Overview

The Combo Box Control combines a text box with an attached list box from which users may choose items. Combo boxes are helpful when the number of acceptable values is both large and not user-defined.

Ancestors:

sashelp.fsp.Object.class
sashelp.fsp.Widget.class
sashelp.classes.Afcontrol.class

Class:

sashelp.classes.Combobox_c.class

Using the Combo Box Control

Key properties to set

allowNewItems attribute Specify whether or not users can add to the list of items that appear in the combo box at run-time.
You can also query or set this attribute using SCL. For example, the following line of code enables users to add to the combo box list at run-time:

```javascript
combobox1.allowNewItems='Yes';
```

**selectedItem attribute** (default attribute)

Specify which of the list items you want selected by default.

You can also query this attribute to find out what the user has selected. For example, the following line of SCL code queries the item that is currently selected:

```javascript
choice = combobox1.selectedItem;
```

**Note:** The following line of code is also valid since the `selectedItem` attribute is the default attribute:

```javascript
choice = combobox1;
```

---

### Setting combo box items

You can set the items that appear in a combo box in these ways:

- You can set the items attribute in the Properties window. When you click the value cell for the items attribute, the Items window appears where you can enter each of the items you want.
- You can populate the combo box with items from a list model by setting the value of the model attribute in the Properties window to the name of the list model.
- You can also populate the combo box with items from a list model by dragging the model from the Components window onto the combo box in the frame.
- Finally, you can specify the values of the items in the combo box at run time.

---

### Build-time attributes

The following attribute is a build-time attribute, meaning it can only be set at build-time if you choose to use it:

- `comboboxStyle`

---

### Using SCL to specify combo box items

You can specify combo box items using an SCL program. For example, the following code creates a list and assigns the list to a combo box:

```javascript
petList = makelist(); rc = insertc(petList, 'Dog', -1); rc = insertc(petList, 'Cat', -1); rc = insertc(petList, 'Bird', -1); combobox1.items = petList;
```
You could just as easily fill a combo box with a list of items that already exists in a model that you have instantiated. The following code assigns color list model items to a combo box:

```javascript
combobox1.items = colorlist1.items;
```

## Attributes

Public attributes specified for the Combo Box Control class are described here. See sashelp.classes.Afcontrol.class for inherited attributes.

## Dictionary

### allowNewItems

Returns or sets whether new items entered in the input area at run-time are added to the dropdown

- **Type:** Character
- **Valid Values:** Yes, No

### backgroundColor

Returns or sets the background color of a control

- **Type:** Character
- **Valid Values:** (All SAS-supported color values)
- **Editor:** sashelp.classes.colorEditor.frame

### comboBoxStyle

Returns or sets the style of the combo box

- **Type:** Character
- **Valid Values:** Dropdown List, Dropdown Combo
**conformSize**

Returns or sets whether to size the height of the control based on the font used for the text

*Type:* Character  
*Valid Values:* Yes, No

**cursorPosition**

Returns or sets the position of the cursor in the text area of the control

*Type:* Numeric  
*Valid Values:*

**defaultAttribute**

Returns or sets the attribute name used for the object’s default attribute

*Type:* Character  
*Valid Values:* \sashelp.classes.defaultattributevalues.scl

**description**

Combo Box Control

*Type:* Character  
*Valid Values:*
**dragInfo**

Returns or sets the information that is transferred from the control when a drag operation occurs

Type: List
Valid Values:
Editor: SASHELP.FSP.DRAGDROPINFOEDITOR.FRAME

**dropInfo**

Returns or sets the information that defines the data representations that can be dropped on the control

Type: List
Valid Values:
Editor: SASHELP.FSP.DRAGDROPINFOEDITOR.FRAME

**font**

Returns or sets the font applied to the text

Type: List
Valid Values:
Editor: sashelp.classes.fonteditor3.scl

**format**

Returns or sets the SAS format to be applied to the text

Type: Character
Valid Values:
Editor: sashelp.classes.formatlnformatEditor.scl
informat

Returns or sets the SAS informat to be applied to the text

Type: Character

Valid Values:

Editor: sashelp.classes.formatInformatEditor.scl

items

Returns or sets the items displayed in the dropdown

Type: List

Valid Values:

Editor: sashelp.classes.stringListEditor.frame

keyFeedback

Returns or sets whether the key event is sent with every keystroke

Type: Character

Valid Values: Yes, No

markedText

Returns the marked text of the input area

Type: Character

Valid Values:
**selectedIndex**

Returns or sets the selected item position

Type: Numeric
Valid Values:

**selectedItem**

Returns or sets the text displayed in the input area

Type: Character
Valid Values: \sashelp.classes.itemsValues.sd

**textColor**

Returns or sets the foreground color of the text

Type: Character
Valid Values: (All SAS-supported color values)
Editor: sashelp.classes.colorEditor.frame

**textCompletion**

Returns or sets whether to complete the text during entry

Type: Character
Valid Values: Yes, No
textOrientation

Returns or sets how the text in the input area is written

Type: Character
Valid Values: Yes, No

Methods

Methods specified for the Combo Box Control class are described here. See sashelp.classes.Afcontrol.class for inherited methods.

__clearMark

Clears the marked text

Syntax

ObjectName_clearMark();

__find

Finds a string in the list of items

Syntax

ObjectName_find( startRow, text );
ObjectName_find( startRow, text, caseSensitive );
<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>startRow</td>
<td>Numeric</td>
<td>Update</td>
<td>Specifies the starting row of the search and returns the row index if found</td>
</tr>
<tr>
<td>text</td>
<td>Character</td>
<td>Input</td>
<td>Specifies the text string to search for</td>
</tr>
<tr>
<td>caseSensitive</td>
<td>Character</td>
<td>Input</td>
<td>Specifies whether to perform a case sensitive search</td>
</tr>
</tbody>
</table>

**_getMark**

Returns the position of the marked text

**Syntax**

```plaintext
objectName_getMark( startPosition, endPosition );
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>startPosition</td>
<td>Numeric</td>
<td>Output</td>
<td>Specifies the starting position of the mark</td>
</tr>
<tr>
<td>endPosition</td>
<td>Numeric</td>
<td>Output</td>
<td>Specifies the ending position of the mark</td>
</tr>
</tbody>
</table>

**_onClick**

Invoked when the click event occurs

**Syntax**

```plaintext
objectName_onClick();
```
_onConformSize

Invoked when the conformSize event occurs

Syntax

objectName_onConformSize();

_onDoubleClick

Invoked when the doubleClick event occurs

Syntax

objectName_onDoubleClick();

_onKey

Invoked when the key event occurs

Syntax

objectName_onKey();

_onPopup

Invoked when the popup event occurs
Syntax

objectName_onPopup​();

__onReturnKey

Invoked when the returnKey event occurs

Syntax

objectName_onReturnKey​();

__onSelect

Invoked when the select event occurs

Syntax

objectName_onSelect​(selectedIndex​);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>selectedIndex</td>
<td>Numeric</td>
<td>Input</td>
<td>contains the selected item position</td>
</tr>
</tbody>
</table>

__onValueChanged

Invoked when the valueChanged event occurs
**_setMark_**

Sets the position of the marked text

Syntax

\[ \text{objectName\_onValueChanged();} \]

---

**Syntax**

\[ \text{return }= \text{objectName\_setMark( startPosition, endPosition );} \]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>return</td>
<td>Numeric</td>
<td>Return</td>
<td>Specifies the starting position of the mark</td>
</tr>
<tr>
<td>startPosition</td>
<td>Numeric</td>
<td>Input</td>
<td>Specifies the ending position of the mark</td>
</tr>
<tr>
<td>endPosition</td>
<td>Numeric</td>
<td>Input</td>
<td>specified the index of the item selected from the menu</td>
</tr>
</tbody>
</table>

**_showPopup_**

Displays a pop-up menu containing items in a list

Syntax

\[ \text{objectName\_showPopup( popupItemList, selectedIndex );} \]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>popupItemList</td>
<td>List</td>
<td>Input</td>
<td>specifies the items to display in the popup</td>
</tr>
<tr>
<td>selectedIndex</td>
<td>Numeric</td>
<td>Update</td>
<td>specifies the index of the item selected from the menu</td>
</tr>
</tbody>
</table>
Events specified for the Combo Box Control class are described here. See sashelp.classes.Afcontrol.class for inherited events.

allowNewItems changed
   Occurs when the allowNewItems attribute is changed
dclick
   Occurs on a click
comboBoxStyle changed
   Occurs when the comboBoxStyle attribute is changed
conformSize
   Occurs when the control is resized due to conformSize
conformSize changed
   Occurs when the conformSize attribute is changed
cursorPosition changed
   Occurs when the cursorPosition attribute is changed
doubleClick
   Occurs on a double-click
font changed
   Occurs when the font attribute is changed
format changed
   Occurs when the format attribute is changed
informat changed
   Occurs when the informat attribute is changed
items changed
   Occurs when the items attribute is changed
key
   Occurs on a keystroke
keyFeedback changed
   Occurs when the keyFeedback attribute is changed
markedText changed
   Occurs when the markedText attribute is changed
popup
   Occurs on a popup request
returnKey
   Occurs on a carriage Return or Enter
select
   Occurs when an item is selected from the dropdown
selectedIndex changed
   Occurs when the selectedIndex attribute is changed
selectedItem changed
   Occurs when the selectedItem attribute is changed
textColor changed
   Occurs when the textColor attribute is changed
textCompletion changed
   Occurs when the textCompletion attribute is changed
textOrientation changed
   Occurs when the textOrientation attribute is changed
valueChanged
   Occurs when the value of the combo box is changed through a key event, cut and paste, or delete

Event Handlers

Event handlers specified for the Combo Box Control class are described here. See sashelp.classes.Afcontrol.class for inherited event handlers.

<table>
<thead>
<tr>
<th>Event</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>.click</td>
<td>_onClick</td>
<td>Executes when the click event occurs</td>
</tr>
<tr>
<td>.conformSize</td>
<td>_onConformSize</td>
<td>Executes when the conformSize event occurs</td>
</tr>
<tr>
<td>.doubleClick</td>
<td>_onDoubleClick</td>
<td>Executes when the doubleClick event occurs</td>
</tr>
<tr>
<td>.key</td>
<td>_onKey</td>
<td>Executes when the key event occurs</td>
</tr>
<tr>
<td>.popup</td>
<td>_onPopup</td>
<td>Executes when the popup event occurs</td>
</tr>
<tr>
<td>.returnKey</td>
<td>_onReturnKey</td>
<td>Executes when the returnKey event occurs</td>
</tr>
<tr>
<td>.select</td>
<td>_onSelect</td>
<td>Executes when the select event occurs</td>
</tr>
<tr>
<td>.valueChanged</td>
<td>_onValueChanged</td>
<td>Executes when the valueChanged event occurs</td>
</tr>
</tbody>
</table>

Interfaces

Interfaces specified for the Combo Box Control class are described here. See sashelp.classes.Afcontrol.class for inherited interfaces.
Required Interfaces
sashelp.classes.staticstringlist.interface