDragCols
± modifierDragRows

- border:selectSlot:
- border:selectSlots:
- border:selectTags:
- border:selectedSlots:
    - border:selectedTags:
- border:slotIsSelected:
- borderClearSelection:
- borderHasMultipleSelection:
- borderHasSelection:

- borderNumSelectedSlots:
- borderSelectAll:
- borderSelectedSlot:
- clearColSelection
- clearRowSelection
- hasColSelection
- hasMultipleColSelection
- hasMultipleRowSelection
- hasRowSelection
- numSelectedCols
- numSelectedRows
- selectAllCols
- selectAllRows

- trackBy:
- trackingBy

Keyboard Cursor	- border:setCursor: - borderClearCursor: - borderCursor: - borderHasValidCursor: - clearCursor - clearCursorCol - clearCursorRow - cursorCol - cursorRow - hasValidCursorCol - hasValidCursorRow - setCursorCol: - setCursorRow:
Scrolling	- scrollCellToVisible:: - scrollColToVisible: - scrollRowToVisible:

- scrollSelToVisible

Titles

± border:setTitlesOn:
± setColTitlesOn:
± setRowTitlesOn:
± borderTitlesOn:
± colTitlesOn
± rowTitlesOn

± border:setTitleMode:
± setColTitleMode:
± setRowTitleMode:
± borderTitleMode:
± colTitleMode
± rowTitleMode

± border:setSlot:title:

	± setCol:title: ± setRow:title: ± border:slotTitle: ± colTitle: ± rowTitle:
Sizing	± border:setUniformSizeSlots: ± setUniformSizeCols: ± setUniformSizeRows: ± uniformSizeSlots: ± uniformSizeCols ± uniformSizeRows ± border:setSizeableSlots: ± setSizeableCols: ± setSizeableRows: ± sizeableSlots: ± sizeableCols:

± sizeableRows:

± border:setSlot:size:
± setCol:size:
± setRow:size:
± border:slotSize:
± colSize:
± rowSize:

± border:setSlot:minSize:
± setCol:minSize:
± setRow:minSize:
± border:slotMinSize:
± colMinSize:
± rowMinSize:

± border:setSlot:maxSize:

± setCol:maxSize:
± setRow:maxSize:
± border:slotMaxSize:
± colMaxSize:
± rowMaxSize:

± border:slotAdjustedSize:
± colAdjustedSize:
± rowAdjustedSize:

± border:setSlot:expandsToData:
± setCol:expandsToData:
± setRow:expandsToData:
± border:slotExpandsToData:
± colExpandsToData:
± rowExpandsToData:

± border:setSlot:autosize:
± setCol:autosize:
± setRow:autosize:
± border:slotIsAutosize:
± colIsAutosize:
± rowIsAutosize:

± border:setSlot:sizeable:
± setCol:sizeable:
± setRow:sizeable:
± border:slotIsSizeable:
± colIsSizeable:
± rowIsSizeable:

- border:constrainMaxTotalSize:
- border:constrainMinTotalSize:
- border:setMaxTotalSize:

- border:setMinTotalSize:
- border:setSlot:dataSize:
- border:slotDataSize:
- colDataSize:
- rowDataSize:
- constrainMaxTotalHeight:
- constrainMaxTotalWidth:
- constrainMinTotalHeight:
- constrainMinTotalWidth:
- constrainSize
- maxTotalHeight
- maxTotalHeightIsConstrained
- maxTotalSize:
- maxTotalSizeIsConstrained:
- maxTotalWidth
- maxTotalWidthIsConstrained
- minTotalHeight

- minTotalHeightIsConstrained
- minTotalSize:
- minTotalSizeIsConstrained:
- minTotalWidth
- minTotalWidthIsConstrained
- setMaxTotalHeight:
- setMaxTotalWidth:
- setMinTotalHeight:
- setMinTotalWidth:
- setCol:dataSize:
- setRow:dataSize:
- totalHeight
- totalSize:
- totalWidth

- border:slotResized:

Dragging	± border:setDraggableSlots: ± setDraggableCols: ± setDraggableRows: ± draggableSlots: ± draggableCols ± draggableRows ± border:setModifierDragSlots: ± setModifierDragCols: ± setModifierDragRows: ± modifierDragSlots: ± modifierDragCols ± modifierDragRows ± border:moveSlotFrom:to: ± moveColFrom:to:

	± moveRowFrom:to:
	± border:slotAtPosition: ± colAtPosition: ± rowAtPosition:
	± border:slotPosition: ± colPosition: ± rowPosition:
	- border:physicalToVisual: - border:visualToPhysical:
	- border:slotDraggedFrom:to:
Inserting and Deleting	± numSlots: ± numCols

± numRows

± addSlot:
± addCol
± addRow

± border:insertSlotAt:
± insertColAt:
± insertRowAt:

± border:deleteSlotAt:
± deleteColAt:
± deleteRowAt:

± empty
± emptyAndFreeCells
± renewRows:

Cell Prototypes	± border:setSlot:cellType: ± setCol:cellType: ± setRow:cellType: ± border:slotCellType: ± colCellType: ± rowCellType:
	± border:setSlot:cellPrototype: ± setCol:cellPrototype: ± setRow:cellPrototype: ± border:slotCellPrototype: ± colCellPrototype: ± rowCellPrototype:
Tags	± tag ± setTag:

± border:setSlot:tag:
± setCol:tag:
± setRow:tag:
± border:slotTag:
± colTag:
± rowTag:
± tagAt::

Drawing
± border:drawSlot:
± drawRow:
± drawCol:
± border:drawSlotTitle:
± drawRowTitle:
± drawColTitle:
± reflectSelection

± drawCellAt::

± setAutodisplay:
- getDocClipFrame:

Data control ± setLazy:
± isLazy
± cellAt::
± buffCount
± addSlot:
± border:insertSlotAt:
± border:deleteSlotAt:
± numSlots:

± addCol
± insertColAt:
± deleteColAt:
± numCols

± addRow
± insertRowAt:

± deleteRowAt:
± numRows
± renewRows
± empty
± emptyAndFreeCells

- doRetireCell:at::
- doReviveCell:at::
- retireCell:at::
- reviveCell:at::
- sizeToCells

- tagAt::
- intValueAt::
- floatValueAt::
- doubleValueAt::
- stringValueAt::

	- titleAt:: - stateAt::
Sorting	- autoSortCols - autoSortRows - autoSortSlots: - border:setAutoSortSlots: - border:setSlot:sortDirection: - border:setSlot:sortFunc: - border:setSlot:sortType: - border:setSlotSortVector:len: - border:slotSortDirection: - border:slotSortFunc: - border:slotSortType: - border:sortSlot: - colSortDirection: - colSortFunc:

- colSortType:
- colSortVectorLen:
- compareSlotFunc
- rowSortDirection:
- rowSortFunc:
- rowSortType:
- rowSortVectorLen:
- setAutoSortCols:
- setAutoSortRows:
- setCol:sortDirection:
- setCol:sortFunc:
- setCol:sortType:
- setColSortVector:len:
- setCompareSlotFunc:
- setRow:sortDirection:
- setRow:sortFunc:
- setRow:sortType:

- setRowSortVector:len:
- slotSortVector:len:
- sortCol:
- sortCols
- sortRow:
- sortRows
- sortSlots:

- slotsAreSorted:
- rowsAreSorted
- colsAreSorted
- border:slotIsSorted:
- colIsSorted:
- rowIsSorted:

- border:compareSlots::
- border:compareSlots::info:

	- compareCols:: - compareCols::info: - compareRows:: - compareRows::info: - sortInfoInit:border: - sortInfoDone:
Font	+ defaultFont - font - setFont:
Color	- backgroundColor - backgroundGray - color + defaultBackgroundColor + defaultHighlightBackgroundColor

+ defaultHighlightTextColor
+ defaultTextColor
- highlightBackgroundColor
- highlightBackgroundGray
- highlightTextColor
- highlightTextGray
- setBackgroundColor:
- setBackgroundGray:
- setColor:
- setHighlightBackgroundColor:
- setHighlightBackgroundGray:
- setHighlightTextColor:
- setHighlightTextGray:
- setTextColor:
- setTextGray:
- textColor
- textGray

Multicast	- makeCellsPerform: - makeCellsPerform:selectedOnly: - makeCellsPerform:with: - makeCellsPerform:with:with: - makeCellsPerform:with:selectedOnly: - makeCellsPerform:with:with:selectedOnly:
Finding Cells / Tags	- border:findSlotWithTag: - findCell:row:col: - findCellWithTag: - findCellWithTag:row:col: - findColWithTag: - findRowWithTag:
Save / Restore	- border:setSlotOrder: - border:setSlotOrderFromString:

- border:slotOrder:
- border:slotOrderAsString:size:canExpand:
- border:setSlotSizes:
- border:setSlotSizesFromString:
- border:slotSizes:
- border:slotSizesAsString:size:canExpand:
- colOrder:
- colOrderAsString:size:canExpand:
- colSizes:
- colSizesAsString:size:canExpand:
- rowOrder:
- rowOrderAsString:size:canExpand:
- rowSizes:
- rowSizesAsString:size:canExpand:
- setColOrder:
- setColOrderFromString:
- setColSizes:

	- setColSizesFromString: - setRowOrder: - setRowOrderFromString: - setRowSizes: - setRowSizesFromString:
Pasteboard and Services	- copy: - cut: - builtinCanWritePboardType: - builtinReadSelectionFromPasteboard: - builtinRegisterServicesTypes - builtinValidRequestorForSendType:andReturnType: - builtinWritePboard:type:toStream: - builtinWriteSelectionToPasteboard:types: - canWritePboardType: - readSelectionFromPasteboard: - registerServicesTypes

- validRequestorForSendType:andReturnType:
- writeNXAsciiPboardTypeToStream:
- writeNXTabularTextPboardTypeToStream:
- writePboard:type:toStream:
- writeSelectionToPasteboard:types:

Methods implemented by delegate - tableScroll:backgroundColorChangedTo:
- tableScroll:border:slotDraggedFrom:to:
- tableScroll:border:slotPrototype:
- tableScroll:border:slotResized:
- tableScroll:border:slotTitle:
- tableScroll:canWritePboardType:
- tableScroll:canWritePboardType:
- tableScroll:cellAt::
- tableScroll:tagAt::
- tableScroll:intValueAt::
- tableScroll:floatValueAt::

- tableScroll:doubleValueAt::
- tableScroll:stringValueAt::
- tableScroll:titleAt::
- tableScroll:stateAt::
- tableScroll:fontChangedFrom:to:
- tableScroll:highlightBackgroundColorChangedTo:
- tableScroll:highlightTextColorChangedTo:
- tableScroll:readSelectionFromPasteboard:
- tableScroll:readSelectionFromPasteboard:
- tableScroll:retireAt::
- tableScroll:retireCell:at::
- tableScroll:retireCell:at::
- tableScroll:reviveAt::
- tableScroll:reviveCell:at::
- tableScroll:reviveCell:at::
- tableScroll:textColorChangedTo:
- tableScroll:validRequestorForSendType:andReturnType:

- tableScroll:validRequestorForSendType:andReturnType:
- tableScroll:writePboard:type:toStream:
- tableScroll:writePboard:type:toStream:
- tableScroll:writeSelectionToPasteboard:types:
- tableScroll:writeSelectionToPasteboard:types:
- tableScrollBuffCount:
- tableScrollRegisterServicesTypes:
- tableScrollRegisterServicesTypes:

Class Methods

defaultBackgroundColor

+ (NXColor)**defaultBackgroundColor**

Returns NX_COLORLTGRAY. This is the default background color for new MiscTableScroll objects.

defaultFont

+ (Font*)**defaultFont**

Returns the user's preferred font at 12pt size. This is the default font for new MiscTableScroll objects.

defaultHighlightBackgroundColor

+ (NXColor)**defaultHighlightBackgroundColor**

Returns NX_COLORWHITE. This is the default highlight background color for new MiscTableScroll objects.

defaultHighlightTextColor

+ (NXColor)**defaultHighlightTextColor**

Returns NX_COLORBLACK. This is the default highlight text color for new MiscTableScroll objects.

defaultTextColor

+ (NXColor)**defaultTextColor**

Returns NX_COLORBLACK. This is the default text color for new MiscTableScroll objects.

Instance Methods

action

- (SEL)**action**

Returns the action associated with a single click

See also: -doubleAction, -setAction:, -setDoubleTarget:, -setTarget:, -target

addCol

- **addCol**

Appends a new column. See **Usage Tips** in the introduction for a more complete discussion. Equivalent to: `-addSlot:MISC_COL_BORDER`.

See also: -addRow, -addSlot:

addRow

- **addRow**

Appends a new row to the table. If you know how many rows you will need in advance, you should use -renewRows: instead; it will be faster. If you do not know the number of rows in advance, use this method. This method is faster than -insertColAt:. Internally, the table pre-allocates rows with a geometric growth pattern so there are only a logarithmic number of allocations. See **Usage Tips** in the introduction for a more complete discussion. Equivalent to: `-addSlot:MISC_ROW_BORDER`.

This method does no drawing, nor does it update the frames of the various subviews. (This enhances performance when adding hundreds or thousands of rows.) After you have finished adding rows, you must call sizeToCells so that the MiscTableScroll can update the frames of the various subviews.

See also: -addCol, -addSlot:, -renewRows:, -sizeToCells

addSlot:

- **addSlot:**(MiscBorderType)*b*

Appends a new row or column to the table. Appending rows is fast (geometric growth, logarithmic allocations, no shifting). Appending columns is slower (linear growth, linear allocations, lots of shifting). See **Usage Tips** in the introduction for a more complete discussion.

See also: -addCol, -addRow, -addSlot, -renewRows:

autoSortCols

- (BOOL)**autoSortCols**

Indicates whether columns will be automatically sorted when the user drags rows. Equivalent to `-autoSortSlots:MISC_COL_BORDER`.

autoSortRows

- (BOOL)**autoSortRows**

Indicates whether rows will be automatically sorted when the user drags columns. Equivalent to `-autoSortSlots:MISC_ROW_BORDER`.

autoSortSlots:

- (BOOL)**autoSortSlots:**(MiscBorderType)*b*

Indicates whether or not slots on the given border will be automatically sorted when the user drags (rearranges) slots on the other border.

backgroundColor

- (NXColor)**backgroundColor**

Returns the current background color for the MiscTableScroll object. The background color is used as the background color of unhighlighted cells in the table body as well as the "exposure color" for areas not covered by cells.

See also: -highlightBackgroundColor, -highlightTextColor, -setBackgroundColor:, -setHighlightBackgroundColor:, -setHighlightTextColor:, -setTextColor:, -textColor

backgroundGray

- (float)**backgroundGray**

This method converts the results of calling -backgroundColor to a grayscale value.

See also: **-backgroundColor**

border:compareSlots::

- (int)**border:**(MiscBorderType)*b*
  compareSlots:(int)*slot1* **:**(int)*slot2*

This method compares two slots. Returns a value less than zero if *slot1* should sort before *slot2*, zero if *slot1* should sort equally with *slot2*, or greater than zero if *slot1* should sort after *slot2*. It calls **-sortInfoInit:border:** to compute the sorting information, then calls **-border:compareSlots::info:**, and finally cleans up with **-slotInfoDone:**.

See also: **-border:compareSlots::info:, -border:slotIsSorted:, -border:sortSlot:**

border:compareSlots::info:

- (int)**border:**(MiscBorderType)*b*
  compareSlots:(int)*slot1* **:**(int)*slot2*

info:(MiscSlotSortInfo*)*sortInfo*

This method compares two slots, given a pointer to the precomputed sorting information. If you call this method, you are responsible for initializing *sortInfo* by calling **-sortInfoInit:border:**, and then releasing the resources by calling **-sortInfoDone:**.

See also: -border:compareSlots::, -sortInfoDone:, -sortInfoInit:border:

border:constrainMaxTotalSize:

- (void)**border:**(MiscBorderType)*b*
  constrainMaxTotalSize:(BOOL)*flag*

This method instructs the MiscTableScroll object whether or not to "tie" the maximum total size of the border, *b*, to the corresponding size of the ScrollView in which the table is displayed. When *flag* is YES, the maximum total size will be constrained to match the corresponding size of the scroll view, and it will be automatically adjusted whenever the size of the scroll view changes. When *flag* is NO, this will not happen. Maximum sizes are not constrained by default.

See also: **-border:constrainMinTotalSize:, -maxTotalSizeIsConstrained:, -border:setMaxTotalSize:**

border:constrainMinTotalSize:

- (void)**border:**(MiscBorderType)*b*
  constrainMinTotalSize:(BOOL)*flag*

This method instructs the MiscTableScroll object whether or not to "tie" the minimum total size of the border, *b*, to the corresponding size of the ScrollView in which the table is displayed. When *flag* is YES, the minimum total size will be constrained to match the corresponding size of the scroll view, and it will be automatically adjusted whenever the size of the scroll view changes. When *flag* is NO, this will not happen. Minimum sizes are constrained by default. <???FIXME: This default behavior is obnoxious!?>

See also: **-border:constrainMaxTotalSize:, -minTotalSizeIsConstrained:, -border:setMinTotalSize:**

border:deleteSlotAt:

- **border:**(MiscBorderType)*b*

deleteSlotAt:(int)*pos*

Deletes a single row or column. All cells and other resources for the slot are deallocated immediately. If you are just emptying the table so that you can refill it with new data, use -renewRows: or -empty instead. See **Usage Tips** in the introduction for a more complete discussion.

See also: -deleteColAt:, -deleteRowAt:, -empty, -emptyAndFreeCells, -renewRows:

border:drawSlot:

- **border:**(MiscBorderType)*b*
  drawSlot:(int)*n*

Draws a single row or column. This method locks focus on the view if needed. You should never need to call this method.

See also: -drawCellAt::, -drawCol:, -drawRow:

border:drawSlotTitle:

- **border:**(MiscBorderType)*b*
  drawSlotTitle:(int)*n*

Draws the title cell for a single row or column. This method locks focus on the view if needed. You should never need to call this method.

See also: -drawColTitle:, -drawRowTitle:

border:findSlotWithTag:

- (int)**border:**(MiscBorderType)*b*
  findSlotWithTag:(int)*x*

Returns the index of the first slot whose tag is *x*, or -1 if no match was found.

See also: -findRowWithTag:, -findColWithTag:, -findCellWithTag:row:col:

border:getDelegateSlotPrototype:

- **border:**(MiscBorderType)*b*
  getDelegateSlotPrototype:(int)*s*

Internal method used whenever the MiscTableScroll object needs to get the cell prototype for a slot which was marked with a "Call-Back" cell type.

border:getDelegateSlotTitle:

- (char const*)**border:**(MiscBorderType)*b*
  getDelegateSlotTitle:(int)*slot*

Internal method used whenever the MiscTableScroll object needs to get a title for a slot, and the title-style for the border is "Delegate".

border:insertSlotAt:

- **border:**(MiscBorderType)*b*

insertSlotAt:(int)*pos*

Inserts a single row or column at the indicated position. Position is a zero-based index. <???FIXME: What is the visual position of slots created this way?> This method performs linear allocation, and is slower than the corresponding -addRow method when adding new rows to a table.

See also: -addCol, -addRow, -addSlot:, -insertColAt:, -insertRowAt:

border:moveSlotFrom:to:

- **border:**(MiscBorderType)*b*
  moveSlotFrom:(int)*from_pos*
  to:(int)*to_pos*

This is equivalent to the user dragging a slot from *from_pos* to *to_pos*. Both *from_pos* and *to_pos* are zero-based indexes into the current visual ordering of the slots.

See also: -moveColFrom:to:, -moveRowFrom:to:

border:physicalToVisual:

- (void)**border:**(MiscBorderType)*b*
  physicalToVisual:(MiscIntList*)*list*

This method converts all the values in *list*. It assumes that all the values in *list* represent physical (original) slot indexes. It converts those values to their corresponding visual (current) values via -border:slotPosition:.

See also: -border:slotPosition:, -border:visualToPhysical:

border:selectSlot:

- (void)**border:**(MiscBorderType)*b*
  selectSlot:(MiscCoord_P)*slot*

Selects the indicated slot. Does not clear the previous selection, so it is added to the existing selection.

See also:

border:selectSlots:

- (void)**border:**(MiscBorderType)*b*
  selectSlots:(MiscIntList*)*slots*

Clears the selection, then selects each slot in *slots*, which should be a list of slot indexes.

See also: **-border:selectedSlots:**

border:selectTags:

- (void)**border:**(MiscBorderType)*b*
  selectTags:(MiscIntList*)*tags*

Clears the selection, then selects all slots whose tag value can be found in *tags*. This method is useful in conjunction with -border:selectedTags: to save and restore the user's selection when you have tag values that uniquely identify the slots.

See also: **-border:selectedTags:**

border:selectedSlots:

- (void)**border:**(MiscBorderType)*b*
  selectedSlots:(MiscIntList*)*slots*

Fills the list *slots* with the indexes of all currently selected slots. You are responsible for allocating the *slots* list, and you are responsible for freeing it when you are finished with it.

See also: **-border:selectSlots:**

border:selectedTags:

- (void)**border:**(MiscBorderType)*b*
  selectedTags:(MiscIntList*)*tags*

Fills the list *tags* with the tags of all currently selected slots. You are responsible for allocating and freeing the

tags list. This method is useful in conjunction with -border:selectTags: to save and restore the user's selection when you have tags that uniquely identify the slots.

See also: -border:selectTags:

border:setAutoSortSlots:

- (void)**border:**(MiscBorderType)*b*
  setAutoSortSlots:(BOOL)*flag*

Instructs the MiscTableScroll object whether or not to automatically sort the slots in border *b*, when the user drags a slot from the other border. For example, when you tell the MiscTableScroll object to auto-sort rows, the rows will be automatically sorted every time the user drags a column to a new position. AutoSort is off by default.

See also:

border:setCursor:

- (void)**border:**(MiscBorderType)*b*
  setCursor:(MiscCoord_P)*slot*

Sets the keyboard cursor to *slot*.

See also:

border:setDraggableSlots:

- (void)**border:**(MiscBorderType)*b*
  setDraggableSlots:(BOOL)*flag*

Enables or disables reordering of the slots. To let the user drag slots, the titles must be displayed, and the slots must be draggable.

See also: -border:setModifierDragSlots:, -border:setSizeableSlots:, -border:setTitlesOn:, -setDraggableCols:, -setDraggableRows:

border:setMaxTotalSize:

- (void)**border:**(MiscBorderType)*b*
  setMaxTotalSize:(NXCoord)*size*

Sets the maximum total size for border *b*. Whenever the normal total size of the border would be more than the maximum total size, all autosize slots shrink proportionately until either the total size meets the global maximum limit, or all autosize slots have shrunk to their minimum sizes. Local, slot-specific minimum sizes take precedence over the global maximum size.

See also: -border:constrainMaxSize:, -border:setMinTotalSize:, -maxTotalSize:

border:setMinTotalSize:

- (void)**border:**(MiscBorderType)*b*
  setMinTotalSize:(NXCoord)*size*

Sets the minimum total size for border *b*. Whenever the normal total size of the border would be less than the

minimum total size, all autosize slots grow proportionately until either the total size meets the global minimum limit, or all autosize slots have grown to their maximum sizes. Local, slot-specific maximum sizes take precedence over the global minimum size.

See also: -border:constrainMinSize:, -border:setMaxTotalSize:, -minTotalSize:

border:setModifierDragSlots:

- (void)**border:**(MiscBorderType)*b*
  setModifierDragSlots:(BOOL)*flag*

This option controls whether a unmodified mouse-down initiates selection, or slot-dragging. When *flag* is YES, an unmodified mouse-down initiates selection, and the user must hold down the command-key to drag a slot. When *flag* is NO, an unmodified mouse-down initiates dragging, and the user must hold down the command-key to select a slot. By default, columns are dragged with an unmodified mouse-down and selected when the command-key modifier is used. By default, rows behave the other way; an unmodified mouse-down initiates selection, and a command-key modifier must be used to initiate dragging.

See also: **-setModifierDragCols:, -setModifierDragRows:**

border:setSizeableSlots:

- (void)**border:**(MiscBorderType)*b*
  setSizeableSlots:(BOOL)*flag*

Enables or disables user-sizing of the slots. Many conditions must be met to enable the user to resize a particular slot. See **Slot Sizing** in the introduction for details.

See also: **-setSizeableCols:, -setSizeableRows:**

border:setSlot:autosize:

- (void)**border:**(MiscBorderType)*b*
  setSlot:(int)*n*
  autosize:(BOOL)*flag*

Enables or disables "autosizing" for a particular slot. When YES, the slot will be adjusted proportionately with all other "autosize" slots in the border to meet global minimum or maximum size restrictions for the border as a whole. Currently, this only has effect for columns in narrow tables displayed in wide ScrollViews. See **Slot Sizing** in the introduction for more details.

See also: -setCol:autosize:, -setRow:autosize:

border:setSlot:cellPrototype:

- (void)**border:**(MiscBorderType)*b*
  setSlot:(int)*n*
  cellPrototype:*cell*

Set the cell prototype for a slot. Currently, only column cell prototypes are used. When new rows are allocated for the table, the cell prototype from each column is sent a **-copy** message. The newly created cell is placed into the newly created row. Thus all prototype cells must implement the **-copy** message appropriately. (They must make copies of all components that are freed when the **-free** message is sent to the cell). The MiscTableScroll object takes "ownership" of the prototype cell that is passed in. The caller must not free the

prototype cell. The MiscTableScroll object will free the prototype cell when it is finished with it.

See also: -border:setSlot:cellType:, -setCol:cellPrototype:, -setRow:cellPrototype:,

border:setSlot:cellType:

- (void)**border:**(MiscBorderType)*b*
  setSlot:(int)*n*
  cellType:(MiscTableCellStyle)*t*

Sets the type of cell that will be used for a particular slot. Currently, only column cell types have any effect; row cell types are ignored. The cell type, *t*, can be any of the following (declared in <MiscTableTypes.h>):

```
MISC_TABLE_CELL_TEXT
MISC_TABLE_CELL_ICON
MISC_TABLE_CELL_BUTTON
MISC_TABLE_CELL_CALLBACK
```

When this method is called, the MiscTableScroll object will create a prototype cell for the indicated slot of the

indicated type. `MISC_TABLE_CELL_TEXT` creates a text-cell; `MISC_TABLE_CELL_ICON` creates an icon-cell; `MISC_TABLE_CELL_BUTTON` creates a button cell; and `MISC_TABLE_CELL_CALLBACK` instructs the MiscTableScroll object to ask the delegate for the for the prototype cell. <???FIXME: What is the interaction with setting an explicit prototype cell? What message is sent to the delegate to get the prototype? When is the message sent? Does it ask for the cell itself, or just a prototype? Is it sent to the delegate, or the data-delegate?>

See also: -border:setSlot:cellPrototype:, -setCol:cellType:, -setRow:cellType:

border:setSlot:dataSize:

- (void)**border:**(MiscBorderType)*b*
  setSlot:(int)*n*
  dataSize:(NXCoord)*size*

No description.

See also:

border:setSlot:expandsToData:

- (void)**border:**(MiscBorderType)*b*
  setSlot:(int)*n*
  expandsToData:(BOOL)*flag*

Sets the "expands-to-data" flag for slot *n*. When *flag* is YES, the slot will expand as needed to accommodate the largest entry in that slot. Only the "adjusted-size" of a slot is affected, and it can only be increased, never decreased. Expansions resulting from expands-to-data will not exceed the maximum size for the slot. When *flag* is NO, no such adjustment will be made. See **Slot Sizing** in the introduction for more details. <???FIXME: How is the largest size determined? When does it get adjusted?>

See also: -setCol:expandsToData:, -setRow:expandsToData:

border:setSlot:maxSize:

- (void)**border:**(MiscBorderType)*b*
  setSlot:(int)*n*

maxSize:(NXCoord)*size*

Sets the maximum size for slot *n*. The *size* argument is in units of screen pixels. <???FIXME: Check this.?> This limit affects user resizing of slots; the "target" size for a slot, and automatic slot expansions performed to accommodate the "expands-to-data" and "autosize" flags. See **Slot Sizing** in the introduction for more details.

See also: -setCol:maxSize:, -setRow:maxSize:

border:setSlot:minSize:

- (void)**border:**(MiscBorderType)*b*
  setSlot:(int)*n*
  minSize:(NXCoord)*size*

Sets the minimum size for slot *n*. The *size* argument is in units of screen pixels. <???FIXME: Check this.?> This limit affects user resizing of slots, and acts as a limit on the "target size" for a slot. See **Slot Sizing** in the introduction for more details.

See also: -setCol:minSize:, -setRow:minSize:

border:setSlot:size:

- (void)**border:**(MiscBorderType)*b*
  setSlot:(int)*n*
  size:(NXCoord)*size*

Sets the "target" size for slot *n*. The *size* argument is in units of screen pixels. <???FIXME: Check this.?> See **Slot Sizing** in the introduction for more details.

See also: -setCol:size:, -setRow:size:

border:setSlot:sizeable:

- (void)**border:**(MiscBorderType)*b*
  setSlot:(int)*n*
  sizeable:(BOOL)*flag*

Sets the user-sizeable flag for slot *n*. When *flag* is YES, the user will be able to resize the slot. When *flag* is NO, the user will not be able to resize the slot. There are many conditions which must be met for a user to be able to resize a slot. See **Slot Sizing** in the introduction for more details.

See also: -setCol:sizeable:, -setRow:sizeable:

border:setSlot:sortDirection:

- (void)**border:**(MiscBorderType)*b*
  setSlot:(int)*n*
  sortDirection:(MiscSortDirection)*x*

Indicates whether slot *n* should be sorted in ascending or descending order. *X* must be one of the following two values from <MiscTableTypes.h>:

```
MISC_SORT_ASCENDING
MISC_SORT_DESCENDING
```

All other values are ignored.

See also: -border:setSlot:sortType:, -border:setSlot:sortFunc:, -border:slotSortDirection:

border:setSlot:sortFunc:

- (void)**border:**(MiscBorderType)*b*
  setSlot:(int)*n*
  sortFunc:(MiscCompareEntryFunc)*func*

Makes *func* the cell-to-cell comparison routine for the cells in slot *n*. The function, *func*, must match the following prototype from <MiscTableTypes.h>.

```
typedef int (*MiscCompareEntryFunc)( int r1, int c1, int r2, int c2, MiscSlotSortInfo* info );
```

The function is given the coordinates of the two cells, and a pointer to a structure containing additional sorting information. The function should return an integer that is: (a) less than zero if the cell at (r1,c1) should sort before the cell at (r2,c2), (b) equal to zero if the two cells should sort equally, or (c) greater than zero if the cell at

(r1,c1) should sort after the cell at (r2,c2).

The sort direction (ascending or descending) is applied to the value returned by the cell-to-cell comparison function by the slot-to-slot comparison function. So if you supply a custom cell-to-cell comparison function you should ignore the sort direction for that slot. You should always return the "ascending" sort-order value.

Use this method when you need to perform custom sorting that the builtin sort-types cannot accommodate.

See also: -border:setSlot:sortDirection:, -border:setSlot:sortType:, -border:slotSortFunc:

border:setSlot:sortType:

- (void)**border:**(MiscBorderType)*b*
  setSlot:(int)*n*
  sortType:(MiscSortType)*x*

Sets the type of sorting to be used by the builtin cell-to-cell comparison function for cells in slot *n*. The sort type, *x*, must be one of the following values from <MiscTableTypes.h>:

```
MISC_SORT_STRING_CASE_INSENSITIVE
MISC_SORT_STRING_CASE_SENSITIVE
MISC_SORT_INT
MISC_SORT_UNSIGNED_INT
MISC_SORT_TAG
MISC_SORT_UNSIGNED_TAG
MISC_SORT_FLOAT
MISC_SORT_DOUBLE
MISC_SORT_SKIP
MISC_SORT_TITLE_CASE_INSENSITIVE
MISC_SORT_TITLE_CASE_SENSITIVE
MISC_SORT_STATE
MISC_SORT_UNSIGNED_STATE
```

All other values are ignored. Each of the types is described below.

```
MISC_SORT_STRING_CASE_INSENSITIVE
MISC_SORT_STRING_CASE_SENSITIVE
```

The cells are compared as strings. The string values are retrieved using the **-stringValueAt::**. `MISC_SORT_STRING_CASE_INSENSITIVE` is the default sort-type.

`MISC_SORT_INT`
`MISC_SORT_UNSIGNED_INT`

The cells are compared as integers. The integer values are retrieved using the **-intValueAt::** message.

`MISC_SORT_TAG`
`MISC_SORT_UNSIGNED_TAG`

The cells are compared as integers. The integer values are retrieved using the **-tagAt::** message. This feature is useful for sorting that is handled "behind-the-scenes". For example, if the slot holds date information, you can put the UNIX time_t date value into the cell tag, and format the string value any way you wish. The slot will sort correctly regardless of the display format. It is also useful for slots that hold keywords from an ordered set of values, like the following enumeration:

```
enum Severity { Notice, Warning, Error, Fatal };
```

Sorting these alphabetically does not make sense, but if you put the enum value into the tag of the cell, you can sort them correctly. This sort type also makes it possible to sort slots that hold icons.

`MISC_SORT_FLOAT`

The cells are compared as single-precision floating point numbers. The values are retrieved using the **-floatValueAt::** message.

`MISC_SORT_DOUBLE`

The cells are compared as double-precision floating point numbers. The values are retrieved using the **-doubleValueAt::** message.

`MISC_SORT_SKIP`

The cells are not compared. All cells in slots with the `MISC_SORT_SKIP` sort-type are considered equal. This can be used for slots that should not affect the sorting.

`MISC_SORT_TITLE_CASE_INSENSITIVE`
`MISC_SORT_TITLE_CASE_SENSITIVE`

The cells are compared as strings. The string values are retrieved using the **-titleAt::**. This is provided to support ButtonCells.

`MISC_SORT_STATE`
`MISC_SORT_UNSIGNED_STATE`

The cells are compared as integers. The integer values are retrieved using the **-stateAt::** message. This is provided to support ButtonCells.

See also: -border:setSlot:sortDirection:, -border:setSlot:sortFunc:, -border:slotSortType:

border:setSlot:tag:

- (void)**border:**(MiscBorderType)*b*
  setSlot:(int)*n*
  tag:(int)*tag*

Sets the tag for slot *n* to *tag*.

See also: -setCol:tag:, -setRow:tag:

border:setSlot:title:

- (void)**border:**(MiscBorderType)*b*
  setSlot:(int)*n*
  title:(char const*)*title*

Sets the title for slot *n* to *title*. This method only works for borders with custom titles (that is, `-border:`*`b`* `setTitleMode:MISC_CUSTOM_TITLE`). If the border does not have custom titles, the request is silently ignored. If the border does have custom titles, the MiscTableScroll object will make a private copy of the *title* string, and update the display if autodisplay is on.

See also: -border:setTitleMode:, -setCol:title:, -setRow:title:

border:setSlotOrder:

- **border:**(MiscBorderType)*b*
  setSlotOrder:(MiscIntList*)*list*

Rearranges the slots to match the order specified by *list*. The list is organized in the original (physical) order of

the slots. Each value in the list is the new (visual) position for the corresponding slot. In other words, *list* is a physical to visual mapping. This is useful for restoring the user's slot-order preference.

See also: -border:setSlotOrderFromString:, -border:setSlotSizes:, -border:slotOrder:

border:setSlotOrderFromString:

- **border:**(MiscBorderType)*b*
  setSlotOrderFromString:(char const*)*s*

This is a convenience method which invokes `-border:setSlotOrder:` using an `MiscIntList` constructed from *s*.

See also: -border:setSlotOrder:, -border:setSlotSizes:, -border:slotOrder:, -initFromString: (MiscIntList), -readFromString: (MiscIntList)

border:setSlotSizes:

- **border:**(MiscBorderType)*b*

setSlotSizes:(MiscIntList*)*list*

Sets the sizes of all slots to the values in *list*. List is organized in original (physical) slot order. Each value is the size of the corresponding slot. This method is useful for restoring the user's slot size preferences.

See also: -border:setSlotOrder:, -border:setSlotSizesFromString:, -border:slotSizes:

border:setSlotSizesFromString:

- **border:**(MiscBorderType)*b*
  setSlotSizesFromString:(char const*)*s*

This is a convenience method which invokes `-border:setSlotSizes:` using an `MiscIntList` constructed from *s*.

See also: -border:setSlotOrder:, -border:setSlotSizes:, -border:slotSizes:, -initFromString: (MiscIntList), -readFromString: (MiscIntList)

border:setSlotSortVector:len:

- (void)**border:**(MiscBorderType)*b*
  setSlotSortVector:(int const*)*v*
  len:(int)*n*

Sets the order in which slots are considered when sorting. Each value in *v* is the original (physical) position of a slot. The slots will be compared in the order that they appear in *v*. Negative values reverse the sort direction (ascending / descending) of the corresponding slot. The current visual slot order is used by default; use this method if that is not what you want.

See also: -slotSortVector:len:

border:setTitleMode:

- (void)**border:**(MiscBorderType)*b*
  setTitleMode:(MiscTableTitleMode)*x*

Sets the title-mode for a border. The title-mode, *x*, can be any of the following (declared in

<MiscTableTypes.h>):

```
MISC_NO_TITLE,              // No titles on row/col cells.
MISC_NUMBER_TITLE,          // Titles are sequential numbers.
MISC_ALPHA_TITLE,           // Titles are sequential alphabetics...
MISC_CUSTOM_TITLE,          // Titles are user-supplied strings...
MISC_DELEGATE_TITLE         // Ask the delegate for titles.
```

See also: -setColTitleMode:, -setRowTitleMode:, -tableScroll:border:slotTitle: (delegate method)

border:setTitlesOn:

- (BOOL)**border:**(MiscBorderType)*b*
  setTitlesOn:(BOOL)*on_off*

Determines whether titles will be displayed or not. When *on_off* is YES, the titles will be displayed. When *on_off* is NO, the titles will not be displayed. The titles must be displayed to let the user resize and drag slots. See **Slot Sizing** in the introduction for more details.

See also: **-setColTitlesOn: -setRowTitlesOn:**

border:setUniformSizeSlots:

- (void)**border:**(MiscBorderType)*b*
  setUniformSizeSlots:(NXCoord)*uniform_size*

Sets or clears the uniform-size for a border. If *uniform_size* is zero, then each slot on that border will be able to have individually varying sizes. If *uniform_size* is non-zero, then every slot on that border will have the size, *uniform_size*. When the slots on a border have a uniform size, the user will not be able to resize the slots. See **Slot Sizing** in the introduction for more details.

See also: **-setUniformSizeCols:, -setUniformSizeRows:**

border:slotAdjustedSize:

- (NXCoord)**border:**(MiscBorderType)*b*

slotAdjustedSize:(int)*slot*

Returns the current display size of *slot*.

See also:

border:slotAtPosition:

- (int)**border:**(MiscBorderType)*b*
  slotAtPosition:(int)*pos*

Returns the original physical position of the slot in visual position *pos*. This is the visual-to-physical conversion routine.

See also: -border:moveSlotFrom:to:, -border:slotPosition:, -colAtPosition:, -rowAtPosition:

border:slotCellPrototype:

- **border:**(MiscBorderType)*b*
  slotCellPrototype:(int)*slot*

Returns the cell prototype for *slot*.

See also: **-border:setSlot:cellPrototype:, -border:setSlot:cellType:, -colCellPrototype:, -rowCellPrototype:**

border:slotCellType:

- (MiscTableCellStyle)**border:**(MiscBorderType)*b*
  slotCellType:(int)*slot*

Returns the cell type for *slot*.

See also: **-border:setSlot:cellPrototype:, -border:setSlot:cellType:, -colCellType:, -rowCellType:**

border:slotDataSize:

- (NXCoord)**border:**(MiscBorderType)*b*
  slotDataSize:(int)*slot*

No description.

See also:

border:slotDraggedFrom:to:

- (void)**border:**(MiscBorderType)*b*
  slotDraggedFrom:(int)*from_pos*
  to:(int)*to_pos*

Internal method, invoked whenever the user drags a slot to a new position.

border:slotExpandsToData:

- (BOOL)**border:**(MiscBorderType)*b*

slotExpandsToData:(int)*slot*

Returns the state of the expands-to-data flag for *slot*. See **Slot Sizing** in the introduction for more details.

See also: **-border:setSlot:expandsToData:, -colExpandsToData:, -rowExpandsToData:**

border:slotIsAutosize:

- (BOOL)**border:**(MiscBorderType)*b*
  slotIsAutosize:(int)*slot*

Returns the state of the autosize flag for *slot*. See **Slot Sizing** in the introduction for more details.

See also: **-border:setSlot:autosize:, -colIsAutosize:, -rowIsAutosize:**

border:slotIsSelected:

- (BOOL)**border:**(MiscBorderType)*b*

slotIsSelected:(MiscCoord_P)*slot*

Returns YES if *slot* is selected, else NO.

See also:

border:slotIsSizeable:

- (BOOL)**border:**(MiscBorderType)*b*
  slotIsSizeable:(int)*slot*

Returns the state of the user-sizeable flag for *slot*. See **Slot Sizing** in the introduction for more details.

See also: -border:setSlot:sizeable:, -colIsSizeable:, -rowIsSizeable:

border:slotIsSorted:

- (BOOL)**border:**(MiscBorderType)*b*

slotIsSorted:(int)*slot*

This method compares *slot* with its neighbors. It returns YES if these slots are sorted relative to each other. It returns NO if any of these slots are out of order with respect to the others. This method can be useful for determining whether or not the table must be resorted when you are changing values in the table.

See also: -border:sortSlot:, -slotsAreSorted:

border:slotMaxSize:

- (NXCoord)**border:**(MiscBorderType)*b*
  slotMaxSize:(int)*slot*

Returns the maximum size for *slot*. See **Slot Sizing** in the introduction for more details.

See also: -border:setSlot:maxSize:, -colMaxSize:, -rowMaxSize:

border:slotMinSize:

- (NXCoord)**border:**(MiscBorderType)*b*
  slotMinSize:(int)*slot*

Returns the minimum size for *slot*. See **Slot Sizing** in the introduction for more details.

See also: -border:setSlot:minSize:, -colMinSize:, -rowMinSize:

border:slotOrder:

- **border:**(MiscBorderType)*b*
  slotOrder:(MiscIntList*)*list*

Puts the current slot order into *list*. The list is organized in original (physical) slot order. Each value in the list is the current (visual) position of the corresponding slot. In other words, *list*, is filled with the physical to visual mapping. This method is useful for saving the user's slot order preference. You are responsible for allocating and freeing *list*.

See also: -border:setSlotOrder:, -border:setSlotSizes:, -border:slotOrderAsString:size:canExpand:

border:slotOrderAsString:size:canExpand:

- (char*)**border:**(MiscBorderType)*b*
  slotOrderAsString:(char*)*buff*
  size:(int)*buff_size*
  canExpand:(BOOL)*canExpand*

Calls -border:slotOrder: with a temporary MiscIntList object, then sends it the -writeToString:size:canExpand: message, with the values provided by the caller.

See also: -border:setSlotOrder:, -border:slotOrder:, -writeToString:size:canExpand: (MiscIntList)

border:slotPosition:

- (int)**border:**(MiscBorderType)*b*
  slotPosition:(int)*slot*

Returns the current visual position of the slot whose original physical position was *slot*. This is the physical-to-visual conversion routine.

See also: -border:moveSlotFrom:to:, -border:slotAtPosition:, -colPosition:, -rowPosition:

border:slotResized:

- (void)**border:**(MiscBorderType)*b*
  slotResized:(int)*n*

Internal method called whenever the user resizes a slot.

border:slotSize:

- (NXCoord)**border:**(MiscBorderType)*b*
  slotSize:(int)*slot*

Returns the target size for *slot*. See **Slot Sizing** in the introduction for more details.

See also: **-border:setSlot:size:, -setCol:size:, -setRow:size:**

border:slotSizes:

- **border:**(MiscBorderType)*b*
  slotSizes:(MiscIntList*)*list*

Fills *list* with the sizes of all the slots. The list is organized in original (physical) slot order. The values are the sizes of the correspond slot. This method is useful for saving the user's slot size preferences.

See also: **-border:setSlotSizes:, -border:slotOrder:, -border:slotSizesAsString:size:canExpand:**

border:slotSizesAsString:size:canExpand:

- (char*)**border:**(MiscBorderType)*b*
  slotSizesAsString:(char*)*buff*
  size:(int)*buff_size*

canExpand:(BOOL)*canExpand*

Calls -border:slotSizes: with a temporary MiscIntList object, then sends it the -writeToString:size:canExpand: message, with the values provided by the caller.

See also: -border:slotSizes:, -writeToString:size:canExpand: (MiscIntList)

border:slotSortDirection:

- (MiscSortDirection)**border:**(MiscBorderType)*b*
  slotSortDirection:(int)*n*

Returns the sort direction (ascending or descending) for slot *n*.

See also: -border:setSlot:sortDirection:

border:slotSortFunc:

- (MiscCompareEntryFunc)**border:**(MiscBorderType)*b*
  slotSortFunc:(int)*n*

Returns the custom sort function for slot *n*, if any, otherwise it returns 0.

See also: **-border:setSlot:sortFunc:**

border:slotSortType:

- (MiscSortType)**border:**(MiscBorderType)*b*
  slotSortType:(int)*n*

Returns the sort type of slot *n*.

See also: **-border:setSlot:sortType:**

border:slotTag:

- (int)**border:**(MiscBorderType)*b*
  slotTag:(int)*slot*

Returns the tag value associated with *slot*.

See also: **-border:setSlot:tag:, -colTag:, -rowTag:**

border:slotTitle:

- (char const*)**border:**(MiscBorderType)*b*
  slotTitle:(int)*slot*

Returns the title for *slot*.

See also: **-border:setSlot:title:, -colTitle:, -rowTitle:**

border:sortSlot:

- (BOOL)**border:**(MiscBorderType)*b*
  sortSlot:(int)*slot*

Re-sorts a single slot. This method can be used to restore the sort order after a single slot has been added or changed in such a way that it might not be in the correct sort position. The results are unpredictable if the other slots are not already sorted.

See also: -sortSlots:, -sortCol:, -sortRow:

border:visualToPhysical:

- (void)**border:**(MiscBorderType)*b*
  visualToPhysical:(MiscIntList*)*list*

Converts a list of current (visual) slot positions to their original (physical) positions.

See also: -border:slotAtPosition:, -border:slotPosition:

borderClearCursor:

- (void)**borderClearCursor:**(MiscBorderType)*b*

Turns off the keyboard cursor.

See also:

borderClearSelection:

- (void)**borderClearSelection:**(MiscBorderType)*b*

Unselects all slots that were selected.

See also:

borderCursor:

- (MiscCoord_P)**borderCursor:**(MiscBorderType)*b*

Returns the index of the slot that the keyboard cursor is currently on, or -1 if the keyboard cursor is not on any slot.

See also:

borderHasMultipleSelection:

- (BOOL)**borderHasMultipleSelection:**(MiscBorderType)*b*

Returns YES if more than one slot is selected, otherwise NO.

See also:

borderHasSelection:

- (BOOL)**borderHasSelection:**(MiscBorderType)*b*

Returns YES if at least one slot is selected, otherwise NO.

See also:

borderHasValidCursor:

- (BOOL)**borderHasValidCursor:**(MiscBorderType)*b*

Returns YES if the keyboard cursor is positioned on a valid slot, otherwise NO.

See also:

borderNumSelectedSlots:

- (unsigned int)**borderNumSelectedSlots:**(MiscBorderType)*b*

Returns the number of slots that are selected.

See also:

borderSelectAll:

- (void)**borderSelectAll:**(MiscBorderType)*b*

Selects all the slots. Does not send the action to the target.

See also: -selectAll:

borderSelectedSlot:

- (MiscCoord_P)**borderSelectedSlot:**(MiscBorderType)*b*

Returns the index of the currently selected slot, or -1 if no slots are selected.

See also:

borderTitleMode:

- (MiscTableTitleMode)**borderTitleMode:**(MiscBorderType)*b*

Returns the title-mode for *slot*.

See also: -border:setSlot:titleMode:, -colTitleMode:, -rowTitleMode:

borderTitlesOn:

- (BOOL)**borderTitlesOn:**(MiscBorderType)*b*

Indicates whether or not the titles for border *b* are displayed.

See also: -border:setTitlesOn:, -colTitlesOn:, -rowTitlesOn:

buffCount

- (int)**buffCount**

This method is only meaningful for lazy tables. If the delegate or dataDelegate provide multiple buffers for

responding to **-tableScroll:cellAt::**, they are encouraged to respond to **-tableScroll:buffCount:** with the number of buffers that they provide. If the delegate and dataDelegate do not respond, a default value of one (1) is returned, which indicates that all values from a call to **-cellAt::** must be copied before making a second call to **-cellAt::**. This method is called internally during **-sortInfoInit:border:** to determine whether copying must be performed during sorting.

See also: -cellAt::, -isLazy, -setLazy:, -tableScroll:buffCount:, -tableScroll:cellAt::

builtinCanWritePboardType:

- (BOOL)**builtinCanWritePboardType:**(NXAtom)*type*

The builtin method for determining which data types can be placed on the pasteboard. This method returns YES for `NXAsciiPboardType` and `NXTabularTextPboardType`. Override this method in your subclass if you will provide additional pasteboard datatypes. This method is called from -canWritePboardType:.

See also: -canWritePboardType:

builtinReadSelectionFromPasteboard:

- **builtinReadSelectionFromPasteboard:***pboard*

This method does nothing. The current implementation of MiscTableScroll never reads anything from the pasteboard. Override this method in your subclass if you want to read data from the pasteboard. This method is called from -readSelectionFromPasteboard:.

See also: -readSelectionFromPasteboard:

builtinRegisterServicesTypes

- (void)**builtinRegisterServicesTypes**

This method sends `-registerServicesMenuSendTypes:andReturnTypes:` to NXApp. It registers `NXTabularTextPboardType` and `NXAsciiPboardType` as send types, and registers nothing as return types. Override this method in your subclass if you want to send or return different data types. Called from `-registerServicesTypes`.

See also: **-registerServicesTypes, -registerServicesMenuSendTypes:andReturnTypes: (Application)**

builtinValidRequestorForSendType:andReturnType:

- **builtinValidRequestorForSendType:**(NXAtom)*t_write*
  andReturnType:(NXAtom)*t_read*

This method returns **self** if *t_write* is either `NXTabularTextPboardType` or `NXAsciiPboardType`, and *t_read* is 0, otherwise it returns 0. Override this method if your subclass can handle different combinations. Called from `-validRequestorForSendType:andReturnType:`.

See also: **-validRequestorForSendType:andReturnType:**

builtinWritePboard:type:toStream:

- (void)**builtinWritePboard:***pb*
  type:(NXAtom)*type*

toStream:(NXStream*)*stream*

If *type* is `NXAsciiPboardType`, then -`writeNXAsciiPboardTypeToStream:` is called, else if *type* is `NXTabularTextPboardType`, then -`writeNXTabularTextPboardTypeToStream:` is called. Otherwise it does nothing. Override this method in your subclass if you can write additional datatypes to the pasteboard. Called from -writePboard:type:toStream:.

See also: -writePboard:type:toStream:

builtinWriteSelectionToPasteboard:types:

- (BOOL)**builtinWriteSelectionToPasteboard:***pboard*
  types:(NXAtom*)*types*

Writes all of the types that can be written to the pasteboard. Each entry in *types* is tested with `-canWritePboardType:`. If the result is YES, and there is a selection, it is passed on to `-writePboard:type:toStream:`. The data is written immediately, the MiscTableScroll object does not register any object as the data-provider. Override this method in your subclass if you need different behavior. Called

from `-writeSelectionToPasteboard:types:`.

See also: -canWritePboardType:, -writePboard:type:toStream:, -writeSelectionToPasteboard:types:

canWritePboardType:

- (BOOL)**canWritePboardType:**(NXAtom)*type*

Responds to queries from `-builtinWriteSelectionToPasteboard:types:`. First it gives the delegate an opportunity to answer via `-tableScroll:canWritePboardType:`. If the delegate does not respond to that message, it gives the dataDelegate an opportunity to answer the same message. If neither object responds, the builtin implementation, `-builtinCanWritePboardType:` is called. Called from `-builtinWriteSelectionToPasteboard:types:`.

See also: -builtinCanWritePboardType:, -builtinWriteSelectionToPasteboard:types:, -tableScroll:canWritePboardType: (delegate method)

cellAt::

- **cellAt:**(int)*row* **:**(int)*col*

Returns a pointer to the cell located at (*row*,*col*).

See also:

cellIsSelected::

- (BOOL)**cellIsSelected:**(MiscCoord_P)*row* **:**(MiscCoord_P)*col*

Returns YES if the cell at *row*, *col* is selected.

See also:

NS_DEV_DOCFOR:objc_method:[Text-changeFont:];,changeFont:

- **changeFont:***sender*

Changes the font of the MiscTableScroll object as well all cells which inherit it. The FontManager sends the -**changeFont:** message whenever the user changes the font using either the FontPanel or the Font Menu. *sender* must respond to the -**convertFont:** message and return a Font which is passed to **-setFont**:. This method sends **-tableScroll:changeFont:to:** and **-tableScroll:fontChangedFrom:to:** messages to the delegate. Returns **self**.

See also: - **setFont:, ± tableScroll:changeFont:to:, ± tableScroll:fontChangedFrom:to:**

clearColSelection

- (void)**clearColSelection**

Equivalent to: `-borderClearSelection:MISC_COL_BORDER`.

See also: **-borderClearSelection:**

clearCursor

- (void)**clearCursor**

Calls `[self clearCursorCol]` and `[self clearCursorRow]`.

See also: -clearCursorRow, -clearCursorCol

clearCursorCol

- (void)**clearCursorCol**

Hides the column keyboard cursor (if any) and sets its position to -1.

See also:

clearCursorRow

- (void)**clearCursorRow**

Hides the row keyboard cursor (if any) and sets its position to -1.

See also:

clearRowSelection

- (void)**clearRowSelection**

Equivalent to: `-borderClearSelection:MISC_ROW_BORDER`.

See also: **-borderClearSelection:**

clearSelection

- (void)**clearSelection**

Calls `[self clearRowSelection]` and `[self clearColSelection]`.

See also:

colAdjustedSize:

- (NXCoord)**colAdjustedSize:**(int)*col*

Returns the current display width of *col*. Equivalent to: `-border:MISC_COL_BORDER slotAdjustedSize:`*`col`*.

See also: -border:slotAdjustedSize:, -rowAdjustedSize:

colAtPosition:

- (int)**colAtPosition:**(int)*pos*

Returns the original physical position of the column at the current visual position *pos*. This is the visual-to-physical conversion routine. Equivalent to: `-border:MISC_COL_BORDER slotAtPosition:`*`pos`*.

See also: -border:slotAtPosition:, -border:slotPosition:, -colPosition:, -rowAtPosition:, -rowPosition:

colCellPrototype:

- **colCellPrototype:**(int)*col*

Returns the cell prototype for column *col*. Equivalent to: `-border:MISC_COL_BORDER slotCellPrototype:col`.

See also: -border:setSlot:cellPrototype:, -border:slotCellPrototype:, -rowCellPrototype:, -setCol:cellPrototype:

colCellType:

- (MiscTableCellStyle)**colCellType:**(int)*col*

Returns the cell type for column *col*. Equivalent to: `-border:MISC_COL_BORDER slotCellType:col`.

See also: -border:setSlot:cellType:, -border:slotCellType:, -rowCellType:, -setCol:cellType:

colDataSize:

- (NXCoord)**colDataSize:**(int)*col*

No description.

See also:

colExpandsToData:

- (BOOL)**colExpandsToData:**(int)*col*

Returns the state of the expands-to-data flag for column *col*. Equivalent to: `-border:MISC_COL_BORDER slotExpandsToData:`*`col`*.

See also: -border:setSlot:expandsToData:, -border:slotExpandsToData:, -rowExpandsToData:, -setCol:expandsToData:

colIsAutosize:

- (BOOL)**colIsAutosize:**(int)*col*

Returns the state of the autosize flag for column *col*. Equivalent to: `-border:MISC_COL_BORDER slotIsAutosize:col`.

See also: -border:setSlot:autosize:, -border:slotIsAutosize:, -rowIsAutosize:, -setCol:autosize:

colIsSelected:

- (BOOL)**colIsSelected:**(MiscCoord_P)*col*

Returns YES if column *col* is selected, else NO. Equivalent to `-border:MISC_COL_BORDER slotIsSelected:col.`

See also: -border:slotIsSelected:

colIsSizeable:

- (BOOL)**colIsSizeable:**(int)*col*

Returns the state of the user-sizeable flag for column *col*. Equivalent to: `-border:MISC_COL_BORDER slotIsSizeable:`*`col`*.

See also: -border:setSlot:sizeable:, -border:slotIsSizeable:, -rowIsSizeable:, -setCol:sizeable:

colIsSorted:

- (BOOL)**colIsSorted:**(int)*col*

Returns YES if *col* is sorted relative to its neighboring columns. Returns NO otherwise. Equivalent to `-border:MISC_COL_BORDER slotIsSorted:`*`col`*.

See also: -border:slotIsSorted:

colMaxSize:

- (NXCoord)**colMaxSize:**(int)*col*

Returns the maximum size for column *col*. Equivalent to: `-border:MISC_COL_BORDER slotMaxSize:`*`col`*.

See also: -border:setSlot:maxSize:, -border:slotMaxSize:, -rowMaxSize:, -setCol:maxSize:

colMinSize:

- (NXCoord)**colMinSize:**(int)*col*

Returns the minimum size for column *col*. Equivalent to: `-border:MISC_COL_BORDER slotMinSize:`*`col`*.

See also: -border:setSlot:minSize:, -border:slotMinSize:, -rowMinSize:, -setCol:minSize:

colOrder:

- **colOrder:**(MiscIntList*)*list*

Equivalent to `-border:MISC_COL_BORDER slotOrder:`*`list`*.

See also: **-border:slotOrder:**

colOrderAsString:size:canExpand:

- (char*)**colOrderAsString:**(char*)*buff*
  size:(int)*buff_size*
  canExpand:(BOOL)*canExpand*

Equivalent to `-border:MISC_COL_BORDER slotOrderAsString:`*buff* `size:`*buff_size* `canExpand:`*canExpand*.

See also: **-border:slotOrderAsString:size:canExpand:**

colPosition:

- (int)**colPosition:**(int)*col*

Returns the current visual position of the column whose original physical position is *pos*. This is the physical-to-

visual conversion routine. Equivalent to: `-border:MISC_COL_BORDER slotPosition:`*`pos`*.

See also: -border:moveSlotFrom:to:, -border:slotAtPosition:, -border:slotPosition:, -colAtPosition:, -moveColFrom:to:, -rowPosition:

colsAreSorted

- (BOOL)**colsAreSorted**

Returns YES if all columns are sorted. Equivalent to `-slotsAreSorted:MISC_COL_BORDER`.

See also: -colIsSorted:, -slotsAreSorted:,

colSize:

- (NXCoord)**colSize:**(int)*col*

Returns the target size for column *col*. Equivalent to: `-border:MISC_COL_BORDER slotSize:`*`col`*.

See also: **-border:setSlot:size:, -border:slotSize:, -rowSize:, -setCol:size:**

colSizes:

- **colSizes:**(MiscIntList*)*list*

Equivalent to: `-border:MISC_COL_BORDER slotSizes:`*`list`*

See also: **-border:slotSizes:**

colSizesAsString:size:canExpand:

- (char*)**colSizesAsString:**(char*)*buff*
  size:(int)*buff_size*
  canExpand:(BOOL)*canExpand*

Equivalent to `-border:MISC_COL_BORDER slotSizesAsString:`*`buff`* `size:`*`buff_size`* `canExpand:`*`canExpand`*.

See also: **-border:slotSizesAsString:size:canExpand:**

colSortDirection:

- (MiscSortDirection)**colSortDirection:**(int)*n*

Returns the sort direction (ascending or descending) of column *n*. Equivalent to: `-border:MISC_COL_BORDER slotSortDirection:`*n*.

See also: **-border:slotSortDirection:**

colSortFunc:

- (MiscCompareEntryFunc)**colSortFunc:**(int)*n*

Equivalent to: `-border:MISC_COL_BORDER slotSortFunc:`*n*.

See also: **-border:slotSortFunc:**

colSortType:

- (MiscSortType)**colSortType:**(int)*n*

Equivalent to `-border:MISC_COL_BORDER slotSortType:`*n*.

See also: **-border:slotSortType:**

colSortVectorLen:

- (int const*)**colSortVectorLen:**(int*)*len*

Equivalent to: `-slotSortVector:MISC_COL_BORDER len:`*len*.

See also: **-slotSortVector:len:**

colTag:

- (int)**colTag:**(int)*col*

Returns the tag for column *col*. Equivalent to: `-border:MISC_COL_BORDER slotTag:`*`col`*.

See also: -border:setSlot:tag:, -border:slotTag:, -rowTag:, -setCol:tag:

colTitle:

- (char const*)**colTitle:**(int)*col*

Returns the title for column *col*. Equivalent to: `-border:MISC_COL_BORDER slotTitle:`*`col`*.

See also: -border:setSlot:title:, -border:slotTitle:, -rowTitle:, -setCol:title:

colTitleMode

- (MiscTableTitleMode)**colTitleMode**

Returns the title-mode for column *col*. Equivalent to: `-border:MISC_COL_BORDER slotTitleMode:`*`col`*.

See also: **-border:setSlot:titleMode:, -border:slotTitleMode:, -rowTitleMode:, -setCol:titleMode:**

colTitlesOn

- (BOOL)**colTitlesOn**

Indicates whether or not column titles are displayed. Equivalent to: `-borderTitlesOn:MISC_COL_BORDER`.

See also: **-border:setTitlesOn:, -borderTitlesOn:, -rowTitlesOn, -setColTitlesOn:**

color

- (NXColor)**color**

Equivalent to: `-backgroundColor`.

See also: **-backgroundColor**

compareCols::

- (int)**compareCols:**(int)*col1* **:**(int)*col2*

Compares two columns. Equivalent to `-border:MISC_COL_BORDER compareSlots:`*`col1`*`:`*`col2`*.

See also: -border:compareSlots::

compareCols::info:

- (int)**compareCols:**(int)*col1* **:**(int)*col2*
  info:(MiscSlotSortInfo*)*sortInfo*

Compares two columns. Equivalent to `-border:MISC_COL_BORDER compareSlots:`*`col1`*`:`*`col2`* `info:`*`sortInfo`*.

See also: -border:compareSlots::info:

compareRows::

- (int)**compareRows:**(int)*row1* **:**(int)*row2*

Compares two columns. Equivalent to `-border:MISC_ROW_BORDER compareSlots:`*`row1`*`:`*`row2`*.

See also: -border:compareSlots::

compareRows::info:

- (int)**compareRows:**(int)*row1* **:**(int)*row2*
  info:(MiscSlotSortInfo*)*sortInfo*

Compares two columns. Equivalent to `-border:MISC_ROW_BORDER compareSlots:`*`row1`*`:`*`row2`* `info:`*`sortInfo`*.

See also: -border:compareSlots::info:

compareSlotFunc

- (MiscCompareSlotFunc)**compareSlotFunc**

Returns the slot comparison function.

See also: **-setCompareSlotFunc:**

constrainMaxTotalHeight:

- (void)**constrainMaxTotalHeight:**(BOOL)*flag*

Equivalent to: `-border:MISC_ROW_BORDER constrainMaxTotalSize:`*`flag`*.

See also: **-border:constrainMaxTotalSize:**

constrainMaxTotalWidth:

- (void)**constrainMaxTotalWidth:**(BOOL)*flag*

Equivalent to: `-border:MISC_COL_BORDER constrainMaxTotalSize:`*`flag`*.

See also: **-border:constrainMaxTotalSize:**

constrainMinTotalHeight:

- (void)**constrainMinTotalHeight:**(BOOL)*flag*

Equivalent to: `-border:MISC_ROW_BORDER constrainMinTotalSize:`*`flag`*.

See also: **-border:constrainMinTotalSize:**

constrainMinTotalWidth:

- (void)**constrainMinTotalWidth:**(BOOL)*flag*

Equivalent to: `-border:MISC_COL_BORDER constrainMinTotalSize:`*`flag`*.

See also: **-border:constrainMinTotalSize:**

constrainSize

- (void)**constrainSize**

Internal method that checks and applies new slot counts and min/max constraints to update the frames of the components of the MiscTableScroll object.

copy:

- **copy:***sender*

Copies the selection to the pasteboard. Calls `-writeSelectionToPasteboard:types:`, with `NXTabularTextPboardType` and `NXAsciiPboardType` for types that should be written. Override this method in your subclass if you want to write different datatypes to the pasteboard.

See also: -writeSelectionToPasteboard:types:

cursorCol

- (MiscCoord_P)**cursorCol**

Returns the column that the column keyboard cursor is on. Meaningless if you are tracking by rows.

See also:

cursorRow

- (MiscCoord_P)**cursorRow**

Returns the row that the row keyboard cursor is on. Meaningless if you are tracking by columns.

See also:

cut:

- **cut:***sender*

Calls `[self copy:`*`sender`*`]`. Nothing is deleted.

See also: -copy:

dataDelegate

- **dataDelegate**

Returns the data delegate of the MiscTableScroll object.

See also: -setDataDelegate

delegate

- **delegate**

Returns the delegate of the MiscTableScroll object.

See also: -setDelegate

deleteColAt:

- **deleteColAt:**(int)*pos*

Deletes column *n*. Equivalent to: `-border:MISC_COL_BORDER deleteSlot:n`.

See also: -addCol, -addSlot:, -border:deleteSlotAt:, -border:insertSlotAt:, -deleteRowAt:, -insertColAt:

deleteRowAt:

- **deleteRowAt:**(int)*pos*

Deletes row *n*. Equivalent to: `-border:MISC_ROW_BORDER deleteSlot:n`.

See also: -addRow, -addSlot:, -border:deleteSlotAt:, -border:insertSlotAt:, -deleteColAt:, -insertRowAt:

doRetireCell:at::

- **doRetireCell:***cell*
  at:(int)*row:(int)col*

This builtin implementation tries to recover storage before the cell is idled. Icon cells are sent `-setIcon:0`. Other cells get the following messages tried in this order: `-setTitle:""`, `-setStringValueNoCopy:""`, `-setStringValue:""`. If the cell responds to the message, that message is sent, otherwise the next message is tried. Override this method in your subclass if you need to do different processing when cells are retired to the cache. Called from: `-retireCell:at::`. Returns *cell*. <???FIXME: Is this also called before the cells are freed??>

See also: -retireCell:at::

doReviveCell:at:

- **doReviveCell:***cell*
  at:(int)*row:(int)col*

This method tries to reset the cell so that it will "`useOwner`..." values for: font, textColor, backgroundColor, highlightTextColor, and highlightBackgroundColor. It tries to set the MiscTableScroll object as the owner of the cell. Then it tries to initialize the font, textColor, backgroundColor, highlightTextColor, and highlightBackgroundColor by first trying the "`setOwner`..." value method, and then trying the straight "`set`..." method if the cell does not respond to the "`setOwner`..." version. Override this method in your subclass if you need different behavior when a cell is brought into active service. Called from `-reviveCell:at::`. Returns *cell*.

See also: -reviveCell:at::

doubleAction

- (SEL)**doubleAction**

Returns the selector message that is sent to the doubleTarget on a double-click event.

See also: -action, -doubleTarget, -setAction:, -setDoubleAction:, -setDoubleTarget:, -setTarget:, -target

doubleTarget

- **doubleTarget**

Returns a pointer to the object which will receive the doubleAction message on a double-click event.

See also: **-action, -doubleAction, -setAction:, -setDoubleAction:, -setDoubleTarget:, -setTarget:, -target**

doubleValueAt::

- (double)**doubleValueAt:**(int)*row* **:**(int)*col*

Returns the value of sending a **-doubleValue** message to the cell at (*row*,*col*). If the table is lazy, the **delegate**, and then the **dataDelegate** are given the opportunity to reply to the **-tableScroll:doubleValueAt::** message. This gives lazy tables an opportunity to return this information directly, without the overhead of preparing and formatting a cell. If the table is not lazy, or the delegate and dataDelegate do not respond to the -tableScroll:doubleValueAt:: message, then the cell is retrieved via **-cellAt::**. If the cell responds to the -doubleValue message, that value is returned; otherwise, zero is returned.

See also: **-cellAt::, -isLazy, -setLazy:, -tableScroll:doubleValueAt::**

draggableCols

- (BOOL)**draggableCols**

Indicates whether or not the user will be allowed to drag (rearrange) the columns. Equivalent to: `-draggableSlots:MISC_COL_BORDER`.

See also: **-border:setSlotsDraggable:, -draggableRows, -draggableSlots:, -setDraggableCols:**

draggableRows

- (BOOL)**draggableRows**

Indicates whether or not the user will be allowed to drag (rearrange) the rows. Equivalent to: `-draggableSlots:MISC_ROW_BORDER`.

See also: **-border:setSlotsDraggable:, -draggableCols, -draggableSlots:, -setDraggableRows:**

draggableSlots:

- (BOOL)**draggableSlots:**(MiscBorderType)*b*

Indicates whether or not the user will be allowed to drag (rearrange) the slots on this border. To enable the user to drag slots, the slots must be draggable, and the titles must be displayed.

See also: **-border:setSlotsDraggable:, -draggableCols, -draggableRows:**

drawCellAt::

- **drawCellAt:**(int)*row* **:**(int)*col*

Instructs the MiscTableScroll object to redraw the cell at position (*row*,*col*). This should be called whenever the contents of a single cell are changed and the screen should be updated to reflect the new state. This method will lock focus on the view if needed.

See also: **-display (View), -drawCol:, -drawRow:**

drawCol:

- **drawCol:**(int)*col*

Instructs the MiscTableScroll object to redraw all the cells in column *col*. This method will lock focus on the view if needed.

See also: **-drawCellAt::, -drawRow:**

drawColTitle:

- **drawColTitle:**(int)*n*

Draws the title for column *col*. This method will lock focus on the view if needed. You should never need to call this.

See also: **-border:drawSlotTitle:, -drawRowTitle:**

drawRow:

- **drawRow:**(int)*row*

Instructs the MiscTableScroll object to redraw all the cells in row *row*. This method will lock focus on the view if needed.

See also: **-drawCellAt::, -drawCol:**

drawRowTitle:

- **drawRowTitle:**(int)*n*

Draws the title for row *row*. This method will lock focus on the view if needed. You should never need to call this.

See also: -border:drawSlotTitle:, -drawColTitle:

empty

- **empty**

Resets the number of rows in the MiscTableScroll to zero. Does not deallocate the rows, nor does it affect the number of columns. The rows are retained in the cache for future use. See **Usage Tips** in the introduction for more details.

See also: -addRow, -border:deleteSlotAt:, -deleteRowAt:, -emptyAndFreeCells, -renewRows:

emptyAndFreeCells

- **emptyAndFreeCells**

Resets the number of rows in the MiscTableScroll to zero; frees all cells stored in the cache, and deallocates all cache resources. Does not affect the number of columns.

See also: **-addRow, -border:deleteSlotAt:, -deleteRowAt:, -empty, -renewRows:**

findCell:row:col:

- **findCell:***cell*
  row:(int*)*row*
  col:(int*)*col*

Finds the location of *cell* in the MiscTableScroll object. If *cell* is found, *row* and *col* are set to the coordinates of the cell in the table, and the method returns **self**. If *cell* is not found, *row* and *col* are set to -1, and the method returns 0.

See also:

findCellWithTag:

- **findCellWithTag:**(int)*x*

Returns the first cell in the body of the table with tag *x*, otherwise 0.

See also:

findCellWithTag:row:col:

- **findCellWithTag:**(int)*x*
  row:(int*)*row*
  col:(int*)*col*

Returns the first cell in the body of the table with tag *x*, and assigns the cell's coordinates to *row* and *col*. If no cell in the table has tag *x*, *row* and *col* are set to -1, and 0 is returned.

See also:

findColWithTag:

- (int)**findColWithTag:**(int)*x*

Returns the index of the first column with tag *x*, or -1 if no columns have tag *x*.

See also:

findRowWithTag:

- (int)**findRowWithTag:**(int)*x*

Returns the index of the first row with tag *x*, or -1 if no row has tag *x*.

See also:

floatValueAt::

- (float)**floatValueAt:**(int)*row* **:**(int)*col*

Returns the value of sending a **-floatValue** message to the cell at (*row*,*col*). If the table is lazy, the **delegate**,

and then the **dataDelegate** are given the opportunity to reply to the **-tableScroll:floatValueAt::** message. This gives lazy tables an opportunity to return this information directly, without the overhead of preparing and formatting a cell. If the table is not lazy, or the delegate and dataDelegate do not respond to the -tableScroll:doubleValueAt:: message, then the cell is retrieved via **-cellAt::**. If the cell responds to the -floatValue message, that value is returned; otherwise, zero is returned.

See also: -cellAt::, -isLazy, -setLazy:, -tableScroll:floatValueAt::

font

- **font**

Returns the current font for the MiscTableScroll object. The current font is used to initialize new cells in the table.

See also:

free

- **free**

Destroys the MiscTableScroll object, reclaiming all resources allocated by it.

See also:

getDocClipFrame:

- (void)**getDocClipFrame:**(NXRect*)*rect*

No description.

See also:

hasColSelection

- (BOOL)**hasColSelection**

Returns YES if any columns are selected, otherwise NO.

See also:

hasMultipleColSelection

- (BOOL)**hasMultipleColSelection**

Returns YES if more than one column is selected, otherwise NO.

See also:

hasMultipleRowSelection

- (BOOL)**hasMultipleRowSelection**

Returns YES if more than one row is selected, otherwise NO.

See also:

hasRowSelection

- (BOOL)**hasRowSelection**

Returns YES if any rows are selected, otherwise NO.

See also:

hasValidCursorCol

- (BOOL)**hasValidCursorCol**

Returns YES if the column keyboard cursor has a valid position in the body of the table, otherwise NO.

See also:

hasValidCursorRow

- (BOOL)**hasValidCursorRow**

Returns YES if the row keyboard cursor has a valid position in the body of the table, otherwise NO.

See also:

highlightBackgroundColor

- (NXColor)**highlightBackgroundColor**

Returns the current highlightBackgroundColor.

See also: -setHighlightBackgroundColor:

highlightBackgroundGray

- (float)**highlightBackgroundGray**

Calls `[self highlightBackgroundColor]`, and converts the color to a gray scale value which is returned.

See also: **-highlightBackgroundColor**

highlightTextColor

- (NXColor)**highlightTextColor**

Returns the current highlightTextColor.

See also: **-setHighlightTextColor:**

highlightTextGray

- (float)**highlightTextGray**

Calls [self highlightTextColor], and converts the color to a gray scale value which is returned.

See also: **-highlightTextColor**

initFrame:

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

Initializes a newly allocated MiscTableScroll object. This is the designated initializer for this class. The newly allocated object will have the following properties set by default:

<???FIXME: write this.?>

See also:

insertColAt:

- **insertColAt:**(int)*pos*

Inserts a new column at position *pos*. Equivalent to: `-border:MISC_COL_BORDER insertSlotAt:`*pos*.

See also: **-addCol:, -addSlot:, -border:deleteSlotAt:, -border:insertSlotAt:, -deleteColAt:, -insertRowAt:**

insertRowAt:

- **insertRowAt:**(int)*pos*

Inserts a new row at position *pos*. Equivalent to: `-border:MISC_ROW_BORDER insertSlotAt:`*pos*.

See also: **-addRow:, -addSlot:, -border:deleteSlotAt:, -border:insertSlotAt:, -deleteRowAt:, -insertColAt:**

intValueAt::

- (int)**intValueAt:**(int)*row* **:**(int)*col*

Returns the value of sending a **-intValue** message to the cell at (*row*,*col*). If the table is lazy, the **delegate**, and then the **dataDelegate** are given the opportunity to reply to the **-tableScroll:intValueAt::** message. This gives lazy tables an opportunity to return this information directly, without the overhead of preparing and formatting a cell. If the table is not lazy, or the delegate and dataDelegate do not respond to the -tableScroll:doubleValueAt::

message, then the cell is retrieved via **-cellAt::**. If the cell responds to the -intValue message, that value is returned; otherwise, zero is returned.

See also: -cellAt::, -isLazy, -setLazy:, -tableScroll:intValueAt::

isEnabled

- (BOOL)**isEnabled**

Indicates whether or not the MiscTableScroll object is enabled for user interaction.

See also: -setEnabled:

isLazy

- (BOOL)**isLazy**

Indicates whether or not the MiscTableScroll object is using lazy-mode memory management.

See also: **-setLazy:**

isSelectable

- (BOOL)**isSelectable**

Indicates whether or not the MiscTableScroll object will allow selection through the user interaction.

See also: **-setSelectable:**

makeCellsPerform:

- (int)**makeCellsPerform:**(SEL)*aSel*

Calls `[self makeCellsPerform:aSel selectedOnly:NO]`.

See also: **-makeCellsPerform:selectedOnly:**

makeCellsPerform:selectedOnly:

- (int)**makeCellsPerform:**(SEL)*aSel*
  selectedOnly:(BOOL)*flag*

Calls `[self makeCellsPerform:aSel with:0 with:0 selectedOnly:flag]`.

See also: -makeCellsPerform:with:with:selectedOnly:

makeCellsPerform:with:

- (int)**makeCellsPerform:**(SEL)*aSel*
  with:*arg1*

Calls `[self makeCellsPerform:aSel with:arg1 selectedOnly:NO]`.

See also: -makeCellsPerform:with:selectedOnly:

makeCellsPerform:with:with:

- (int)**makeCellsPerform:**(SEL)*aSel*
  with:*arg1*
  with:*arg2*

Calls `[self makeCellsPerform:aSel with:arg1 with:arg2 selectedOnly:NO].`

See also: -makeCellsPerform:with:with:selectedOnly:

makeCellsPerform:with:selectedOnly:

- (int)**makeCellsPerform:**(SEL)*aSel*
  with:*arg1*
  selectedOnly:(BOOL)*flag*

Calls `[self makeCellsPerform:aSel with:arg1 with:0 selectedOnly:flag].`

See also: -makeCellsPerform:with:with:selectedOnly:

makeCellsPerform:with:with:selectedOnly:

- (int)**makeCellsPerform:**(SEL)*aSel*
  with:*arg1*
  with:*arg2*
  selectedOnly:(BOOL)*flag*

Sends the message *aSel* to the cells in the table. When *flag* is YES, the message is sent only to selected cells. When *flag* is NO, the message is sent to all cells. First the cell is tested with `-respondsTo:`*`aSel`*. If the cell responds to the message, then the message is sent. Then the return value from the call is inspected. If the cell returns any non-zero value, the process continues. The first cell that returns 0 stops the process. The process also terminates when all cells have been processed. This method returns the number of cells that returned non-zero values.

See also:

maxTotalHeight

- (NXCoord)**maxTotalHeight**

Equivalent to: `-maxTotalSize:MISC_ROW_BORDER`.

See also: -maxTotalSize:

maxTotalHeightIsConstrained

- (BOOL)**maxTotalHeightIsConstrained**

Equivalent to `-maxTotalSizeIsConstrained:MISC_ROW_BORDER`.

See also: -maxTotalSizeIsConstrained:

maxTotalSize:

- (NXCoord)**maxTotalSize:**(MiscBorderType)*b*

Returns the global size limit for the border.

See also: **-border:setMaxTotalSize:**

maxTotalSizeIsConstrained:

- (BOOL)**maxTotalSizeIsConstrained:**(MiscBorderType)*b*

Returns YES if the maximum total size is constrained to the size of the ScrollView in which the table is displayed, otherwise NO.

See also: **-border:constrainMaxTotalSize:**

maxTotalWidth

- (NXCoord)**maxTotalWidth**

Equivalent to: `-maxTotalSize:MISC_COL_BORDER`.

See also: **-maxTotalSize:**

maxTotalWidthIsConstrained

- (BOOL)**maxTotalWidthIsConstrained**

Equivalent to `-maxTotalSizeIsConstrained:MISC_COL_BORDER`.

See also: **-maxTotalSizeIsConstrained:**

minTotalHeight

- (NXCoord)**minTotalHeight**

Equivalent to: `-minTotalSize:MISC_ROW_BORDER`.

See also: **-minTotalSize:**

minTotalHeightIsConstrained

- (BOOL)**minTotalHeightIsConstrained**

Equivalent to: `-minTotalSizeIsConstrained:MISC_ROW_BORDER`.

See also: -minTotalSizeIsConstrained:

minTotalSize:

- (NXCoord)**minTotalSize:**(MiscBorderType)*b*

Returns the minimum total size for the border.

See also: -border:setMinTotalSize:

minTotalSizeIsConstrained:

- (BOOL)**minTotalSizeIsConstrained:**(MiscBorderType)*b*

Returns YES if the minimum total size for border *b* is constrained to the size of the ScrollView in which the table is displayed, otherwise NO.

See also: **-border:constrainMinTotalSize:**

minTotalWidth

- (NXCoord)**minTotalWidth**

Equivalent to: `-minTotalSize:MISC_COL_BORDER`.

See also: **-minTotalSize:**

minTotalWidthIsConstrained

- (BOOL)**minTotalWidthIsConstrained**

Equivalent to: `-minTotalSizeIsConstrained:MISC_COL_BORDER`.

See also: **-minTotalSizeIsConstrained:**

modifierDragCols

- (BOOL)**modifierDragCols**

Indicates whether or not the command-key must be held down to drag columns. It is false by default. Equivalent to: `-modifierDragSlots:MISC_COL_BORDER`.

See also: **-border:setModifierDragSlots:, -modifierDragRows, -modifierDragSlots:, -setModifierDragCols:**

modifierDragRows

- (BOOL)**modifierDragRows**

Indicates whether or not the command-key must be held down to drag rows. It is true by default. Equivalent to:

`-modifierDragSlots:MISC_ROW_BORDER`.

See also: **-border:setModifierDragSlots:, -modifierDragCols, -modifierDragSlots:, -setModifierDragRows:**

modifierDragSlots:

- (BOOL)**modifierDragSlots:**(MiscBorderType)*b*

Indicates whether or not the command-key must be held down to drag the slots on this border.

See also: **-border:setModifierDragSlots:, -modifierDragCols, -modifierDragRows**

moveColFrom:to:

- **moveColFrom:**(int)*from_pos*
  to:(int)*to_pos*

Moves the column at visual position *from_pos* to visual position *to_pos*. Equivalent to `-border:MISC_COL_BORDER moveSlotFrom:`*`from_pos`* `to:`*`to_pos`*.

See also: -border:moveSlotFrom:to:, -border:slotAtPosition:, -border:slotPosition:, -colAtPosition:, -colPosition:, -moveRowFrom:to:

moveRowFrom:to:

- **moveRowFrom:**(int)*from_pos*
  to:(int)*to_pos*

Moves the row at visual position *from_pos* to visual position *to_pos*. Equivalent to `-border:MISC_ROW_BORDER moveSlotFrom:`*`from_pos`* `to:`*`to_pos`*.

See also: -border:moveSlotFrom:to:, -border:slotAtPosition:, -border:slotPosition:, -moveColFrom:to:, -rowAtPosition:, -rowPosition:,

nextText

- **nextText**

Returns the object that will become the first responder when the user presses the TAB key while the MiscTableScroll object is the first responder.

See also: -previousText, -setNextText:, -setPreviousText:

numCols

- (int)**numCols**

Returns the number of columns in the MiscTableScroll object. Equivalent to: -numSlots:MISC_COL_BORDER.

See also: -addCol, -addSlot:, -border:deleteSlotAt:, -border:insertSlotAt:, -deleteColAt:, -insertColAt:, -numRows, -numSlots:

numRows

- (int)**numRows**

Returns the number of rows in the MiscTableScroll object. This is the number of "active" rows currently being displayed. The MiscTableScroll object performs caching on a row-oriented basis. There may be additional rows allocated, and stored in the cache. Equivalent to: -numSlots:MISC_ROW_BORDER.

See also: -addRow, -addSlot:, -border:deleteSlotAt:, -border:insertSlotAt:, -deleteRowAt:, -insertRowAt:, -numCols, -numSlots:

numSelectedCols

- (unsigned int)**numSelectedCols**

Returns the number of selected columns.

See also:

numSelectedRows

- (unsigned int)**numSelectedRows**

Returns the number of selected rows.

See also:

numSlots:

- (int)**numSlots:**(MiscBorderType)*b*

Returns the number of slots for the border *b*.

See also: -addSlot:, -border:deleteSlotAt:, -border:insertSlotAt:, -numCols, -numRows

previousText

- **previousText**

Returns a pointer to the object that will become the first responder if the user presses the SHIFT-TAB key while the MiscTableScroll object is the first responder.

See also: -nextText, -setNextText:, -setPreviousText:

readSelectionFromPasteboard:

- **readSelectionFromPasteboard:***pboard*

This method is invoked when a service returns some data. If the delegate responds to the `-tableScroll:readSelectionFromPasteboard:` message, it is sent to the delegate. If not, then the dataDelegate is given the opportunity. If neither responds to the message, `-builtinReadSelectionFromPasteboard:` is called. Returns the results of the subroutine that was called. Override this method in your subclass if you need different behavior.

See also: -builtinReadSelectionFromPasteboard:, -tableScroll:readSelectionFromPasteboard: (delegate method), -readSelectionFromPasteboard: (NXServicesRequests), -writeSelectionToPasteboard:types: (NXServicesRequests)

reflectSelection

- (void)**reflectSelection**

Updates the display to reflect the currently selected rows and columns. When the selection is modified programmatically, the display is not updated until and unless this method is called. Whenever you change the selection programmatically, you must call this function to update the display. This method will lock focus on the view if needed.

See also:

registerServicesTypes

- (void)**registerServicesTypes**

If the delegate responds to the `-tableScrollRegisterServicesTypes:` message, the message is sent to the delegate. If not, the dataDelegate is tried. If neither responds to the message,

`-builtinRegisterServicesTypes` is called. This method is invoked when an MiscTableScroll object is initialized. Override this method in your subclass if you need different behavior.

See also: -builtinRegisterServicesTypes, -tableScrollRegisterServicesTypes: (delegate method)

renewRows:

- **renewRows:**(int)*count*

Sets the number of active rows in the MiscTableScroll object to *count*; does not affect the number of columns. This is the fastest way to change the size of a MiscTableScroll object when you know the number of rows in advance. See **Usage Tips** in the introduction for more details.

See also: -addRow, -addSlot:, -border:deleteSlotAt:, -border:insertSlotAt:, -deleteRowAt:, -empty, -emptyAndFreeCells, -insertRowAt:

retireCell:at::

- **retireCell:***cell*
  at:(int)*row* **:**(int)*col*

Internal method called whenever a cell is being removed from active use and returned to the cache. If the delegate responds to the -tableScroll:retireCell:at:: message, it is sent to the delegate. If not, the dataDelegate is tried. If the dataDelegate also does not respond to the message, the cell itself is checked. If none of these objects responds to the message, a builtin default method, -doRetireCell:at:: is called. Override this method in your subclass if you need different behavior.

See also: -doRetireCell:at::, -tableScroll:retireCell:at:: (delegate method)

reviveCell:at::

- **reviveCell:***cell*
  at:(int)*row* **:**(int)

Internal method called whenever a cell is is being moved into active use. This method is applied to both newly created cells returned by the `-copy` method of the column's cell prototype and cells retrieved from the cache. If

the delegate responds to the `-tableScroll:reviveCell:at:` message, it is sent to the delegate. If not, the dataDelegate is checked. If neither the delegate nor the dataDelegate respond to the message, the cell itself is checked. If none of these objects respond to the message, a builtin default method `-doReviveCell:at:` is called. Override this method in your subclass if you need different behavior.

See also: -doReviveCell:at:, -tableScroll:reviveCell:at: (delegate method)

rowAdjustedSize:

- (NXCoord)**rowAdjustedSize:**(int)*row*

Returns the current display height of *row*. Equivalent to: `-border:MISC_ROW_BORDER slotAdjustedSize:`*row*.

See also: -border:slotAdjustedSize:, -colAdjustedSize:

rowAtPosition:

- (int)**rowAtPosition:**(int)*pos*

Returns the original physical position of the row at the current visual position *pos*. This is the visual-to-physical conversion routine. Equivalent to: `-border:MISC_ROW_BORDER slotAtPosition:`*pos*.

See also: -border:slotAtPosition:, -border:slotPosition:, -colAtPosition:, -colPosition:, -rowPosition:

rowCellPrototype:

- **rowCellPrototype:**(int)*row*

Returns the cell prototype for row *row*. Equivalent to: `-border:MISC_ROW_BORDER slotCellPrototype:`*row*.

See also: -border:setSlot:cellPrototype:, -border:slotCellPrototype:, -colCellPrototype:, -setRow:cellPrototype:

rowCellType:

- (MiscTableCellStyle)**rowCellType:**(int)*row*

Returns the cell type for row *row*. Equivalent to: `-border:MISC_ROW_BORDER slotCellType:`*`row`*.

See also: -border:setSlot:cellType:, -border:slotCellType:, -colCellType:, -setRow:cellType:

rowDataSize:

- (NXCoord)**rowDataSize:**(int)*row*

No description.

See also:

rowExpandsToData:

- (BOOL)**rowExpandsToData:**(int)*row*

Returns the state of the expands-to-data flag for row *row*. Equivalent to: `-border:MISC_ROW_BORDER slotExpandsToData:`*`row`*.

See also: **-border:setSlot:expandsToData:, -border:slotExpandsToData:, -colExpandsToData:, -setRow:expandsToData:**

rowIsAutosize:

- (BOOL)**rowIsAutosize:**(int)*row*

Returns the state of the autosize flag for row *row*. Equivalent to: `-border:MISC_ROW_BORDER slotIsAutosize:`*row*.

See also: **-border:setSlot:autosize:, -border:slotIsAutosize:, -colIsAutosize:, -setRow:autosize:**

rowIsSelected:

- (BOOL)**rowIsSelected:**(MiscCoord_P)*row*

Returns YES if *row* is selected, otherwise NO.

See also:

rowIsSizeable:

- (BOOL)**rowIsSizeable:**(int)*row*

Returns the state of the user-sizeable flag for row *row*. Equivalent to: `-border:MISC_ROW_BORDER slotIsSizeable:`*`row`*.

See also: -border:setSlot:sizeable:, -border:slotIsSizeable:, -colIsSizeable:, -setRow:sizeable:

rowIsSorted:

- (BOOL)**rowIsSorted:**(int)*col*

Returns YES if *row* is sorted relative to its neighboring rows. Returns NO otherwise. Equivalent to `-border:MISC_ROW_BORDER slotIsSorted:`*`row`*.

See also: **-border:slotIsSorted:**

rowMaxSize:

- (NXCoord)**rowMaxSize:**(int)*row*

Returns the maximum size for row *row*. Equivalent to: `-border:MISC_ROW_BORDER slotMaxSize:`*row*.

See also: **-border:setSlot:maxSize:, -border:slotMaxSize:, -colMaxSize:, -setRow:maxSize:**

rowMinSize:

- (NXCoord)**rowMinSize:**(int)*row*

Returns the minimum size for row *row*. Equivalent to: `-border:MISC_ROW_BORDER slotMinSize:`*row*.

See also: **-border:setSlot:minSize:, -border:slotMinSize:, -colMinSize:, -setRow:minSize:**

rowOrder:

- **rowOrder:**(MiscIntList*)*list*

Equivalent to: `-border:MISC_ROW_BORDER slotOrder:`*`list`*.

See also: **-border:slotOrder:**

rowOrderAsString:size:canExpand:

- (char*)**rowOrderAsString:**(char*)*buff*
  size:(int)*buff_size*
  canExpand:(BOOL)*canExpand*

Equivalent to: `-border:MISC_ROW_BORDER slotOrderAsString:`*`buff`* `size:`*`buff_size`* `canExpand:`*`canExpand`*.

See also: **-border:slotOrderAsString:size:canExpand:**

rowPosition:

- (int)**rowPosition:**(int)*row*

Returns the current visual position of the row whose original physical position is *pos*. This is the physical-to-visual conversion routine. Equivalent to: `-border:MISC_ROW_BORDER slotPosition:`*pos*.

See also: -border:moveSlotFrom:to:, -border:slotAtPosition:, -border:slotPosition:, -colAtPosition:, -colPosition:, -moveRowFrom:to:, -rowAtPosition:

rowsAreSorted

- (BOOL)**rowsAreSorted**

Returns YES if all rows are sorted. Equivalent to `-slotsAreSorted:MISC_ROW_BORDER`.

See also: -rowIsSorted:, -slotsAreSorted:,

rowSize:

- (NXCoord)**rowSize:**(int)*row*

Returns the target size for row *row*. For the actual current display size, use **-rowAdjustedSize:**. Equivalent to: `-border:MISC_ROW_BORDER slotSize:`*row*.

See also: -border:setSlot:size:, -border:slotSize:, -colSize:, -rowAdjustedSize:, -setRow:size:

rowSizes:

- **rowSizes:**(MiscIntList*)*list*

Equivalent to: `-border:MISC_ROW_BORDER slotSizes:`*list*.

See also: -border:slotSizes:

rowSizesAsString:size:canExpand:

- (char*)**rowSizesAsString:**(char*)*buff*
  size:(int)*buff_size*
  canExpand:(BOOL)*canExpand*

Equivalent to: `-border:MISC_ROW_BORDER slotSizesAsString:`*`buff`* `size:`*`buff_size`* `canExpand:`*`canExpand`*.

See also: -border:slotSizesAsString:size:canExpand:

rowSortDirection:

- (MiscSortDirection)**rowSortDirection:**(int)*n*

Equivalent to: `-border:MISC_ROW_BORDER slotSortDirection:`*`n`*.

See also: -border:slotSortDirection:

rowSortFunc:

- (MiscCompareEntryFunc)**rowSortFunc:**(int)*n*

Equivalent to: `-border:MISC_ROW_BORDER slotSortFunc:`*n*.

See also: -border:slotSortFunc:

rowSortType:

- (MiscSortType)**rowSortType:**(int)*n*

Equivalent to: `-border:MISC_ROW_BORDER slotSortType:`*n*.

See also: -border:slotSortType:

rowSortVectorLen:

- (int const*)**rowSortVectorLen:**(int*)*len*

Equivalent to: `-slotSortVector:MISC_ROW_BORDER len:`*len*.

See also: **-slotSortVector:len:**

rowTag:

- (int)**rowTag:**(int)*row*

Returns the tag for row *row*. Equivalent to: `-border:MISC_ROW_BORDER slotTag:`*row*.

See also: **-border:setSlot:tag:, -border:slotTag:, -colTag:, -setRow:tag:**

rowTitle:

- (char const*)**rowTitle:**(int)*row*

Returns the title for row *row*. Equivalent to: `-border:MISC_ROW_BORDER slotTitle:`*row*.

See also: **-border:setSlot:title:, -border:slotTitle:, -colTitle:, -setRow:title:**

rowTitleMode

- (MiscTableTitleMode)**rowTitleMode**

Returns the title-mode for row *row*. Equivalent to: `-border:MISC_ROW_BORDER slotTitleMode:`*row*.

See also: -border:setSlot:titleMode:, -border:slotTitleMode:, -colTitleMode:, -setRow:titleMode:

rowTitlesOn

- (BOOL)**rowTitlesOn**

Indicates whether or not row titles are displayed. Equivalent to: `-borderTitlesOn:MISC_ROW_BORDER`.

See also: -border:setTitlesOn:, -borderTitlesOn:, -colTitlesOn, -setRowTitlesOn:

scrollCellToVisible::

- (void)**scrollCellToVisible:**(int)*row* **:**(int)*col*

Scrolls the display as necessary until the cell at position *row*, *col* is visible.

See also:

scrollColToVisible:

- (void)**scrollColToVisible:**(int)*col*

Scrolls the display as necessary until *col* is visible.

See also:

scrollRowToVisible:

- (void)**scrollRowToVisible:**(int)*row*

Scrolls the display as necessary until *row* is visible.

See also:

scrollSelToVisible

- **scrollSelToVisible**

Scrolls the display as necessary until the selection is visible.

See also:

selectAll:

- **selectAll:** *sender*

Calls `[self selectAllRows]`, then `[self sendActionIfEnabled]`.

See also: -selectAllRows, -sendActionIfEnabled

selectAllCols

- (void)**selectAllCols**

Equivalent to: `-borderSelectAll:MISC_COL_BORDER`.

See also: -borderSelectAll:

selectAllRows

- (void)**selectAllRows**

Equivalent to: `-borderSelectAll:MISC_ROW_BORDER`.

See also: -borderSelectAll:

selectCol:

- (void)**selectCol:**(MiscCoord_P)*col*

Equivalent to: `-border:MISC_COL_BORDER selectSlot:`*`col`*.

See also: -border:selectSlot:

selectColTags:

- (void)**selectColTags:**(MiscIntList*)*tags*

Equivalent to: `-border:MISC_COL_BORDER selectTags:`*`tags`*.

See also: -border:selectTags:

selectCols:

- (void)**selectCols:**(MiscIntList*)*cols*

Equivalent to: `-border:MISC_COL_BORDER selectSlots:`*`cols`*.

See also: **-border:selectSlots:**

selectRow:

- (void)**selectRow:**(MiscCoord_P)*row*

Equivalent to: `-border:MISC_ROW_BORDER selectSlot:`*`row`*.

See also: **-border:selectSlot:**

selectRowTags:

- (void)**selectRowTags:**(MiscIntList*)*tags*

Equivalent to: `-border:MISC_ROW_BORDER selectTags:`*`tags`*.

See also: **-border:selectTags:**

selectRows:

- (void)**selectRows:**(MiscIntList*)*rows*

Equivalent to: `-border:MISC_ROW_BORDER selectSlots:`*rows*.

See also: -border:selectSlots:

selectText:

- **selectText:***sender*

Makes the MiscTableScroll object the first responder, enabling it to accept keyboard input.

See also: -isEnabled, -isSelectable, -nextText, -previousText, -setEnabled:, -setNextText:, -setPreviousText:, -setSelectable:

selectedCol

- (MiscCoord_P)**selectedCol**

Equivalent to: `-borderSelectedSlot:MISC_COL_BORDER`.

See also: **-borderSelectedSlot:**

selectedColTags:

- (void)**selectedColTags:**(MiscIntList*)*tags*

Equivalent to: `-border:MISC_COL_BORDER selectedTags:`*tags*.

See also: **-border:selectedTags:**

selectedCols:

- (void)**selectedCols:**(MiscIntList*)*cols*

Equivalent to: `-border:MISC_COL_BORDER selectedSlots:`*`cols`*.

See also: -border:selectedSlots:

selectedRow

- (MiscCoord_P)**selectedRow**

Equivalent to: `-borderSelectedSlot:MISC_ROW_BORDER`.

See also: -borderSelectedSlot:

selectedRowTags:

- (void)**selectedRowTags:**(MiscIntList*)*tags*

Equivalent to: `-border:MISC_ROW_BORDER selectedTags:`*`tags`*.

See also: -border:selectedTags:

selectedRows:

- (void)**selectedRows:**(MiscIntList*)*rows*

Equivalent to: `-border:MISC_ROW_BORDER selectedSlots:`*`cols`*.

See also: -border:selectedSlots:

selectionMode

- (MiscSelectionMode)**selectionMode**

Returns the current setting of the selection mode.

See also: -setSelectionMode:

sendAction

- **sendAction**

Sends the **action** message to the **target** object. Implemented via -sendAction:to:. Returns **self**. <???FIXME: Should match the behavior of Matrix.?>

See also: -**action**, ± **doubleAction**, ± **setDoubleAction:**, ± **target**, ± **doubleTarget**, ± **setTarget:**, ± **setDoubleTarget:**

sendAction:to:

- **sendAction:**(SEL)*theAction*
  to:*theTarget*

Uses the Application class's -sendAction:to:from: method to send the message *theAction* to the object *theTarget* from the MiscTableScroll object itself. Returns self. <???FIXME: Should match the behavior of Matrix.?>

See also: -€**action**, ± **doubleAction**, ± **setDoubleAction:**, ± **target**, ± **doubleTarget**, ± **setTarget:**, ± **setDoubleTarget:**

sendAction:to:forAllCells:

- **sendAction:**(SEL)*aSelector*
  to:*anObject*
  forAllCells:(BOOL)*flag*

Sends the message *aSelector* to *anObject* for each cell in the table. <???FIXME: Should match the behavior of Matrix.?>

See also: -€**action**, ± **doubleAction**, ± **setDoubleAction:**, ± **target**, ± **doubleTarget**, ± **setTarget:**, ± **setDoubleTarget:**

sendActionIfEnabled

- **sendActionIfEnabled**

If `[self isEnabled]` returns YES, then `[self sendAction]` is called.

See also: **-isEnabled, -sendAction**

sendDoubleAction

- **sendDoubleAction**

Sends the **doubleAction** message to the **doubleTarget** object.

See also: **-€action**, ± **doubleAction**, ± **setDoubleAction:**, ± **target**, ± **doubleTarget**, ± **setTarget:**, ± **setDoubleTarget:**

sendDoubleActionIfEnabled

- **sendDoubleActionIfEnabled**

If `[self isEnabled]` returns YES, then `[self sendDoubleAction]` is called.

See also: **-isEnabled, -sendDoubleAction**

setAction:

- **setAction:**(SEL)*new_sel*

Sets the action method to *new_sel*. The action message is sent to the **target** upon a single mouse click. The argument of an action method is the table scroll. Returns **self**.

See also: -€**action**, ± **doubleAction**, ± **setDoubleAction:**, ± **target**, ± **doubleTarget**, ± **setTarget:**, ± **setDoubleTarget:**

setAutoSortCols:

- (void)**setAutoSortCols:**(BOOL)*flag*

Equivalent to: `-border:MISC_COL_BORDER setAutoSortSlots:`*`flag`*.

See also: **-border:setAutoSortSlots:**

setAutoSortRows:

- (void)**setAutoSortRows:**(BOOL)*flag*

Equivalent to: `-border:MISC_ROW_BORDER setAutoSortSlots:`*`flag`*.

See also: -border:setAutoSortSlots:

setAutodisplay:

- **setAutodisplay:**(BOOL)*x*

Overridden from View. Propagates the autodisplay setting to the component subviews. Returns **self**.

See also: -€setAutodisplay: (View)

setBackgroundColor:

- **setBackgroundColor:**(NXColor)*value*

Sets the backgroundColor. The backgroundColor is used to initialize new cells, and also to paint the background of areas that are not covered by cells of the table. By default, this is the value returned by `+defaultBackgroundColor`.

See also: +defaultBackgroundColor

setBackgroundGray:

- **setBackgroundGray:**(float)*value*

The gray scale value, *value*, is converted to a color value, and the color value is sent to -setBackgroundColor:.

See also: -setBackgroundColor:

setCol:autosize:

- (void)**setCol:**(int)*col*
  autosize:(BOOL)*flag*

Sets the autosize flag for column *col*. Equivalent to: `-border:MISC_COL_BORDER setSlot:`*`col`*` autosize:`*`flag`*.

See also: -border:setSlot:autosize:, -border:slotIsAutosize:, -colIsAutosize:, -setRow:autosize:

setCol:cellPrototype:

- (void)**setCol:**(int)*col*
  cellPrototype:*cell*

Sets the cell prototype for column *col* to *cell*. Equivalent to: `-border:MISC_COL_BORDER setSlot:`*`col`*` cellPrototype:`*`cell`*.

See also: -border:setSlot:cellPrototype:, -border:slotCellPrototype:, -colCellPrototype:, -setRow:cellPrototype:

setCol:cellType:

- (void)**setCol:**(int)*col*
  cellType:(MiscTableCellStyle)*type*

Sets the cell type for column *col* to *type*. Equivalent to: `-border:MISC_COL_BORDER setSlot:`*`col`* `cellType:`*`type`*.

See also: -border:setSlot:cellType:, -border:slotCellType:, -colCellType:, -setRow:cellType:

setCol:dataSize:

- (void)**setCol:**(int)*col*
  dataSize:(NXCoord)*size*

No description.

See also:

setCol:expandsToData:

- (void)**setCol:**(int)*col*
  expandsToData:(BOOL)*flag*

Sets the expands-to-data flag for column *col* to *flag*. Equivalent to: `-border:MISC_COL_BORDER setSlot:`*`col`* `expandsToData:`*`flag`*.

See also: -border:setSlot:expandsToData:, -border:slotExpandsToData:, -colExpandsToData:, -setRow:expandsToData:

setCol:maxSize:

- (void)**setCol:**(int)*col*
  maxSize:(NXCoord)*size*

Sets the maximum size of column *col* to *size*. Equivalent to: `-border:MISC_COL_BORDER setSlot:`*`col`* `maxSize:`*`size`*.

See also: **-border:setSlot:maxSize:, -border:slotMaxSize:, -colMaxSize:, -setRow:maxSize:**

setCol:minSize:

- (void)**setCol:**(int)*col*
  minSize:(NXCoord)*size*

Sets the minimum size of column *col* to *size*. Equivalent to: `-border:MISC_COL_BORDER setSlot:`*`col`*` minSize:`*`size`*.

See also: **-border:setSlot:minSize:, -border:slotMinSize:, -colMinSize:, -setRow:minSize:**

setCol:size:

- (void)**setCol:**(int)*col*
  size:(NXCoord)*size*

Sets the target size of column *col* to *size*. Equivalent to: `-border:MISC_COL_BORDER setSlot:col size:size`.

See also: -border:setSlot:size:, -border:slotSize:, -colSize:, -setRow:size:

setCol:sizeable:

- (void)**setCol:**(int)*col*
  sizeable:(BOOL)*flag*

Sets the user-sizeable flag for column *col* to *flag*. Equivalent to: `-border:MISC_COL_BORDER setSlot:col sizeable:flag`.

See also: -border:setSlot:sizeable:, -border:slotIsSizeable:, -colIsSizeable:, -setRow:sizeable:

setCol:sortDirection:

- (void)**setCol:**(int)*n*
  sortDirection:(MiscSortDirection)*x*

Equivalent to: `-border:MISC_COL_BORDER setSlot:`*n* `sortDirection:`*x*.

See also: **-border:setSlot:sortDirection:**

setCol:sortFunc:

- (void)**setCol:**(int)*n*
  sortFunc:(MiscCompareEntryFunc)*x*

Equivalent to: `-border:MISC_COL_BORDER setSlot:`*n* `sortFunc:`*x*.

See also: **-border:setSlot:sortFunc:**

setCol:sortType:

- (void)**setCol:**(int)*n*
  sortType:(MiscSortType)*x*

Equivalent to: `-border:MISC_COL_BORDER setSlot:`*`n`* `sortType:`*`x`*.

See also: -border:setSlot:sortType:

setCol:tag:

- (void)**setCol:**(int)*col*
  tag:(int)*tag*

Sets the tag for column *col* to *tag*. Equivalent to: `-border:MISC_COL_BORDER setSlot:`*`col`* `tag:`*`tag`*.

See also: -border:setSlot:tag:, -border:slotTag:, -colTag:, -setRow:tag:

setCol:title:

- (void)**setCol:**(int)*col*
  title:(char const*)*title*

Sets the title for column *col* to *title*. Equivalent to: `-border:MISC_COL_BORDER setSlot:`*`col`* `title:`*`title`*.

See also: -border:setSlot:title:, -border:slotTitle:, -colTitle:, -setRow:title:

setColOrder:

- **setColOrder:**(MiscIntList*)*list*

Equivalent to: `-border:MISC_COL_BORDER setSlotOrder:`*`list`*

See also: -border:setSlotOrder:

setColOrderFromString:

- **setColOrderFromString:**(char const*)*s*

Equivalent to: `-border:MISC_COL_BORDER setSlotOrderFromString:`*`s`*.

See also: -border:setSlotOrderFromString:

setColSizes:

- **setColSizes:**(MiscIntList*)*list*

Equivalent to: `-border:MISC_COL_BORDER setSlotSizes:`*`list`*.

See also: **-border:setSlotSizes:**

setColSizesFromString:

- **setColSizesFromString:**(char const*)*s*

Equivalent to: `-border:MISC_COL_BORDER setSlotSizesFromString:s`.

See also: **-border:setSlotSizesFromString:**

setColSortVector:len:

- (void)**setColSortVector:**(int const*)*v*
  len:(int)*n*

Equivalent to: `-border:MISC_COL_BORDER setSlotSortVector:`*v* `len:`*n*.

See also: -border:setSlotSortVector:len:

setColTitleMode:

- (void)**setColTitleMode:**(MiscTableTitleMode)*x*

Sets the title-mode for column *col* to *x*. Equivalent to: `-border:MISC_COL_BORDER setSlot:`*col* `titleMode:`*x*.

See also: -border:setSlot:titleMode:, -border:slotTitleMode:, -colTitleMode:, -setRow:titleMode:

setColTitlesOn:

- (BOOL)**setColTitlesOn:**(BOOL)*on_off*

Turns the column titles on or off. When *on_off* is YES, column titles will be displayed. When *on_off* is NO, column titles will not be displayed. Column titles are displayed by default. Equivalent to: `-border:MISC_COL_BORDER setTitlesOn:`*`on_off`*.

See also: -border:setTitlesOn:, -borderTitlesOn:, -colTitlesOn, -setRowTitlesOn:

setColor:

- **setColor:**(NXColor)*value*

Equivalent to: `-setBackgroundColor:`*`value`*.

See also: -setBackgroundColor:

setCompareSlotFunc:

- (void)**setCompareSlotFunc:**(MiscCompareSlotFunc)*f*

Makes *f* the slot comparison function to be used for sorting. It must conform to the following prototype from `<MiscTableTypes.h>`:

```
typedef int (*MiscCompareSlotFunc)( int slot1, int slot2, MiscSlotSortInfo* );
```

The function must return an integer value which is: (a) less than zero if *slot1* should come before *slot2*, or (b) equal to zero if *slot1* should sort equally with *slot2*, or (c) greater than zero if *slot1* should come after *slot2*. This function is responsible for comparing the cells of the two slots in the order defined by the slotSortVector, or visual order if no explicit slotSortVector has been set. This function is also responsible for applying the sort direction to the individual cell-wise comparisons. This function is also responsible for calling user-installed custom slot sorting functions, or interpreting and applying the sort-type for slots that do not have a custom function. The default, builtin implementation of this function is `MiscDefaultCompareSlotFunc`.

See also: -border:setSlot:sortDirection:, -border:setSlot:sortFunc:, -border:setSlot:sortType:, -border:setSlotSortVector:len:, -compareSlotFunc, -sortInfoDone:, -sortInfoInit:border:

setCursorCol:

- (void)**setCursorCol:**(MiscCoord_P)*col*

Equivalent to: `-border:MISC_COL_BORDER setCursor:`*`col`*.

See also: **-border:setCursor:**

setCursorRow:

- (void)**setCursorRow:**(MiscCoord_P)*row*

Equivalent to: `-border:MISC_ROW_BORDER setCursor:`*`row`*.

See also: **-border:setCursor:**

setDataDelegate:

- **setDataDelegate:***obj*

Makes *obj* the data delegate for the MiscTableScroll object. <???FIXME: When is the data delegate used, what messages are passed to it, etc.?> Returns **self**.

See also: -dataDelegate, -delegate, -setDelegate:, -setLazy:

setDelegate:

- **setDelegate:***obj*

Makes *obj* the delegate for the MiscTableScroll object. <???FIXME: When is the delegate used, what messages are passed to it, etc.?> Returns **self**.

See also: -dataDelegate, -delegate, -setDataDelegate:

setDoubleAction:

- **setDoubleAction:**(SEL)*new_sel*

Sets the double-action method to *new_sel*. The double-action message is sent to the **doubleTarget** upon a double mouse click. The argument of an action method is the table scroll. Returns **self**.

See also: -€**action**, ± **doubleAction**, ± **setAction:**, ± **target**, ± **doubleTarget**, ± **setTarget:**, ± **setDoubleTarget:**

setDoubleTarget:

- **setDoubleTarget:***obj*

Makes *obj* the **doubleTarget** of the MiscTableScroll object.

See also: **-action, -doubleAction, -setAction:, -setDoubleAction:, -target, -doubleTarget, -setDoubleAction:, -setTarget:**

setDraggableCols:

- (void)**setDraggableCols:**(BOOL)*flag*

Enables or disables user-dragging of columns. When *flag* is YES, columns will be user-draggable. When *flag* is NO, columns will not be user-draggable. The column titles must be displayed to enable the user to drag columns. Equivalent to: `-border:MISC_COL_BORDER setDraggableSlots:`*`flag`*.

See also: -border:setDraggableSlots:, -border:setTitlesOn:, -draggableCols, -draggableSlots:, -setColTitlesOn:, -setDraggableRows:

setDraggableRows:

- (void)**setDraggableRows:**(BOOL)*flag*

Enables or disables user-dragging of rows. When *flag* is YES, rows will be user-draggable. When *flag* is NO, rows will not be user-draggable. The row titles must be displayed to enable the user to drag rows. Equivalent to: `-border:MISC_ROW_BORDER setDraggableSlots:`*`flag`*.

See also: -border:setDraggableSlots:, -border:setTitlesOn:, -draggableRows, -draggableSlots:,

-setDraggableCols:, -setRowTitlesOn:

setEnabled:

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

Enables or disables user-interaction with the MiscTableScroll object. When *flag* is YES, the user will be able to interact with the MiscTableScroll object normally. When *flag* is NO, the user will be able to scroll the MiscTableScroll object, but nothing else; no selection, no dragging, no resizing, no keyboard interaction. <??? FIXME: Check this.?>

See also: -isEnabled, -isSelectable, -setSelectable:

setFont:

- **setFont:***newFont*

Sets the font for the MiscTableScroll object. The font is used to initialize new cells in the table. If rows are

uniformly sized, the uniform row size is adjusted proportionately based on the sizes of the old font and the new font. Then all the cells are updated. If the cells respond to the `-setOwnerFont:` message, that message is sent. Otherwise the `-setFont:` message is tried. Then the `-tableScroll:fontChangedFrom:to:` message is sent to the delegate if the delegate responds to it. Finally, the display is updated.

See also: -tableScroll:fontChangedFrom:to: (delegate), -setOwnerFont: (MiscTableCell), -setFont: (Cell, MiscTableCell)

setHighlightBackgroundColor:

- **setHighlightBackgroundColor:**(NXColor)*value*

Sets the highlightBackgroundColor for the MiscTableScroll object. The highlightBackgroundColor is used to initialize new cells added to the table. This information is propagated to the cells of the table as follows. If the cells respond to the `-setOwnerHighlightBackgroundColor:` message, that message is sent, else if the cells respond to the `-setHighlightBackgroundColor:` message, that message is sent instead. If the cells do not respond to either of these messages, no message is sent to the cell. Finally, the display is updated.

See also: **-setHighlightBackgroundColor: (MiscTableCell), -setOwnerHighlightBackgroundColor: (MiscTableCell)**

setHighlightBackgroundGray:

- **setHighlightBackgroundGray:**(float)*value*

This method takes the gray scale value, *value*, converts it to a color value which is passed to `-setHighlightBackgroundColor:`.

See also: **-setHighlightBackgroundColor:**

setHighlightTextColor:

- **setHighlightTextColor:**(NXColor)*value*

Sets the highlightTextColor for the MiscTableScroll object. The highlightTextColor is used to initialize new cells added to the table. This message is also propagated to the existing cells of the table as follows. If the cells

respond to the `-setOwnerHighlightTextColor:` message, that message is sent, else if the cells respond to the `-setHighlightTextColor:` message, that message is sent. If the cells do not respond to either of these messages, no message is sent to the cell. Finally, the display is updated.

See also: -setHighlightTextColor: (MiscTableCell), -setOwnerHighlightTextColor: (MiscTableCell)

setHighlightTextGray:

- **setHighlightTextGray:**(float)*value*

This method takes the gray scale value, *value*, converts it to a color value which is passed to `-setHighlightTextColor:`.

See also: -setHighlightTextColor:

setLazy:

- (void)**setLazy:**(BOOL)*flag*

Enables or disables lazy-mode memory management. When *flag* is YES, the MiscTableScroll object will use lazy-mode memory management, asking the dataDelegate to provide the cells in the body of the table. When *flag* is NO, the MiscTableScroll object will use eager-mode memory management, maintaining a dense, 2-D array of cell pointers, one pointer for each cell in the table, and caching cells on a row-wise basis. MiscTableScroll use eager-mode memory management by default. See **Usage Tips**, and **Lazy vs. Eager**, in the introduction for more details.

See also: -dataDelegate, -isLazy, -setDataDelegate:

setMaxTotalHeight:

- (void)**setMaxTotalHeight:**(NXCoord)*size*

Equivalent to: `-border:MISC_ROW_BORDER setMaxTotalSize:`*`size`*.

See also: -border:setMaxTotalSize:

setMaxTotalWidth:

- (void)**setMaxTotalWidth:**(NXCoord)*size*

Equivalent to: `-border:MISC_COL_BORDER setMaxTotalSize:`*`size`*.

See also: -border:setMaxTotalSize:

setMinTotalHeight:

- (void)**setMinTotalHeight:**(NXCoord)*size*

Equivalent to: `-border:MISC_ROW_BORDER setMinTotalSize:`*`size`*.

See also: -border:setMinTotalSize:

setMinTotalWidth:

- (void)**setMinTotalWidth:**(NXCoord)*size*

Equivalent to: `-border:MISC_COL_BORDER setMinTotalSize:`*`size`*.

See also: -border:setMinTotalSize:

setModifierDragCols:

- (void)**setModifierDragCols:**(BOOL)*flag*

Sets whether or not the command-key must be held down to drag columns. By default, columns require the command-key to perform selection. Equivalent to `-border:MISC_COL_BORDER setModifierDragSlots:`*`flag`*.

See also: -border:setModifierDragSlots:, -modifierDragCols, -modifierDragSlots:, -setModifierDragRows:

setModifierDragRows:

- (void)**setModifierDragRows:**(BOOL)*flag*

Sets whether or not the command-key must be held down to drag rows. By default, rows do not require the command-key to perform selection. Equivalent to `-border:MISC_ROW_BORDER setModifierDragSlots:`*`flag`*.

See also: -border:setModifierDragSlots:, -modifierDragRows, -modifierDragSlots:, -setModifierDragCols:

setNextText:

- **setNextText:***obj*

Makes *obj* the object that will become first responder when the user presses the TAB key while the MiscTableScroll object is the first responder. The **-setPreviousText:** message will be sent to *obj* if *obj* responds to that message, so that the next-text chain will be maintained properly.

See also: -nextText, -previousText, -setPreviousText:

setPreviousText:

- **setPreviousText:***obj*

Makes *obj* the object that will become first responder when the user presses the SHIFT-TAB key while the MiscTableScroll object is the first responder.

See also: -nextText, -previousText, -setNextText:

setRow:autosize:

- (void)**setRow:**(int)*row*
  autosize:(BOOL)*flag*

Sets the autosize flag for row *row*. Equivalent to: `-border:MISC_ROW_BORDER setSlot:`*`row`* `autosize:`*`flag`*.

See also: -border:setSlot:autosize:, -border:slotIsAutosize:, -rowIsAutosize:, -setCol:autosize:

setRow:cellPrototype:

- (void)**setRow:**(int)*row*
  cellPrototype:*cell*

Sets the cell prototype for row *row* to *cell*. Currently, only column cell prototypes are used. Equivalent to: `-border:MISC_ROW_BORDER setSlot:`*`row`*` cellPrototype:`*`cell`*.

See also: -border:setSlot:cellPrototype:, -border:slotCellPrototype:, -rowCellPrototype:, -setCol:cellPrototype:

setRow:cellType:

- (void)**setRow:**(int)*row*
  cellType:(MiscTableCellStyle)*type*

Sets the cell type for row *row* to *type*. Equivalent to: `-border:MISC_ROW_BORDER setSlot:`*`row`*` cellType:`*`type`*.

See also: -border:setSlot:cellType:, -border:slotCellType:, -rowCellType:, -setCol:cellType:

setRow:dataSize:

- (void)**setRow:**(int)*row*
  dataSize:(NXCoord)*size*

No description.

See also:

setRow:expandsToData:

- (void)**setRow:**(int)*row*
  expandsToData:(BOOL)*flag*

Sets the expands-to-data flag for row *row* to *flag*. Equivalent to: `-border:MISC_ROW_BORDER setSlot:`*`row`*` expandsToData:`*`flag`*.

See also: -border:setSlot:expandsToData:, -border:slotExpandsToData:, -rowExpandsToData:, -setCol:expandsToData:

setRow:maxSize:

- (void)**setRow:**(int)*row*
  maxSize:(NXCoord)*size*

Sets the maximum size of row *row* to *size*. Equivalent to: `-border:MISC_ROW_BORDER setSlot:`*`row`*` maxSize:`*`size`*.

See also: -border:setSlot:maxSize:, -border:slotMaxSize:, -rowMaxSize:, -setCol:maxSize:

setRow:minSize:

- (void)**setRow:**(int)*row*
  minSize:(NXCoord)*size*

Sets the minimum size of row *row* to *size*. Equivalent to: `-border:MISC_ROW_BORDER setSlot:`*`row`*` minSize:`*`size`*.

See also: **-border:setSlot:minSize:, -border:slotMinSize:, -rowMinSize:, -setCol:minSize:**

setRow:size:

- (void)**setRow:**(int)*row*
  size:(NXCoord)*size*

Sets the target size of row *row* to *size*. Equivalent to: `-border:MISC_ROW_BORDER setSlot:`*`row`* `size:`*`size`*.

See also: **-border:setSlot:size:, -border:slotSize:, -rowSize:, -setCol:size:**

setRow:sizeable:

- (void)**setRow:**(int)*row*
  sizeable:(BOOL)*flag*

Sets the user-sizeable flag for row *row* to *flag*. Equivalent to: `-border:MISC_ROW_BORDER setSlot:`*`row`*

`sizeable:`*`flag`*.

See also: **-border:setSlot:sizeable:, -border:slotIsSizeable:, -rowIsSizeable:, -setCol:sizeable:**

setRow:sortDirection:

- (void)**setRow:**(int)*n*
  sortDirection:(MiscSortDirection)*x*

Equivalent to: `-border:MISC_ROW_BORDER setSlot:`*`n`* `sortDirection:`*`x`*.

See also: **-border:setSlot:sortDirection:**

setRow:sortFunc:

- (void)**setRow:**(int)*n*
  sortFunc:(MiscCompareEntryFunc)*x*

Equivalent to: `-border:MISC_ROW_BORDER setSlot:`*`n`* `sortFunc:`*`x`*.

See also: **-border:setSlot:sortFunc:**

setRow:sortType:

- (void)**setRow:**(int)*n*
  sortType:(MiscSortType)*x*

Equivalent to: `-border:MISC_ROW_BORDER setSlot:`*`n`* `sortType:`*`x`*.

See also: **-border:setSlot:sortType:**

setRow:tag:

- (void)**setRow:**(int)*row*
  tag:(int)*tag*

Sets the tag for row *row* to *tag*. Equivalent to: `-border:MISC_ROW_BORDER setSlot:`*`row`* `tag:`*`tag`*.

See also: -border:setSlot:tag:, -border:slotTag:, -rowTag:, -setCol:tag:

setRow:title:

- (void)**setRow:**(int)*row*
  title:(char const*)*title*

Sets the title for row *row* to *title*. Equivalent to: `-border:MISC_ROW_BORDER setSlot:`*`row`* `title:`*`title`*.

See also: -border:setSlot:title:, -border:slotTitle:, -rowTitle:, -setCol:title:

setRowOrder:

- **setRowOrder:**(MiscIntList*)*list*

Equivalent to: `-border:MISC_ROW_BORDER setSlotOrder:`*`list`*.

See also: **-border:setSlotOrder:**

setRowOrderFromString:

- **setRowOrderFromString:**(char const*)*s*

Equivalent to: `-border:MISC_ROW_BORDER setSlotOrderFromString:`*s*.

See also: **-border:setSlotOrderFromString:**

setRowSizes:

- **setRowSizes:**(MiscIntList*)*list*

Equivalent to: `-border:MISC_ROW_BORDER setSlotSizes:`*list*.

See also: **-border:setSlotSizes:**

setRowSizesFromString:

- **setRowSizesFromString:**(char const*)*s*

Equivalent to: `-border:MISC_ROW_BORDER setSlotSizesFromString:`*s*.

See also: **-border:setSlotSizesFromString:**

setRowSortVector:len:

- (void)**setRowSortVector:**(int const*)*v*
  len:(int)*n*

Equivalent to: `-border:MISC_ROW_BORDER setSlotSortVector:`*v* `len:`*n*.

See also: **-border:setSlotSortVector:len:**

setRowTitleMode:

- (void)**setRowTitleMode:**(MiscTableTitleMode)*x*

Sets the title-mode for row *row* to *x*. Equivalent to: `-border:MISC_ROW_BORDER setSlot:`*`row`*` titleMode:`*`x`*.

See also: -border:setSlot:titleMode:, -border:slotTitleMode:, -rowTitleMode:, -setCol:titleMode:

setRowTitlesOn:

- (BOOL)**setRowTitlesOn:**(BOOL)*on_off*

Turns the row titles on or off. When *on_off* is YES, row titles will be displayed. When *on_off* is NO, row titles will not be displayed. Row titles are not displayed by default. Equivalent to: `-border:MISC_ROW_BORDER setTitlesOn:`*`on_off`*.

See also: -border:setTitlesOn:, -borderTitlesOn:, -rowTitlesOn, -setColTitlesOn:

setSelectionMode:

- (void)**setSelectionMode:**(MiscSelectionMode)*x*

Sets the selection mode for the MiscTableScroll object. The selection mode, *x*, can be any of the following:

```
MISC_LIST_MODE,
MISC_RADIO_MODE,
MISC_HIGHLIGHT_MODE
```

The modes each correspond to the similarly named selection modes declared in the Matrix class. The MiscTableScroll object extends the highlight mode selection by implementing the Alternate-key modifier in the same fashion that it works in list mode.

See also: -selectionMode

setSizeableCols:

- (void)**setSizeableCols:**(BOOL)*flag*

Enables or disables user-sizing of columns. Equivalent to: `-border:MISC_COL_BORDER setSizeableSlots:`*`flag`*.

See also: **-border:setSizeableSlots:, -setSizeableRows:, -sizeableCols, -sizeableSlots:**

setSizeableRows:

- (void)**setSizeableRows:**(BOOL)*flag*

Enables or disables user-sizing of rows. Equivalent to: `-border:MISC_ROW_BORDER setSizeableSlots:`*flag*.

See also: **-border:setSizeableSlots:, -setSizeableCols:, -sizeableRows, -sizeableSlots:**

setTag:

- **setTag:**(int)*x*

Sets the **tag** of the MiscTableScroll object to *x*. Returns **self**.

See also: **-border:setSlot:tag:, -setCol:tag:, -setRow:tag:, -tag**

setTarget:

- **setTarget:***obj*

Makes *obj* the object which will receive the **action** message whenever there is a single mouse-click on the body of the table. Returns **self**.

See also: -action, -doubleAction, -doubleTarget, -setAction:, -setDoubleAction:, -setDoubleTarget:, -target

setTextColor:

- **setTextColor:**(NXColor)*value*

Sets the textColor for the MiscTableScroll object. The textColor is used to initialize new cells added to the table. The message is propagated to existing cells as follows. If the cell responds to the `-setOwnerTextColor:` message, that message is sent, else if the cell responds to the `-setTextColor:` message, that message is sent. If the cell does not respond to either of these messages, no message is sent to the cell. Finally, the display is

updated.

See also: -setOwnerTextColor: (MiscTableCell), -setTextColor: (Cell, MiscTableCell).

setTextGray:

- **setTextGray:**(float)*value*

The gray scale value, *value*, is converted to a color value which is passed to the -setTextColor: method.

See also: -setTextColor:

setUniformSizeCols:

- (void)**setUniformSizeCols:**(NXCoord)*uniform_size*

Sets or clears the uniform-size property for columns. When *uniform_size* is a non-zero value, all columns will have the same, fixed, (uniform) size. When *uniform_size* is zero, each column can be assigned sizes

individually. By default, columns are not uniformly sized. Equivalent to: `-border:MISC_COL_BORDER setUniformSizeSlots:`*`uniform_size`*.

See also: -border:setUniformSizeSlots:, -setUniformSizeRows:, -uniformSizeCols, -uniformSizeSlots:

setUniformSizeRows:

- (void)**setUniformSizeRows:**(NXCoord)*uniform_size*

Sets or clears the uniform-size property for rows. When *uniform_size* is a non-zero value, all rows will have the same, fixed, (uniform) size. When *uniform_size* is zero, each row can be assigned sizes individually. By default, row are uniformly sized. Equivalent to: `-border:MISC_ROW_BORDER setUniformSizeSlots:`*`uniform_size`*.

See also: -border:setUniformSizeSlots:, -setUniformSizeCols:, -uniformSizeRows, -uniformSizeSlots:

sizeToCells

- **sizeToCells**

Instructs the MiscTableScroll object to adjust the frames of its subviews.

See also: -addRow

sizeableCols

- (BOOL)**sizeableCols**

Indicates whether or not columns can be resized by the user. Equivalent to: `-sizeableSlots:MISC_COL_BORDER`.

See also: -border:setSizeableSlots:, -setColsSizeable:, -sizeableRows

sizeableRows

- (BOOL)**sizeableRows**

Indicates whether or not rows can be resized by the user. Equivalent to: `-sizeableSlots:MISC_ROW_BORDER`.

See also: **-border:setSizeableSlots:, -setRowsSizeable:, -sizeableCols**

sizeableSlots:

- (BOOL)**sizeableSlots:**(MiscBorderType)*b*

Indicates whether or not the user can resize the slots on border *b*.

See also: **-border:setSizeableSlots:, -sizeableCols, -sizeableRows**

slotsAreSorted:

- (BOOL)**slotsAreSorted:**(MiscBorderType)*b*

Returns YES if the slots are sorted, NO otherwise.

See also: **-border:slotIsSorted:, -border:sortSlot:, -sortSlots:**

slotSortVector:len:

- (int const*)**slotSortVector:**(MiscBorderType)*b*
  len:(int*)*len*

Returns the current slotSortVector for border *b*, and puts its length into *len*.

See also: -border:setSlotSortVector:len:

sortCol:

- (void)**sortCol:**(int)*n*

Re-sorts a single column. Equivalent to: -border:MISC_COL_BORDER sortSlot:*n*.

See also: -border:sortSlot:, -sortSlots:

sortCols

- (void)**sortCols**

Equivalent to: -sortSlots:MISC_COL_BORDER.

See also: -sortSlots:

sortInfoDone:

- (void)**sortInfoDone:**(MiscSlotSortInfo*)*sortInfo*

This method reclaims temporary storage held in the *sortInfo* structure. You must call this method whenever you are finished using a *sortInfo* object.

See also: -sortInfoInit:border:

sortInfoInit:border:

- (void)**sortInfoInit:**(MiscSlotSortInfo*)*sortInfo*
  border:(MiscBorderType)*b*

This method precomputes the sorting information needed by the sorting methods. If you call any of the sorting methods that accept an **info:** argument, you must initialize the *sortInfo* structure by calling this method first. After you have finished using the *sortInfo* structure, you must reclaim the storage by passing the *sortInfo* structure to **-sortInfoDone:**. NOTE: The *sortInfo* structure stores the current sorting information for the table. Any changes made to the sorting environment after the *sortInfo* structure has been initialized will not affect the contents of the *sortInfo* structure, and therefore will not affect comparisons made using the *sortInfo* structure. Actions that affect the sorting environment include: rearranging columns/rows, installing a slotSortVector, installing a custom slot comparison function, changing the sort-type or sort-direction of a slot. Actions which alter the structure of the "other" border (like removing columns/rows) can potentially cause catastrophic failures.

See also: -border:compareSlots::info:,

sortRow:

- (void)**sortRow:**(int)*n*

Re-sorts a single row. Equivalent to: -border:MISC_ROW_BORDER sortSlot:*n*.

See also: -border:sortSlot:, -sortSlots:

sortRows

- (void)**sortRows**

Equivalent to: -sortSlots:MISC_ROW_BORDER.

See also: -sortSlots:

sortSlots:

- (void)**sortSlots:**(MiscBorderType)*b*

Sorts the slots in border *b*.

See also: **-border:setSlot:sortDirection:, -border:setSlot:sortFunc:, -border:setSlot:sortType:, -border:setSlotSortVector:len:, -setCompareSlotFunc:,**

stateAt::

- (int)**stateAt:**(int)*row* **:**(int)*col*

Returns the value of sending a **-state** message to the cell at (*row*,*col*). If the table is lazy, the **delegate**, and then the **dataDelegate** are given the opportunity to reply to the **-tableScroll:stateAt::** message. This gives lazy tables an opportunity to return this information directly, without the overhead of preparing and formatting a cell. If the table is not lazy, or the delegate and dataDelegate do not respond to the -tableScroll:doubleValueAt:: message, then the cell is retrieved via **-cellAt::**. If the cell responds to the -state message, that value is returned; otherwise, zero is returned.

See also: **-cellAt::, -isLazy, -setLazy:, -tableScroll:stateAt::, -state (ButtonCell)**

stringValueAt::

- (char const*)**stringValueAt:**(int)*row* **:**(int)*col*

Returns the value of sending a **-stringValue** message to the cell at (*row*,*col*). If the table is lazy, the **delegate**, and then the **dataDelegate** are given the opportunity to reply to the **-tableScroll:stringValueAt::** message. This gives lazy tables an opportunity to return this information directly, without the overhead of preparing and formatting a cell. If the table is not lazy, or the delegate and dataDelegate do not respond to the -tableScroll:doubleValueAt:: message, then the cell is retrieved via **-cellAt::**. If the cell responds to the -stringValue message, that value is returned; otherwise, zero (a NULL pointer) is returned. NOTE: If you are using ButtonCells, you probably want **-titleAt::**, not this method.

See also: -cellAt::, -isLazy, -setLazy:, -tableScroll:stringValueAt::, -titleAt::

tag

- (int)**tag**

Returns the **tag** of the MiscTableScroll object.

See also: -border:slotTag:, -colTag:, -rowTag:, -setTag:

tagAt::

- (int)**tagAt:**(int)*row* **:**(int)*col*

Returns the value of sending a **-tag** message to the cell at (*row*,*col*). If the table is lazy, the **delegate**, and then the **dataDelegate** are given the opportunity to reply to the **-tableScroll:tagAt::** message. This gives lazy tables an opportunity to return this information directly, without the overhead of preparing and formatting a cell. If the table is not lazy, or the delegate and dataDelegate do not respond to the -tableScroll:doubleValueAt:: message, then the cell is retrieved via **-cellAt::**. If the cell responds to the -tag message, that value is returned; otherwise, zero is returned.

See also: -cellAt::, -isLazy, -setLazy:, -tableScroll:tagAt::

target

- **target**

Returns a pointer to the object which receives the **action** message on a single mouse-click event.

See also: **-action, -doubleAction, -doubleTarget, -setAction:, -setDoubleAction:, -setDoubleTarget:, -setTarget:**

textColor

- (NXColor)**textColor**

Returns the current textColor.

See also: **-setTextColor:**

textGray

- (float)**textGray**

Calls `[self textColor]`, and converts the color to a gray scale value which is returned.

See also: **-textColor**

titleAt::

- (char const*)**titleAt:**(int)*row* **:**(int)*col*

Returns the value of sending a **-title** message to the cell at (*row*,*col*). If the table is lazy, the **delegate**, and then the **dataDelegate** are given the opportunity to reply to the **-tableScroll:titleAt::** message. This gives lazy tables an opportunity to return this information directly, without the overhead of preparing and formatting a cell. If the table is not lazy, or the delegate and dataDelegate do not respond to the -tableScroll:doubleValueAt:: message, then the cell is retrieved via **-cellAt::**. If the cell responds to the -title message, that value is returned; otherwise, zero (a NULL pointer) is returned. NOTE: ButtonCell implements the **-stringValue** message by formatting the integer value of the ButtonCell's state as a string. To retrieve the text label displayed on the button, you must use the **-title** method.

See also: **-cellAt::, -isLazy, -setLazy:, -tableScroll:titleAt::, -title (ButtonCell)**

totalHeight

- (NXCoord)**totalHeight**

Equivalent to: `-totalSize:MISC_ROW_BORDER`.

See also: -totalSize:

totalSize:

- (NXCoord)**totalSize:**(MiscBorderType)*b*

Returns the total display size. The sum of `-border:`*b* `slotAdjustedSize:` for all slots on the border.

See also: -border:slotAdjustedSize:

totalWidth

- (NXCoord)**totalWidth**

Equivalent to: `-totalSize:MISC_COL_BORDER`.

See also: **-totalSize:**

trackBy:

- (void)**trackBy:**(MiscBorderType)*b*

Sets keyboard tracking to either row-wise or column-wise tracking.

See also:

trackingBy

- (MiscBorderType)**trackingBy**

Returns the current orientation of keyboard tracking.

See also:

uniformSizeCols

- (NXCoord)**uniformSizeCols**

Returns the uniform size for columns. If columns are not being sized uniformly, this method will return zero. Any non-zero value indicates the size that all columns have the same size as the value returned by this method. By default, columns are not uniformly sized. Equivalent to: `-uniformSizeSlots:MISC_COL_BORDER`.

See also: -border:setUniformSize:, -setUniformSizeCols:, -uniformSizeRows, -uniformSizeSlots:

uniformSizeRows

- (NXCoord)**uniformSizeRows**

Returns the uniform size for rows. If rows are not being sized uniformly, this method will return zero. Any non-zero value indicates the size that all rows have the same size as the value returned by this method. Rows are uniformly sized by default. Equivalent to: `-uniformSizeSlots:MISC_ROW_BORDER`.

See also: **-border:setUniformSize:, -setUniformSizeRows:, -uniformSizeCols, -uniformSizeSlots:**

uniformSizeSlots:

- (NXCoord)**uniformSizeSlots:**(MiscBorderType)*b*

Returns the uniform size for slots in border *b*. If slots are not being sized uniformly, this method will return zero. Any non-zero value indicates the size that all slots have the same size as the value returned by this method.

See also: **-border:setUniformSize:, -uniformSizeCols, -uniformSizeRows**

validRequestorForSendType:andReturnType:

- **validRequestorForSendType:**(NXAtom)*t_write*
  andReturnType:(NXAtom)*t_read*

This method is called by the services system to update the services menu. If the delegate responds to the

`-tableScroll:validRequestorForSendType:andReturnType:` message, it is sent to the delegate. If not, the dataDelegate is checked. If neither object responds, `-builtinValidRequestorForSendType:andReturnType:` is called. Override this method in your subclass if you need different behavior.

See also: -builtinValidRequestorForSendType:andReturnType:, -tableScroll:validRequestorForSendType:andReturnType: (delegate method)

writeNXAsciiPboardTypeToStream:

- (void)**writeNXAsciiPboardTypeToStream:**(NXStream*)*stream*

Writes the selected cells to *stream* as ASCII text. Columns are separated by tab characters (ASCII decimal 9). Rows are terminated with newline characters (ASCII decimal 10). The text is retrieved from the cells by first trying the -title message. If the cell does not respond to the -title message, then the -stringValue message is tried. Each tab character in the text retrieved from the cell is replaced with a single space character (ASCII decimal 32) before the text is written to *stream*. The selection is written in the current (visual) ordering. Called from `-builtinWritePboard:type:toStream:`, and `-writeNXTabularTextPboardTypeToStream:`. Override this method in your subclass if you want different behavior.

See also: **-builtinWritePboard:type:toStream:, -writeNXTabularTextPboardTypeToStream:**

writeNXTabularTextPboardTypeToStream:

- (void)**writeNXTabularTextPboardTypeToStream:**(NXStream*)*stream*

Calls `[self writeNXAsciiPboardTypeToStream:`*`stream`*`]`. Called from `-builtinWritePboard:type:toStream:`. Override this method in your subclass if you want different behavior.

See also: **-builtinWritePboard:type:toStream:, -writeNXAsciiPboardTypeToStream:**

writePboard:type:toStream:

- (void)**writePboard:***pboard*
  type:(NXAtom)*type*
  toStream:(NXStream*)*stream*

This method is responsible for writing data to the pasteboard. If the delegate responds to `-tableScroll:writePboard:type:toStream:`, the message is sent to the delegate. If not, the dataDelegate is tried. If neither object responds to the message, the default method, `-builtinWritePboard:type:toStream:`, is called. Called from `-builtinWriteSelectionToPasteboard:types:`. Override this method in your subclass if you want different behavior.

See also: -builtinWritePboard:type:toStream:, -builtinWriteSelectionToPasteboard:types:, -tableScroll:writePboard:type:toStream: (delegate method)

writeSelectionToPasteboard:types:

- (BOOL)**writeSelectionToPasteboard:***pboard*
  types:(NXAtom*)*types*

This method is invoked from the services system. If the delegate responds to `-tableScroll:writeSelectionToPasteboard:types:`, the message is sent to the delegate. If not, the dataDelegate is tried. If neither object responds to the message, the default method, `-builtinWriteSelectionToPasteboard:types:` is called. Override this method in your subclass if you want

different behavior.

See also: -builtinWriteSelectionToPasteboard:types:, -tableScroll:writeSelectionToPasteboard:types: (delegate method), -readSelectionFromPasteboard: (NXServicesRequests), -writeSelectionToPasteboard:types: (NXServicesRequests)

Methods Implemented by the Delegate

tableScroll:backgroundColorChangedTo:

- **tableScroll:**(MiscTableScroll*)*scroll*
  backgroundColorChangedTo:(NXColor)*newColor*

This message is sent to the **delegate** and then the **dataDelegate** when the MiscTableScroll receives a `-setBackgroundColor:` message that actually changes the background color.

See also: -setBackgroundColor:

tableScroll:border:slotDraggedFrom:to:

- **tableScroll:**(MiscTableScroll*)*scroll*
  border:(MiscBorderType)*b*
  slotDraggedFrom:(int)*from_pos*
  to:(int)*to_pos*

Notifies the delegate whenever the user drags a slot to a new position.

See also:

tableScroll:border:slotPrototype:

- **tableScroll:**(MiscTableScroll*)*scroll*
  border:(MiscBorderType)*b*
  slotPrototype:(int)*slot*

Sent to the delegate whenever the MiscTableScroll object needs the prototype cell for a column which has the

`MISC_TABLE_CELL_CALLBACK` cell type. If the delegate does not respond to the message, the dataDelegate is tried.

See also: -border:setSlot:cellType:

tableScroll:border:slotResized:

- **tableScroll:**(MiscTableScroll*)*scroll*
  border:(MiscBorderType)*b*
  slotResized:(int)*n*

Notifies the delegate whenever the user resizes a slot.

See also:

tableScroll:border:slotTitle:

- (char const*)**tableScroll:**(MiscTableScroll*)*scroll*

border:(MiscBorderType)*b*
slotTitle:(int)*slot*

Sent to the delegate whenever the MiscTableScroll object needs a title for a border which has the `MISC_DELEGATE_TITLE` title mode. If the delegate does not respond to the message, the dataDelegate is tried.

See also: -border:setTitleMode:

tableScroll:canWritePboardType:

- (BOOL)**tableScroll:**(MiscTableScroll*)*scroll*
  canWritePboardType:(NXAtom)*type*

If the delegate responds to this message, the delegate has the opportunity to select which datatypes will be written to the pasteboard. If the delegate does not respond, the dataDelegate is given the opportunity.

See also: -canWritePboardType:

tableScroll:cellAt::

- **tableScroll:**(MiscTableScroll*)*scroll*
  cellAt:(int)*row* **:**(int)*col*

If the table scroll is in *lazy* mode this message is sent first to the **delegate** and then to the **dataDelegate** (if delegate does not respond) whenever the cell at *row*, *col* is needed. You must implement this method in either the delegate or the dataDelegate whenever you use a MiscTableScroll in lazy mode. The table scroll does not manage the cells for itself in lazy mode; the delegate or the dataDelegate must.

See also: ± **setLazy:**, ± **isLazy**

tableScroll:changeFont:to:

- **tableScroll:**(MiscTableScroll*)*scroll*
  changeFont:(Font*)*oldFont*
  to:(Font*)*newFont*

This message is sent to the **delegate** and then the **dataDelegate** whenever a **changeFont:** message is received

and the new font is different than the current font. The FontManager sends the **changeFont:** message whenever the user changes the font using either the FontPanel or the Font Menu. This is distinguished from programmatic changes via the **setFont:** method so that you can record user preferences. This notification message is sent after the font the change has been applied, but before the new font is displayed.

See also: -changeFont:, -setFont:, -tableScroll:fontChangedFrom:to:

tableScroll:doubleValueAt::

- (double) **tableScroll:**(MiscTableScroll*)*scroll*
  doubleValueAt:(int)*row* **:**(int)*col*

Lazy tables send this message to the **delegate** and then the **dataDelegate** to retrieve the value for **-doubleValueAt::**. If the delegate or dataDelegate respond to this message, that value is returned. If neither responds to the message, or if the table is not lazy, the cell is retrieved via **-cellAt::**. If the cell responds to the **-doubleValue** message, that value is returned; otherwise zero is returned. This method gives lazy tables the opportunity to provide the information content of cells without the overhead of preparing and formatting a cell. You should implement this method in your delegate or dataDelegate if you have any slots that contain `double`

values.

See also: -cellAt::, -doubleValueAt::, -isLazy, -setLazy:

tableScroll:floatValueAt::

- (float) **tableScroll:**(MiscTableScroll*)*scroll*
  floatValueAt:(int)*row* **:**(int)*col*

Lazy tables send this message to the **delegate** and then the **dataDelegate** to retrieve the value for **-floatValueAt::**. If the delegate or dataDelegate respond to this message, that value is returned. If neither responds to the message, or if the table is not lazy, the cell is retrieved via **-cellAt::**. If the cell responds to the **-floatValue** message, that value is returned; otherwise zero is returned. This method gives lazy tables the opportunity to provide the information content of cells without the overhead of preparing and formatting a cell. You should implement this method in your delegate or dataDelegate if you have any slots that contain `float` values.

See also: -cellAt::, -floatValueAt::, -isLazy, -setLazy:

tableScroll:fontChangedFrom:to:

- **tableScroll:**(MiscTableScroll*)*scroll*
  fontChangedFrom:(Font*)*oldFont*
  to:(Font*)*newFont*

This message is sent to the **delegate** and then the **dataDelegate** whenever a **setFont:** message is received and the new font is different than the current font. This notification message is sent after all font changes, both user-initiated and programmatic. This message is sent after the font change has been applied, but before the new font is displayed.

See also: -changeFont:, -setFont:, -tableScroll:changeFont:to:

tableScroll:highlightBackgroundColorChangedTo:

- **tableScroll:**(MiscTableScroll*)*scroll*
  highlightBackgroundColorChangedTo:(NXColor)*newColor*

This message is sent to the **delegate** and then the **dataDelegate** when the MiscTableScroll receives a `-setHighlightBackgroundColor:` message that actually changes the background color.

See also: **-setHighlightBackgroundColor:**

tableScroll:highlightTextColorChangedTo:

- **tableScroll:**(MiscTableScroll*)*scroll*
  highlightTextColorChangedTo:(NXColor)*newColor*

This message is sent to the **delegate** and then the **dataDelegate** when the MiscTableScroll receives a `-setHighlightTextColor:` message that actually changes the background color.

See also: **-setHighlightTextColor:**

tableScroll:intValueAt::

- (int) **tableScroll:**(MiscTableScroll*)*scroll*

intValueAt:(int)*row* **:**(int)*col*

Lazy tables send this message to the **delegate** and then the **dataDelegate** to retrieve the value for **-intValueAt::**. If the delegate or dataDelegate respond to this message, that value is returned. If neither responds to the message, or if the table is not lazy, the cell is retrieved via **-cellAt::**. If the cell responds to the **-intValue** message, that value is returned; otherwise zero is returned. This method gives lazy tables the opportunity to provide the information content of cells without the overhead of preparing and formatting a cell. You should implement this method in your delegate or dataDelegate if you have any slots that contain `int` values.

See also: -cellAt::, -intValueAt::, -isLazy, -setLazy:

tableScroll:stateAt::

- (int) **tableScroll:**(MiscTableScroll*)*scroll*
  stateAt:(int)*row* **:**(int)*col*

Lazy tables send this message to the **delegate** and then the **dataDelegate** to retrieve the value for **-stateAt::**.

If the delegate or dataDelegate respond to this message, that value is returned. If neither responds to the message, or if the table is not lazy, the cell is retrieved via **-cellAt::**. If the cell responds to the **-state** message, that value is returned; otherwise zero is returned. This method gives lazy tables the opportunity to provide the information content of cells without the overhead of preparing and formatting a cell. You should implement this method in your delegate or dataDelegate if you have any slots that contain `state` values.

See also: -cellAt::, -stateAt::, -isLazy, -setLazy:, -state (ButtonCell)

tableScroll:stringValueAt::

- (char const*) **tableScroll:**(MiscTableScroll*)*scroll*
  stringValueAt:(int)*row* **:**(int)*col*

Lazy tables send this message to the **delegate** and then the **dataDelegate** to retrieve the value for **-stringValueAt::**. If the delegate or dataDelegate respond to this message, that value is returned. If neither responds to the message, or if the table is not lazy, the cell is retrieved via **-cellAt::**. If the cell responds to the **-stringValue** message, that value is returned; otherwise zero is returned. This method gives lazy tables the opportunity to provide the information content of cells without the overhead of preparing and formatting a cell.

You should implement this method in your delegate or dataDelegate if you have any slots that contain `string` values.

See also: -cellAt::, -stringValueAt::, -isLazy, -setLazy:

tableScroll:tagAt::

- (int) **tableScroll:**(MiscTableScroll*)*scroll*
  tagAt:(int)*row* **:**(int)*col*

Lazy tables send this message to the **delegate** and then the **dataDelegate** to retrieve the value for **-tagAt::**. If the delegate or dataDelegate respond to this message, that value is returned. If neither responds to the message, or if the table is not lazy, the cell is retrieved via **-cellAt::**. If the cell responds to the **-tag** message, that value is returned; otherwise zero is returned. This method gives lazy tables the opportunity to provide the information content of cells without the overhead of preparing and formatting a cell. You should implement this method in your delegate or dataDelegate if you have any slots that contain `tag` values.

See also: -cellAt::, -tagAt::, -isLazy, -setLazy:

tableScroll:titleAt::

- (char const*) **tableScroll:**(MiscTableScroll*)*scroll*
  titleAt:(int)*row* **:**(int)*col*

Lazy tables send this message to the **delegate** and then the **dataDelegate** to retrieve the value for **-titleAt::**. If the delegate or dataDelegate respond to this message, that value is returned. If neither responds to the message, or if the table is not lazy, the cell is retrieved via **-cellAt::**. If the cell responds to the **-title** message, that value is returned; otherwise zero is returned. This method gives lazy tables the opportunity to provide the information content of cells without the overhead of preparing and formatting a cell. You should implement this method in your delegate or dataDelegate if you have any slots that contain `title` values.

See also: -cellAt::, -stringValueAt::, -isLazy, -setLazy:, -title (ButtonCell)

tableScroll:readSelectionFromPasteboard:

- **tableScroll:**(MiscTableScroll*)*scroll*

readSelectionFromPasteboard:*pboard*

If the delegate responds to this message, the delegate has the opportunity to take over the process of reading data from the pasteboard. If the delegate does not respond to this message, the dataDelegate is tried.

See also: -readSelectionFromPasteboard:

tableScroll:retireAt::

- **tableScroll:**(MiscTableScroll*)*scroll*
  retireAt:(int)*row* **:**(int)*col*

If the cell responds to this message, the cell has the opportunity to perform special handling when it is being retired from active use and returned to the cache.

See also: -retireCell:at::

tableScroll:retireCell:at::

- **tableScroll:**(MiscTableScroll*)*scroll*
  retireCell:*cell*
  at:(int)*row* **:**(int)*col*

If the delegate responds to this message, the delegate has the opportunity to perform special handling of cells that are being retired to the cache. If the delegate does not respond, the dataDelegate is tried. If the dataDelegate does not respond either, the cell itself is tried (with -tableScroll:retireAt::).

See also: -retireCell:at::, -tableScroll:retireAt:: (delegate method)

tableScroll:reviveAt::

- **tableScroll:**(MiscTableScroll*)*scroll*
  reviveAt:(int)*row* **:**(int)*col*

If the cell responds to this message, the cell has the opportunity to perform special handling when it is being brought into use for the first time, or is being retrieved from the cache for reuse.

See also: **-reviveCell:At::**

tableScroll:reviveCell:at::

- **tableScroll:**(MiscTableScroll*)*scroll*
  reviveCell:*cell*
  at:(int)*row* **:**(int)*col*

If the delegate responds to this message, the delegate has the opportunity to perform special handling of cells that are being brought into use for the first time, or are being retrieved from the cache for reuse. If the delegate does not respond, the dataDelegate is tried. If the dataDelegate does not respond either, the cell itself is tried (with -tableScroll:reviveAt::).

See also: **-reviveCell:at::**

tableScroll:textColorChangedTo:

- **tableScroll:**(MiscTableScroll*)*scroll*

textColorChangedTo:(NXColor)*newColor*

This message is sent to the **delegate** and then the **dataDelegate** when the MiscTableScroll receives a `-setTextColor:` message that actually changes the background color.

See also: -setTextColor:

tableScroll:validRequestorForSendType:andReturnType:

- **tableScroll:**(MiscTableScroll*)*scroll*
  validRequestorForSendType:(NXAtom)*t_write*
  andReturnType:(NXAtom)*t_read*

If the delegate responds to this message, the delegate has the opportunity to interact with the services system using different combinations of send and return types than the MiscTableScroll object alone normally does. If the delegate does not respond, the dataDelegate is tried.

See also: -validRequestorForSendType:andReturnType:

tableScroll:writePboard:type:toStream:

- (void)**tableScroll:**(MiscTableScroll*)*scroll*
  writePboard:*pboard*
  type:(NXAtom)*type*
  toStream:(NXStream*)*stream*

If the delegate responds to this message, the delegate has the opportunity to write new datatypes to the pasteboard, or change the way that the builtin types are written. If the delegate does not respond, the dataDelegate is tried.

See also: -writePboard:type:toStream:

tableScroll:writeSelectionToPasteboard:types:

- (BOOL)**tableScroll:**(MiscTableScroll*)*scroll*
  writeSelectionToPasteboard:*pboard*
  types:(NXAtom*)*types*

If the delegate responds to this message, the delegate has the opportunity to completely take over the writing of data to the pasteboard. If the delegate does not respond, the dataDelegate is tried.

See also: -writeSelectionToPasteboard:types:

tableScrollRegisterServicesTypes:

- (void)**tableScrollRegisterServicesTypes:**(MiscTableScroll*)*scroll*

If the delegate responds to this message, the delegate has the opportunity to register different datatypes with the services system. If the delegate does not respond, the dataDelegate is tried.

See also: -registerServicesTypes:

Constants and Defined Types

```
typedef int MiscPixels;
```

```
typedef int MiscCoord_V;       // Visual coordinate.
typedef int MiscCoord_P;       // Physical coordinate.

#define MISC_MIN_PIXELS_SIZE   ((MiscPixels) 10)
#define MISC_MAX_PIXELS_SIZE   ((MiscPixels) 0x7FFF0000)

typedef enum
    {
    MISC_COL_BORDER,
    MISC_ROW_BORDER
    } MiscBorderType;

#define MISC_MAX_BORDER        MISC_ROW_BORDER


typedef enum
    {
    MISC_NO_TITLE,             // No titles on row/col cells.
    MISC_NUMBER_TITLE,         // Titles are sequential numbers.
    MISC_ALPHA_TITLE,          // Titles are sequential alphabetics...
    MISC_CUSTOM_TITLE,         // Titles are user-supplied strings...
```

```
    MISC_DELEGATE_TITLE        // Ask the delegate for titles.
    } MiscTableTitleMode;

#define MISC_MAX_TITLE         MISC_DELEGATE_TITLE


typedef enum
    {
    MISC_LIST_MODE,
    MISC_RADIO_MODE,
    MISC_HIGHLIGHT_MODE
    } MiscSelectionMode;

#define MISC_MAX_MODE          MISC_HIGHLIGHT_MODE


typedef enum
    {
    MISC_TABLE_CELL_TEXT,
    MISC_TABLE_CELL_ICON,
    MISC_TABLE_CELL_BUTTON,
```

```
    MISC_TABLE_CELL_CALLBACK
    } MiscTableCellStyle;

#define MISC_TABLE_CELL_MAX   MISC_TABLE_CELL_CALLBACK


#define MISC_SIZING_SPRINGY_BIT (1 << 0) // Adjusts for global limits.
#define MISC_SIZING_DATA_BIT   (1 << 1) // Expands to size of data.
#define MISC_SIZING_USER_BIT   (1 << 2) // User can resize.


typedef enum
    {
    MISC_NUSER_NDATA_NSPRINGY_SIZING,
    MISC_NUSER_NDATA_SPRINGY_SIZING,
    MISC_NUSER_DATA_NSPRINGY_SIZING,
    MISC_NUSER_DATA_SPRINGY_SIZING,
    MISC_USER_NDATA_NSPRINGY_SIZING,
    MISC_USER_NDATA_SPRINGY_SIZING,
    MISC_USER_DATA_NSPRINGY_SIZING,
    MISC_USER_DATA_SPRINGY_SIZING,
```

```
    } MiscTableSizing;

#define MISC_MAX_SIZING       MISC_USER_DATA_SPRINGY_SIZING


typedef enum
    {
    MISC_SORT_ASCENDING,
    MISC_SORT_DESCENDING
    } MiscSortDirection;

#define MISC_SORT_DIR_MAX     MISC_SORT_DESCENDING


typedef enum                          // Selector used to get data:
    {
    MISC_SORT_STRING_CASE_INSENSITIVE, // -stringValue
    MISC_SORT_STRING_CASE_SENSITIVE,   // -stringValue
    MISC_SORT_INT,                     // -intValue
    MISC_SORT_UNSIGNED_INT,            // -intValue
    MISC_SORT_TAG,                     // -tag
```

```
    MISC_SORT_UNSIGNED_TAG,            // -tag
    MISC_SORT_FLOAT,                   // -floatValue
    MISC_SORT_DOUBLE,                  // -doubleValue
    MISC_SORT_SKIP,                    // Don't compare cells in this slot.
    MISC_SORT_TITLE_CASE_INSENSITIVE,  // -title
    MISC_SORT_TITLE_CASE_SENSITIVE,    // -title
    MISC_SORT_STATE,                   // -state
    MISC_SORT_UNSIGNED_STATE,          // -state
    } MiscSortType;

#define MISC_SORT_TYPE_MAX    MISC_SORT_UNSIGNED_STATE


typedef int (*MiscCompareEntryFunc)
                    ( int r1, int c1, int r2, int c2, MiscSlotSortInfo* info );
typedef int (*MiscCompareSlotFunc)
                    ( int slot1, int slot2, MiscSlotSortInfo* );

extern MiscCompareSlotFunc  MiscDefaultCompareSlotFunc;
```

```
struct MiscEntrySortInfo
        {
        int slot;
        int ascending;
        MiscSortType sort_type;
        MiscCompareEntryFunc compare_func;
        };


struct MiscSlotSortInfo
        {
        MiscTableScroll* table_scroll;
        NXZone* zone;
        MiscBorderType border_type;
        int num_entries;
        MiscEntrySortInfo const* entry_info;
        BOOL need_copy;
        char* buff;
        int buff_size;
        };
```