# MiscValueCell

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

**Declared In:** MiscValueCell.h

## Class Description

A **MiscValueCell** adds ªupº and ªdownº arrows to a TextFieldCell. The primary advantage of this class is that it allows the user to scroll through a list of choices yet sends far fewer action messages than the equivalent combination of individual controls, sending the target the action only after the button is released. A **MiscStringArray** or a **StringList** (found as a MiniExample) can be attached to provide text strings that get shown instead of numbers.

A valid string list object responds to:

-(char *) **stringAt:**(int)item;
-(unsigned int) **count**;

The value sent to **stringAt:** will be in the range 0 - **count**. When the **stringList** instance variable is set, the ranges as set in IB (or wherever) are ignored and the string list defines the range of its values. Of course, **stringAt:** can create its return value any way it likes.

## Instance Variables

```
id          idUp;
id          idDown;
id          stringList;
double      actualValue;
double      limitMinValue;
double      limitMaxValue;
double      limitMinBound;
double      limitMaxBound;
double      sizeStep;
double      sizeAltStep;
BOOL        fExpandLow;
BOOL        fExpandHigh;
BOOL        fLoop;
BOOL        fValidRect;
int         fAltKeyDown;
NXRect      rectFrame;
```

| | |
|---|---|
| idUp | The ButtonCell for the up arrow button. |
| idDown | The ButtonCell for the down arrow button. |
| stringList | An object to provide strings in place of values. |
| actualValue | The numeric value since the display may be a string. |
| limitMinValue | The absolute lower limit of allowable values. |
| limitMaxValue | The absolute upper limit of allowable values. |
| limitMinBound | The adjustable lower limit for stopping the value when using the buttons. |
| limitMaxBound | The adjustable upper limit for stopping the value when using the buttons. |
| sizeStep | The amount to change the value by when a button is clicked. |
| sizeAltStep | The amount to use when the Alternate key is down during the click. |
| fExpandLow | Declares that limitMinBound is actually being used. |
| fExpandHigh | Declares that limitMaxBound is actually being used. |
| fLoop | Used to keep things sane in the setXXXValue: methods. (Stops endless looping). |
| fValidRect | Used by the drawing methods to keep track of the width adjustment. |
| fAltKeyDown | Set from the triggering event so the actions for the buttons can know. |

| | |
|---|---|
| rectFrame | Used by the drawing methods to keep track of the width adjustment. |

## Method Types

| | |
|---|---|
| Initializing | ± initTextCell:<br>± copyFromZone: |
| Changing a MiscValueCell | ± setBezeled:<br>± setBordered:<br>± setEditable: |
| Manipulating a MiscValueCell | ± setMinValue:<br>± setMaxValue:<br>± setMinBoundary:<br>± setMaxBoundary:<br>± setStepSize:<br>± setAltStepSize:<br>± setExpandMin:<br>± setExpandMax:<br>± setStringList: |
| Querying values | ± minValue |

| | |
|---|---|
| | ± maxValue<br>± minBoundary<br>± maxBoundary<br>± stepSize<br>± altStepSize<br>± expandMin<br>± expandMax<br>± stringList |
| Archiving | ± read:<br>± write:<br>± awake<br>± awakeFromNib |

## Instance Methods

**altStepSize**

±(double) **altStepSize**

Returns the amount the value will change when a button is clicked while Alternate is held down.

See also: ± **setAltStepSize:, ± setStepSize:, ± stepSize**

**awake**
**± awake**

Sends a message so the value will be properly displayed. Returns **self**.

See also: **-awakeFromNib**

**awakeFromNib**
± **awakeFromNib**

Sends self an awake message. Returns **self**.

See also: ± **awake**

**copyFromZone:**
± **copyFromZone:**(NXZone *)*zone*

Copies the MiscValueCell and also copies the ButtonCells inside. Returns the new object.

**expandMax**
±(BOOL) **expandMax**

Returns YES if **maxBoundary** is in effect stopping the arrow buttons from reaching **maxValue**.

See also: **± expandMin, ± maxBoundary, ± maxValue, ± minBoundary, ± minValue, ± setExpandMax, ±**

**€setExpandMin, ± setMaxBoundary, ± setMaxValue, ± setMinBoundary, ± setMinValue**

**expandMin**

±(BOOL) **expandMin**

Returns YES if **minBoundary** is in effect stopping the arrow buttons from reaching **minValue**.

See also: **± expandMax, ± maxBoundary, ± maxValue, ± minBoundary, ± minValue, ± setExpandMax, ± €setExpandMin, ± setMaxBoundary, ± setMaxValue, ± setMinBoundary, ± setMinValue**

**initTextCell**

± **initTextCell:**(const char *)*sz*

Initializes the receiver to contain the string *sz*. The hard limits are set to 0 and 100 and the soft limits are not active. the step size is set to 1 and the Alternate step size is set to 10. The Cell has the Bezeled style, is Editable and is not Continuous. Returns **self**.

**maxBoundary**

±(double) **maxBoundary**

Returns the current limit that the value will stop at when the up arrow button is being used to change the value. This limit can be changed by entering a larger value using the keyboard. This limit will be set the highest value entered that is still within the bounds of **maxValue**. It cannot be reduced except programmatically.

See also: **± expandMax, ± expandMin, ± maxValue, ± minBoundary, ± minValue, ± setExpandMax, ± €setExpandMin, ± setMaxBoundary, ± setMaxValue, ± setMinBoundary, ± setMinValue**

**maxValue**

±(double) **maxValue**

Returns the highest value this field is allowed to reach. This limit cannot be exceeded in any way. **maxBoundary** can match this value and then have no effect.

See also: **± expandMax, ± expandMin, ± maxBoundary, ± minBoundary, ± minValue, ± setExpandMax, ± €setExpandMin, ± setMaxBoundary, ± setMaxValue, ± setMinBoundary, ± setMinValue**

**minBoundary**

±(double) **minBoundary**

Returns the current limit that the value will stop at when the down arrow button is being used to change the value. This limit can be changed by entering a smaller value using the keyboard. This limit will be set the lowest value entered that is still within the bounds of **minValue**. It cannot be increased except programmatically.

See also: **± expandMax, ± expandMin, ± maxBoundary, ± maxValue, ± minValue, ± setExpandMax, ± €setExpandMin, ± setMaxBoundary, ± setMaxValue, ± setMinBoundary, ± setMinValue**

**minValue**

±(double) **minValue**

Returns the lowest value this field is allowed to reach. This value cannot be exceeded in any way. **minBoundary** can match this value and then have no effect.

See also: **± expandMax, ± expandMin, ± maxBoundary, ± maxValue, ± minBoundary, ± setExpandMax, ± €setExpandMin, ± setMaxBoundary, ± setMaxValue, ± setMinBoundary, ± setMinValue**

**read:**

± **read:**(NXTypedStream *)*stream*

Reads a MiscValueCell from *stream*. This also reads in the ButtonCells used as subcells. Returns **self**.

See also: ± **write:**

**setAltStepSize:**

± **setAltStepSize:**(double)*size*

Sets the amount the value will change when a button is clicked while Alternate is being held down. Returns **self**.

See also: ± **altStepSize:, ± setStepSize:, ± stepSize**

**setBezeled:**

± **setBezeled:**(BOOL)*flag*

if *flag* is YES, sets both the TextFieldCell and the ButtonCell subcells to have the (standard) bezeled frames. If *flag* is NO, the TextFieldCell is set depending on the state of the **bordered** attribute and the ButtonCells and set to have no frame. Returns **self**.

See also: ± **setBordered:**

**setBordered:**

± **setBordered:**(BOOL)*flag*

If *flag* is YES, the TextFieldCell gets a black frame, otherwise it has no frame at all. The ButtonCells are set to have no frames regardless of *flag* because they do not have a bordered-not-bezeled style. Returns **self**.

See also: ± **setBezeled:**

**setEditable:**

± **setEditable:**(BOOL)*flag*

If *flag* is YES, sets the TextFieldCell to normal editable mode. If *flag* is NO, sets non-editable, but also sends **setSelectable:YES** so the buttons will still work. Returns **self**.

**setExpandMax:**

± **setExpandMax:**(BOOL)*flag*

If *flag* is YES then **maxBoundary** has an effect on the upper limit of the field value when it's adjusted with the arrow buttons. Returns **self**.

See also: **± expandMax, ± expandMin, ± maxBoundary, ± maxValue, ± minBoundary, ± minValue, ± €setExpandMin, ± setMaxBoundary, ± setMaxValue, ± setMinBoundary, ± setMinValue**

**setExpandMin:**

± **setExpandMin:**(BOOL)*flag*

If *flag* is YES then **minBoundary** has an effect on the lower limit of the field value when it's adjusted with the arrow buttons. Returns **self**.

See also: **± expandMax, ± expandMin, ± maxBoundary, ± maxValue, ± minBoundary, ± minValue, ± €setExpandMax, ± setMaxBoundary, ± setMaxValue, ± setMinBoundary, ± setMinValue**

**setMaxBoundary**

± **setMaxBoundary:**(double)*value*

Sets the highest value the field will allow when using the arrow buttons to adjust the value. This value can be exceeded and altered by typing a value greater than this limit. This value will be adjusted to match the entered amount with an additional limitation that it will stop at **maxValue**. Returns **self**.

See also: **± expandMax, ± expandMin, ± maxBoundary, ± maxValue, ± minBoundary, ± minValue, ± €setExpandMax, ± setExpandMin, ± setMaxValue, ± setMinBoundary, ± setMinValue**

**setMaxValue**

± **setMaxValue:**(double)*value*

Sets the highest value the field will allow. There is no way to exceed this limit from the interface. **maxBoundary** will stop expanding when it gets to this value. Returns **self**.

See also: **± expandMax, ± expandMin, ± maxBoundary, ± maxValue, ± minBoundary, ± minValue, ± €setExpandMax, ± setExpandMin, ± setMaxBoundary, ± setMinBoundary, ± setMinValue**

**setMinBoundary**

± **setMinBoundary:**(double)*value*

Sets the lowest value the field will allow when using the arrow buttons to adjust the value. This value can be exceeded and altered by typing a value lower than this limit. This value will be adjusted to match the entered amount with an additional limitation that it will stop at **minValue**. Returns **self**.

See also: **± expandMax, ± expandMin, ± maxBoundary, ± maxValue, ± minBoundary, ± minValue, ± €setExpandMax, ± setExpandMin, ± setMaxBoundary, ± setMaxValue, ± setMinValue**

**setMinValue**

± **setMinValue:**(double)*value*

Sets the lowest value the field will allow. There is no way to exceed this limit from the interface. **minBoundary**

will stop expanding when it gets to this value. Returns **self**.

See also: **± expandMax, ± expandMin, ± maxBoundary, ± maxValue, ± minBoundary, ± minValue, ± €setExpandMax, ± setExpandMin, ± setMaxBoundary, ± setMaxValue, ± setMinBoundary**

**setStepSize:**

± **setStepSize:**(double)*size*

Sets the amount the value will change when a button is clicked. Returns **self**.

See also: ± **altStepSize:, ± setAltStepSize:, ± stepSize**

**setStringList:**

± **setStringList:***anObject*

Sets the object to be asked for display strings. *anObject* should respond to **stringAt:** and **count** messages. When a string list is set the range limits are all ignored and the cell is set to non-editable. The value of the cell will only fit within a range defined by *anObject's* **count** method. Returns **self**.

*anObject's* **stringAt:** method should take an **int** as its argument and return a **char *** with the correct string to be displayed for the given value. The value of the argument will range from 0 to **count** - 1. This arrangement has been created to allow the MiscValueCell to be connected to a MiscStringArray or a StringList (found in the MiniExamples) object in Interface Builder so lists may be created, displayed and dealt with with no additional code.

See also: ± **stringList**

**stepSize**
± (double) **stepSize**

Returns the amount the value will change when a button is clicked.

See also: ± **altStepSize:, ± setAltStepSize:, ± setStepSize:**

**stringList**
± **stringList**

Returns the object that is taking the task of providing the display strings.

See also: ± **setStringList:**

**write:**
± **write:**(NXTypedStream *)*stream*

Writes the MiscValueCell and its ButtonCells to *stream*. The images in the ButtonCells only get archived once so redundancy is kept down. Returns **self**.

See also: ± **read:**