NSRuleEditor Class Reference

Cocoa > User Experience

2008-02-08

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Simultaneously published in the United States and Canada.

Contents

NSRuleEditor Class Reference

Inherits from	NSControl : NSView : NSResponder : NSObject
Conforms to	NSAnimatablePropertyContainer (NSView) NSCoding (NSResponder) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.5 and later.
Declared in	NSRuleEditor.h
Companion guides	Control and Cell Programming Topics for Cocoa Predicate Programming Guide

Overview

An `NSRuleEditor` object is a view that allows the user to create and configure a list of options. The view has a delegate which offers a tree of choices to the view. The choices are presented by the view to the user as a row of popup buttons, static text fields, and custom views. Each row in the list represents a particular path down the tree of choices.

`NSRuleEditor` exposes one binding, `rows`. You can bind `rows` to an ordered collection (such as an instance of `NSMutableArray`). Each object in the collection should have the following properties:

@"rowType"
: An integer representing the type of the row (`NSRuleEditorRowType`).

@"subrows"
: An ordered to-many relation (such as an instance of `NSMutableArray`) containing the directly nested subrows for the given row.

@"displayValues"
: An ordered to-many relation containing the display values for the row.

@"criteria"
: An ordered to-many relation containing the criteria for the row.

Tasks

Configuring a Rule Editor

- `delegate` (page 10)
  Returns the receiver's delegate.
- `setDelegate:` (page 21)
  Sets the receiver's delegate.
- `isEditable` (page 13)
  Returns a Boolean value that indicates whether the receiver is editable.
- `setEditable:` (page 22)
  Sets whether the receiver is editable.
- `nestingMode` (page 13)
  Returns the nesting mode for the receiver.
- `setNestingMode:` (page 23)
  Sets the nesting mode for the receiver.
- `canRemoveAllRows` (page 9)
  Returns a Boolean value that indicates whether all the rows can be removed.
- `setCanRemoveAllRows:` (page 19)
  Sets whether all the rows can be removed.
- `rowHeight` (page 17)
  Returns the row height for the receiver.
- `setRowHeight:` (page 24)
  Sets the row height for the receiver.

Working with Formatting

- `formattingDictionary` (page 12)
  Returns the formatting dictionary for the receiver.
- `setFormattingDictionary:` (page 22)
  Sets the formatting dictionary for the receiver.
- `formattingStringsFilename` (page 12)
  Returns the name of the strings file for the receiver.
- `setFormattingStringsFilename:` (page 23)
  Sets the name of the strings file used for formatting.

Providing Data

- `reloadCriteria` (page 15)
  Instructs the receiver to refetch criteria from its delegate.
- `setCriteria:andDisplayValues:forRowAtIndex:` (page 20)
  Modifies the row at a given index to contain the given items and values.

- `criteriaForRow:` (page 9)
  Returns the currently chosen items for a given row.
- `ruleEditor:child:forCriterion:withRowType:` (page 26) *delegate method*
  Returns the child of a given item at a given index.
- `displayValuesForRow:` (page 11)
  Returns the chosen values for a given row.
- `ruleEditor:displayValueForCriterion:inRow:` (page 27) *delegate method*
  Returns the value for a given criterion.
- `ruleEditor:numberOfChildrenForCriterion:withRowType:` (page 28) *delegate method*
  Returns the number of child items of a given criterion or row type.
- `ruleEditor:predicatePartsForCriterion:withDisplayValue:inRow:` (page 28) *delegate method*
  Returns a dictionary representing the parts of the predicate determined by the given criterion and value.

Obtaining Row Information

- `numberOfRows` (page 14)
  Returns the number of rows in the receiver.
- `parentRowForRow:` (page 14)
  Returns the index of the parent of a given row.
- `rowForDisplayValue:` (page 17)
  Returns the index of the row containing a given value.
- `rowTypeForRow:` (page 18)
  Returns the type of a given row.
- `subrowIndexesForRow:` (page 25)
  Returns the immediate subrows of a given row.

Working with the Selection

- `selectedRowIndexes` (page 19)
  Returns the indexes of the receiver's selected rows.
- `selectRowIndexes:byExtendingSelection:` (page 19)
  Sets in the receiver the indexes of rows that are selected.

Manipulating Rows

- `addRow:` (page 9)
  Adds a row to the receiver.
- `insertRowAtIndex:withType:asSubrowOfRow:animate:` (page 12)
  Adds a new row of a given type at a given location.
- `removeRowAtIndex:` (page 16)
  Removes the row at a given index.

- `removeRowsAtIndexes:includeSubrows:` (page 16)
  Removes the rows at given indexes.
- `ruleEditorRowsDidChange:` (page 29) *delegate method*
  Notifies the receiver that a rule editor's rows changed.

Working with Predicates

Note that there is a subclass of `NSRuleEditor`, `NSPredicateEditor`, that is designed to facilitate creation of predicates.

- `predicate` (page 14)
  Returns the predicate for the receiver.
- `reloadPredicate` (page 15)
  Instructs the receiver to regenerate its predicate by invoking the corresponding delegate method.
- `predicateForRow:` (page 15)
  Returns the predicate for a given row.

Supporting Bindings

`NSRuleEditor` is key-value coding and key-value observing compliant for the keys described in this section. If you override any of these methods, you must ensure that you invoke the relevant change notification methods to maintain key-value observing compliance. For more information, see *Key-Value Coding Programming Guide* and *Key-Value Observing Programming Guide*. For more about Cocoa bindings, see *Cocoa Bindings Programming Topics*.

- `rowClass` (page 17)
  Returns the class used to create a new row in the "rows" binding.
- `setRowClass:` (page 24)
  Sets the class to use to create a new row in the "rows" binding.
- `rowTypeKeyPath` (page 18)
  Returns the key path for the row type.
- `setRowTypeKeyPath:` (page 24)
  Sets the key path for the row type.
- `subrowsKeyPath` (page 26)
  The key path for the subrows.
- `setSubrowsKeyPath:` (page 25)
  Set the key path for the subrows.
- `criteriaKeyPath` (page 10)
  Returns the criteria key path.
- `setCriteriaKeyPath:` (page 21)
  Sets the criteria key path.
- `displayValuesKeyPath` (page 11)
  Returns the display values key path.
- `setDisplayValuesKeyPath:` (page 21)
  Sets the display values key path.

Overriding ViewDidMoveToWindow

`- viewDidMoveToWindow` (page 26)
Overrides the `NSView` implementation.

Instance Methods

addRow:

Adds a row to the receiver.

```
- (void)addRow:(id)sender
```

Parameters

sender
Typically the object that sent the message.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSRuleEditor.h`

canRemoveAllRows

Returns a Boolean value that indicates whether all the rows can be removed.

```
- (BOOL)canRemoveAllRows
```

Return Value
`YES` if all the rows can be removed, otherwise `NO`.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- setCanRemoveAllRows:` (page 19)

Declared In
`NSRuleEditor.h`

criteriaForRow:

Returns the currently chosen items for a given row.

```
- (NSArray *)criteriaForRow:(NSInteger)row
```

Parameters

row

The index of a row in the receiver.

Return Value
The currently chosen items for row *row*.

Discussion
The items returned are the same as those returned from the delegate method ruleEditor:child:forCriterion:withRowType: (page 26).

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSRuleEditor.h

criteriaKeyPath

Returns the criteria key path.

```
- (NSString *)criteriaKeyPath
```

Return Value
The criteria key path.

Discussion
The default value is `@"criteria"`.

The key path is used to get the criteria for a row in the "rows" binding. The criteria objects are what the delegate returns from ruleEditor:child:forCriterion:withRowType: (page 26). The corresponding property should be an ordered to-many relationship.

Availability
Available in Mac OS X v10.5 and later.

See Also
- setCriteriaKeyPath: (page 21)

Declared In
NSRuleEditor.h

delegate

Returns the receiver's delegate.

```
- (id)delegate
```

Return Value
The receiver's delegate.

Availability
Available in Mac OS X v10.5 and later.

See Also
- setDelegate: (page 21)

Declared In
NSRuleEditor.h

displayValuesForRow:

Returns the chosen values for a given row.

```
- (NSArray *)displayValuesForRow:(NSInteger)row
```

Parameters

row
The index of a row in the receiver.

Return Value
The chosen values (strings, views, or menu items) for row *row*.

Discussion
The values returned are the same as those returned from the delegate method ruleEditor:displayValueForCriterion:inRow: (page 27).

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSRuleEditor.h

displayValuesKeyPath

Returns the display values key path.

```
- (NSString *)displayValuesKeyPath
```

Return Value
The display values key path.

Discussion
The default is @"displayValues".

The key path is used to get the display values for a row in the "rows" binding. The display values are what the delegate returns from ruleEditor:displayValueForCriterion:inRow: (page 27). The corresponding property should be an ordered to-many relationship.

Availability
Available in Mac OS X v10.5 and later.

See Also
- setDisplayValuesKeyPath: (page 21)

Declared In
NSRuleEditor.h

formattingDictionary

Returns the formatting dictionary for the receiver.

```
- (NSDictionary *)formattingDictionary
```

Return Value
The formatting dictionary for the receiver.

Availability
Available in Mac OS X v10.5 and later.

See Also
- setFormattingStringsFilename: (page 23)

Declared In
NSRuleEditor.h

formattingStringsFilename

Returns the name of the strings file for the receiver.

```
- (NSString *)formattingStringsFilename
```

Return Value
The name of the strings file for the receiver.

Availability
Available in Mac OS X v10.5 and later.

See Also
- setFormattingStringsFilename: (page 23)

Declared In
NSRuleEditor.h

insertRowAtIndex:withType:asSubrowOfRow:animate:

Adds a new row of a given type at a given location.

```
- (void)insertRowAtIndex:(NSInteger)rowIndex withType:(NSRuleEditorRowType)rowType
    asSubrowOfRow:(NSInteger)parentRow animate:(BOOL)shouldAnimate
```

Parameters

rowIndex
The index at which the new row should be inserted. *rowIndex* must be greater than *parentRow*, and much specify a row that does not fall amongst the children of some other parent.

rowType
The type of the new row.

parentRow
The index of the row of which the new row is a child. Pass -1 to indicate that the new row should be a root row.

shouldAnimate
YES if creation of the new row should be animated, otherwise NO.

Special Considerations

Important: If *parentRow* is greater than or equal to *rowIndex*, or if *rowIndex* would fall amongst the children of some other parent, or if the nesting mode forbids this configuration, an NSInvalidArgumentException is raised.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSRuleEditor.h

isEditable

Returns a Boolean value that indicates whether the receiver is editable.

```
- (BOOL)isEditable
```

Return Value
YES if the receiver is editable, otherwise NO.

Discussion
The default is YES.

Availability
Available in Mac OS X v10.5 and later.

See Also
- setEditable: (page 22)

Declared In
NSRuleEditor.h

nestingMode

Returns the nesting mode for the receiver.

```
- (NSRuleEditorNestingMode)nestingMode
```

Return Value
The nesting mode for the receiver.

Availability
Available in Mac OS X v10.5 and later.

See Also
- setNestingMode: (page 23)

Declared In
NSRuleEditor.h

numberOfRows

Returns the number of rows in the receiver.

```
- (NSInteger)numberOfRows
```

Return Value
The number of rows in the receiver.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSRuleEditor.h`

parentRowForRow:

Returns the index of the parent of a given row.

```
- (NSInteger)parentRowForRow:(NSInteger)rowIndex
```

Parameters
rowIndex
The index of a row in the receiver.

Important: Raises an `NSRangeException` if *rowIndex* is less than `0` or greater than or equal to the number of rows.

Return Value
The index of the parent of the row at *rowIndex*. If the row at *rowIndex* is a root row, returns `-1`.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSRuleEditor.h`

predicate

Returns the predicate for the receiver.

```
- (NSPredicate *)predicate
```

Return Value
If the delegate implements ruleEditor:predicatePartsForCriterion:withDisplayValue:inRow: (page 28), the predicate for the receiver. If the delegate does implement ruleEditor:predicatePartsForCriterion:withDisplayValue:inRow:, or if the delegate does not return enough parts to construct a full predicate, returns `nil`.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSRuleEditor.h`

predicateForRow:

Returns the predicate for a given row.

```
- (NSPredicate *)predicateForRow:(NSInteger)row
```

Parameters

row
The index of a row in the receiver.

Return Value
The predicate for the row at *row*.

Discussion
You should rarely have a need to call this directly, but you can override this method in a subclass to perform specialized predicate handling for certain criteria or display values.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSRuleEditor.h`

reloadCriteria

Instructs the receiver to refetch criteria from its delegate.

```
- (void)reloadCriteria
```

Discussion
You can use this method to indicate that the available criteria may have changed and should be refetched from the delegate and the popups recalculated. If any item in a given row is “orphaned” (that is, is no longer reported as a child of its previous parent), its criteria and display values are set to valid choices.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSRuleEditor.h`

reloadPredicate

Instructs the receiver to regenerate its predicate by invoking the corresponding delegate method.

```
- (void)reloadPredicate
```

Discussion
You typically invoke this method because something has changed (for example, a view's value).

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSRuleEditor.h`

removeRowAtIndex:

Removes the row at a given index.

```
- (void)removeRowAtIndex:(NSInteger)rowIndex
```

Parameters

rowIndex
The index of a row in the receiver.

Important: Raises an `NSRangeException` if *rowIndex* is less than `0` or greater than or equal to the number of rows.

Discussion
Any subrows of the deleted row are adopted by the parent of the deleted row, or are made root rows.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSRuleEditor.h`

removeRowsAtIndexes:includeSubrows:

Removes the rows at given indexes.

```
- (void)removeRowsAtIndexes:(NSIndexSet *)rowIndexes
    includeSubrows:(BOOL)includeSubrows
```

Parameters

rowIndexes
Indexes of one or more rows in the receiver.

Important: Raises an `NSRangeException` if any index in *rowIndexes* is less than `0` or greater than or equal to the number of rows.

includeSubrows
If `YES`, then sub-rows of deleted rows are also deleted; if `NO`, then each sub-row is adopted by its first non-deleted ancestor, or becomes a root row.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSRuleEditor.h`

rowClass

Returns the class used to create a new row in the “rows” binding.

```
- (Class)rowClass
```

Return Value
The class used to create a new row in the "rows" binding.

Discussion
By default, this is `NSMutableDictionary`.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- setRowClass:` (page 24)

Declared In
`NSRuleEditor.h`

rowForDisplayValue:

Returns the index of the row containing a given value.

```
- (NSInteger)rowForDisplayValue:(id)displayValue
```

Parameters
displayValue
The display value (string, view, or menu item) of an item in the receiver. This value must not be `nil`.

Important: Raises `NSInvalidArgumentException` if *displayValue* is `nil`.

Return Value
The index of the row containing *displayValue*, or `NSNotFound`.

Discussion
This method searches each row via pointer equality for the given display value, which may be present as an alternative in a popup menu for that row.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSRuleEditor.h`

rowHeight

Returns the row height for the receiver.

```
- (CGFloat)rowHeight
```

Return Value
The row height for the receiver.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- setRowHeight:` (page 24)

Declared In
`NSRuleEditor.h`

rowTypeForRow:

Returns the type of a given row.

```
- (NSRuleEditorRowType)rowTypeForRow:(NSInteger)rowIndex
```

Parameters
rowIndex
The index of a row in the receiver.

Important: Raises an `NSRangeException` if *rowIndex* is less than `0` or greater than or equal to the number of rows.

Return Value
The type of the row at *rowIndex*.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSRuleEditor.h`

rowTypeKeyPath

Returns the key path for the row type.

```
- (NSString *)rowTypeKeyPath
```

Return Value
The key path for the row type.

Discussion
The default value is `@"rowType"`.

The key path is used to get the row type in the "rows" binding. The corresponding property should be a number that specifies an `NSRuleEditorRowType` value (see "Row Types" (page 30)).

Availability
Available in Mac OS X v10.5 and later.

See Also
- `setRowTypeKeyPath:` (page 24)

Declared In
`NSRuleEditor.h`

selectedRowIndexes

Returns the indexes of the receiver's selected rows.

```
- (NSIndexSet *)selectedRowIndexes
```

Return Value
The indexes of the receiver's selected rows.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSRuleEditor.h`

selectRowIndexes:byExtendingSelection:

Sets in the receiver the indexes of rows that are selected.

```
- (void)selectRowIndexes:(NSIndexSet *)indexes byExtendingSelection:(BOOL)extend
```

Parameters
indexes
The indexes of rows in the receiver to select.

Important: Raises an `NSRangeException` if any index in *rowIndexes* is less than `0` or greater than or equal to the number of rows.

extend
If `NO`, the selected rows are specified by *indexes*. If `YES`, the rows indicated by *indexes* are added to the collection of already selected rows, providing multiple selection.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSRuleEditor.h`

setCanRemoveAllRows:

Sets whether all the rows can be removed.

```
- (void)setCanRemoveAllRows:(BOOL)val
```

Parameters

val

YES if all the rows can be removed, otherwise NO.

Availability
Available in Mac OS X v10.5 and later.

See Also
- canRemoveAllRows (page 9)

Declared In
NSRuleEditor.h

setCriteria:andDisplayValues:forRowAtIndex:

Modifies the row at a given index to contain the given items and values.

```
- (void)setCriteria:(NSArray *)criteria andDisplayValues:(NSArray *)values
    forRowAtIndex:(NSInteger)rowIndex
```

Parameters

criteria

The array of criteria for the row at *rowIndex*. Pass an empty array to force the receiver to query its delegate. This value must not be nil.

Important: Raises an NSInvalidArgumentException if *criteria* is nil.

values

The array of values for the row at *rowIndex*. Pass an empty array to force the receiver to query its delegate. This value must not be nil.

Important: Raises an NSInvalidArgumentException if *values* is nil.

rowIndex

The index of a row in the receiver.

Important: Raises an NSRangeException if *rowIndex* is equal to or larger than the number of rows, or less than 0.

Discussion
It is your responsibility to ensure that each item in the array is a child of the previous item, and that the first item is a root item for the row type. If the last item has child items, then the items array will be extended by querying the delegate for child items until a childless item is reached. If *values* contains fewer objects than the (possibly extended) criteria array, then the delegate is queried to construct the remaining display values. If you want the delegate to be queried for all the criteria or all the display values, pass empty arrays; do not pass nil.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSRuleEditor.h

setCriteriaKeyPath:

Sets the criteria key path.

```
- (void)setCriteriaKeyPath:(NSString *)keyPath
```

Parameters

keyPath
The criteria key path.

Discussion
The criteria key path is described in criteriaKeyPath (page 10).

Availability
Available in Mac OS X v10.5 and later.

See Also
- criteriaKeyPath (page 10)

Declared In
NSRuleEditor.h

setDelegate:

Sets the receiver's delegate.

```
- (void)setDelegate:(id)delegate
```

Parameters

delegate
The delegate for the receiver.

Availability
Available in Mac OS X v10.5 and later.

See Also
- delegate (page 10)

Declared In
NSRuleEditor.h

setDisplayValuesKeyPath:

Sets the display values key path.

```
- (void)setDisplayValuesKeyPath:(NSString *)keyPath
```

Parameters

keyPath

The display values key path.

Discussion

The display values key path is described in `displayValuesKeyPath` (page 11).

Availability

Available in Mac OS X v10.5 and later.

See Also

`- displayValuesKeyPath` (page 11)

Declared In

`NSRuleEditor.h`

setEditable:

Sets whether the receiver is editable.

```
- (void)setEditable:(BOOL)editable
```

Parameters

editable

`YES` if the receiver is editable, otherwise `NO`.

Availability

Available in Mac OS X v10.5 and later.

See Also

`- isEditable` (page 13)

Declared In

`NSRuleEditor.h`

setFormattingDictionary:

Sets the formatting dictionary for the receiver.

```
- (void)setFormattingDictionary:(NSDictionary *)dictionary
```

Parameters

dictionary

The formatting dictionary for the receiver.

Discussion

If you set the formatting dictionary with this method, it sets the current to formatting strings file name `nil` (see `formattingStringsFilename` (page 12)).

Availability

Available in Mac OS X v10.5 and later.

See Also

`- setFormattingStringsFilename:` (page 23)

Declared In
NSRuleEditor.h

setFormattingStringsFilename:

Sets the name of the strings file used for formatting.

```
- (void)setFormattingStringsFilename:(NSString *)stringsFilename
```

Parameters

stringsFilename
The name of the strings file for the receiver.

Discussion
NSRuleEditor looks for a strings file with the given name in the main bundle and (if appropriate) the bundle containing the nib file from which it was loaded. If it finds a strings file resource with the given name, NSRuleEditor loads it and sets it as the formatting dictionary for the receiver. You can obtain the resulting dictionary using formattingDictionary (page 12)].

If you set the formatting dictionary with setFormattingDictionary: (page 22), it sets the current to formatting strings file name nil (see formattingStringsFilename (page 12)).

Availability
Available in Mac OS X v10.5 and later.

See Also
- formattingDictionary (page 12)

Declared In
NSRuleEditor.h

setNestingMode:

Sets the nesting mode for the receiver.

```
- (void)setNestingMode:(NSRuleEditorNestingMode)mode
```

Parameters

mode
The nesting mode for the receiver.

Discussion
You typically set the nesting mode at view creation time and do not subsequently modify it. The default is NSRuleEditorNestingModeCompound.

Availability
Available in Mac OS X v10.5 and later.

See Also
- nestingMode (page 13)

Declared In
NSRuleEditor.h

setRowClass:

Sets the class to use to create a new row in the "rows" binding.

```
- (void)setRowClass:(Class)rowClass
```

Parameters

rowClass

The class to use to create a new row in the "rows" binding.

Availability
Available in Mac OS X v10.5 and later.

See Also
- rowClass (page 17)

Declared In
NSRuleEditor.h

setRowHeight:

Sets the row height for the receiver.

```
- (void)setRowHeight:(CGFloat)height
```

Parameters

height

The row height for the receiver.

Availability
Available in Mac OS X v10.5 and later.

See Also
- rowHeight (page 17)

Declared In
NSRuleEditor.h

setRowTypeKeyPath:

Sets the key path for the row type.

```
- (void)setRowTypeKeyPath:(NSString *)keyPath
```

Parameters

keyPath

The key path for the row type.

Discussion
The row type key path is described in rowTypeKeyPath (page 18).

Availability
Available in Mac OS X v10.5 and later.

See Also
- `rowTypeKeyPath` (page 18)

Declared In
`NSRuleEditor.h`

setSubrowsKeyPath:

Set the key path for the subrows.

```
- (void)setSubrowsKeyPath:(NSString *)keyPath
```

Parameters

keyPath
The key path for the subrows.

Discussion
The subrows key path is described in `subrowsKeyPath` (page 26).

Availability
Available in Mac OS X v10.5 and later.

See Also
- `subrowsKeyPath` (page 26)

Declared In
`NSRuleEditor.h`

subrowIndexesForRow:

Returns the immediate subrows of a given row.

```
- (NSIndexSet *)subrowIndexesForRow:(NSInteger)rowIndex
```

Parameters

rowIndex
The index of a row in the receiver, or `-1` to get the top-level rows.

Important: Raises an `NSRangeException` if *rowIndex* is less than `-1` or greater than or equal to the number of rows.

Return Value
The immediate subrows of the row at *rowIndex*.

Discussion
Rows are numbered starting at `0`.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSRuleEditor.h`

subrowsKeyPath

The key path for the subrows.

```
- (NSString *)subrowsKeyPath
```

Return Value
The key path for the subrows.

Discussion
The default value is `@"subrows"`.

The key path is used to get the nested rows in the “rows” binding. The corresponding property should be an ordered to-many relationship containing additional bound row objects.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- setSubrowsKeyPath:` (page 25)

Declared In
`NSRuleEditor.h`

viewDidMoveToWindow

Overrides the `NSView` implementation.

```
- (void)viewDidMoveToWindow
```

Special Considerations

If you override this method in a subclass, you must invoke super’s implementation.

Delegate Methods

ruleEditor:child:forCriterion:withRowType:

Returns the child of a given item at a given index.

```
- (id)ruleEditor:(NSRuleEditor *)editor child:(NSInteger)index
    forCriterion:(id)criterion withRowType:(NSRuleEditorRowType)rowType
```

Parameters

editor

The rule editor that sent the message.

index

The index of the requested child criterion. This value must be in the range from `0` up to (but not including) the number of children, as reported by the delegate in ruleEditor:numberOfChildrenForCriterion:withRowType: (page 28).

criterion

The parent of the requested child, or `nil` if the rule editor is requesting a root criterion.

rowType

The type of the row.

Return Value

An object representing the requested child (or root) criterion. This object is used by the delegate to represent that position in the tree, and is passed as a parameter in subsequent calls to the delegate.

Special Considerations

The delegate must implement this method.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`NSRuleEditor.h`

ruleEditor:displayValueForCriterion:inRow:

Returns the value for a given criterion.

```
- (id)ruleEditor:(NSRuleEditor *)editor displayValueForCriterion:(id)criterion
    inRow:(NSInteger)row
```

Parameters

editor

The rule editor that sent the message.

criterion

The criterion for which the value is required.

row

The row number of *criterion*.

Return Value

The value for *criterion*.

Discussion

The value should be an instance of `NSString`, `NSView`, or `NSMenuItem`. If the value is an `NSView` or `NSMenuItem`, you must ensure it is unique for every invocation of this method; that is, do not return a particular instance of `NSView` or `NSMenuItem` more than once.

Special Considerations

The delegate must implement this method.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`NSRuleEditor.h`

ruleEditor:numberOfChildrenForCriterion:withRowType:

Returns the number of child items of a given criterion or row type.

```
- (NSInteger)ruleEditor:(NSRuleEditor *)editor
    numberOfChildrenForCriterion:(id)criterion
    withRowType:(NSRuleEditorRowType)rowType
```

Parameters

editor

The rule editor that sent the message.

criterion

The criterion for which the number of children is required.

rowType

The type of row of *criterion*.

Return Value

The number of child items of *criterion*. If *criterion* is `nil`, return the number of root criteria for the row type *rowType*.

Special Considerations

The delegate must implement this method.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`NSRuleEditor.h`

ruleEditor:predicatePartsForCriterion:withDisplayValue:inRow:

Returns a dictionary representing the parts of the predicate determined by the given criterion and value.

```
- (NSDictionary *)ruleEditor:(NSRuleEditor *)editor
    predicatePartsForCriterion:(id)criterion withDisplayValue:(id)value
    inRow:(NSInteger)row
```

Parameters

editor

The rule editor that sent the message.

criterion

The criterion for which the predicate parts are required.

value

row

The row number of *criterion*.

Return Value

A dictionary representing the parts of the predicate determined by the given criterion and value. The keys of the dictionary should be the string constants specified in "Predicate Part Keys" (page 31) with corresponding appropriate values.

Discussion

Implementation of this method is optional.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSRuleEditor.h`

ruleEditorRowsDidChange:

Notifies the receiver that a rule editor's rows changed.

```
- (void)ruleEditorRowsDidChange:(NSNotification *)notification
```

Parameters

notification
A `NSRuleEditorRowsDidChangeNotification` notification.

Discussion
If this method is implemented, `NSRuleEditor` automatically registers its delegate to receive `NSRuleEditorRowsDidChangeNotification` notifications to this method.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSRuleEditor.h`

Constants

NSRuleEditorNestingMode

Specifies a type for nesting modes.

```
typedef NSUInteger NSRuleEditorNestingMode;
```

Discussion
See "Nesting Modes" (page 29) for possible values.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSRuleEditor.h`

Nesting Modes

These constants specify the nesting mode for the rule editor.

```
enum {
    NSRuleEditorNestingModeSingle,
    NSRuleEditorNestingModeList,
    NSRuleEditorNestingModeCompound,
    NSRuleEditorNestingModeSimple
};
```

Constants

`NSRuleEditorNestingModeSingle`

Only a single row is allowed.

Plus/minus buttons are not shown.

Available in Mac OS X v10.5 and later.

Declared in `NSRuleEditor.h`.

`NSRuleEditorNestingModeList`

Allows a single list, with no nesting and no compound rows.

Available in Mac OS X v10.5 and later.

Declared in `NSRuleEditor.h`.

`NSRuleEditorNestingModeCompound`

Unlimited nesting and compound rows.

This is the default.

Available in Mac OS X v10.5 and later.

Declared in `NSRuleEditor.h`.

`NSRuleEditorNestingModeSimple`

One compound row at the top with subrows beneath it, and no further nesting allowed.

Available in Mac OS X v10.5 and later.

Declared in `NSRuleEditor.h`.

Declared In

`NSRuleEditor.h`

NSRuleEditorRowType

Specifies a type for row types.

```
typedef NSUInteger NSRuleEditorRowType;
```

Discussion

See "Row Types" (page 30) for possible values.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`NSRuleEditor.h`

Row Types

Specify the type of a rule editor row.

```
enum {
    NSRuleEditorRowTypeSimple,
    NSRuleEditorRowTypeCompound
};
```

Constants

`NSRuleEditorRowTypeSimple`

Specifies a simple row.

Available in Mac OS X v10.5 and later.

Declared in `NSRuleEditor.h`.

`NSRuleEditorRowTypeCompound`

Specifies a compound row.

Available in Mac OS X v10.5 and later.

Declared in `NSRuleEditor.h`.

Declared In

`NSRuleEditor.h`

Predicate Part Keys

These strings are used as keys to the dictionary returned from the optional delegate method ruleEditor:predicatePartsForCriterion:withDisplayValue:inRow: (page 28). To construct a valid predicate, the union of the dictionaries for each item in the row must contain the required parts.

```
APPKIT_EXTERN NSString * const NSRuleEditorPredicateLeftExpression;
APPKIT_EXTERN NSString * const NSRuleEditorPredicateRightExpression;
APPKIT_EXTERN NSString * const NSRuleEditorPredicateComparisonModifier;
APPKIT_EXTERN NSString * const NSRuleEditorPredicateOptions;
APPKIT_EXTERN NSString * const NSRuleEditorPredicateOperatorType;
APPKIT_EXTERN NSString * const NSRuleEditorPredicateCustomSelector;
APPKIT_EXTERN NSString * const NSRuleEditorPredicateCompoundType;
```

Constants

`NSRuleEditorPredicateLeftExpression`

The corresponding value is an `NSExpression` object representing the left expression in the predicate.

This value is required for a non-`nil` comparison predicate.

Available in Mac OS X v10.5 and later.

Declared in `NSRuleEditor.h`.

`NSRuleEditorPredicateRightExpression`

The corresponding value is an `NSExpression` object representing the right expression in the predicate.

This value is required for a non-`nil` comparison predicate.

Available in Mac OS X v10.5 and later.

Declared in `NSRuleEditor.h`.

`NSRuleEditorPredicateComparisonModifier`

The corresponding value is an `NSNumber` object representing a `NSComparisonPredicateModifier` constant the of the predicate.

This value is optional—if not specified, `NSDirectPredicateModifier` is assumed.

Available in Mac OS X v10.5 and later.

Declared in `NSRuleEditor.h`.

`NSRuleEditorPredicateOptions`
: The corresponding value is an `NSNumber` object representing a `NSComparisonPredicateOptions` bitfield.

  If no value is specified, `0` (no options) is assumed.

  Available in Mac OS X v10.5 and later.

  Declared in `NSRuleEditor.h`.

`NSRuleEditorPredicateOperatorType`
: The corresponding value is an `NSNumber` object representing a `NSPredicateOperatorType` constant.

  This value is required for a non-`nil` comparison predicate.

  Available in Mac OS X v10.5 and later.

  Declared in `NSRuleEditor.h`.

`NSRuleEditorPredicateCustomSelector`
: The corresponding value is an `NSString` object representing a custom selector.

  If specified, this overrides the operator type, options, and comparison modifier.

  Available in Mac OS X v10.5 and later.

  Declared in `NSRuleEditor.h`.

`NSRuleEditorPredicateCompoundType`
: The corresponding value is an `NSNumber` object representing a `Compound Predicate Types` constant.

  If specified, the other keys are ignored and the predicate for the row will be an `NSCompoundPredicate` predicate whose subpredicates are the predicates of the subrows of the given row.

  Available in Mac OS X v10.5 and later.

  Declared in `NSRuleEditor.h`.

Declared In
`NSRuleEditor.h`

Notifications

NSRuleEditorRowsDidChangeNotification

This notification is posted to the default notification center whenever the view's rows change.

The object is the rule editor; there is no `userInfo` object.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSRuleEditor.h`

Document Revision History

This table describes the changes to *NSRuleEditor Class Reference*.

Date	Notes
2008-02-08	Removed a broken link.
2007-10-31	Corrected grammatical errors.
2007-02-21	First version of this document.

Document Revision History

Index

S

V