Overview

Use the Extended Text Entry class to create data entry objects for use exclusively in graphical environments that accept only character data as input. In addition, use the Extended Text Entry class to enable
- the use of fonts provided by the host environment
- font and region resizing, marking of text, scrolling of text, or paragraph-style word wrap
- applications to display text using customized colors as well as CPARMS colors.

Extended text entry objects have the following characteristics that distinguish them from other data entry objects:

- for use in graphical environments only
  extended text entry objects do not have alternate definitions for non-graphical environments.

- can accept multiple lines of character-only data
  Each input and display line has a maximum length of 200 characters.

- support the use of host fonts
  Extended text entry objects can use various fonts provided by the host environment.

Note: The value of an extended text entry is numeric. △

The text entered into an extended text entry object is handled through an SCL list. Each line is stored as an item in the list. In the SCL program, you access all of the lines of text for the extended text entry object by using _getValue, or specific lines of text by using _getText or _getLine.

Parent:
  sashelp.fsp.widget.class

Class:
  sashelp.fsp.GLabel.class
Methods specific to the Extended Text Entry class are described here. Inherited methods are described in the Object class and the Widget class.

Dictionary

_clear

Removes all text and sets all lines to blank

Syntax
CALL NOTIFY (ext_text_name, '_clear');

Details
The _clear method clears all lines and scrolls the object to the top.

_clearMarks

Clears marks set with the _setSelect method or dragged by the user

Syntax
CALL NOTIFY (ext_text_name, '_clearMarks');

_cursor

Positions the cursor within the object

Syntax
CALL NOTIFY (ext_text_name, '_cursor<< line-num >>offset>>);
Argument | Type | Description
---|---|---
line-num | N | specifies the line number of the new position
offset | N | specifies the character offset into current line

**Details**
The location of the cursor is the insertion point for all paste operations and for keyboard input. The default is to position the cursor in the (1,1), or upper left, location in the field.

---

**_feedback**

Handles each character as an event

---

**Syntax**

```sh
_feedback:METHOD event$20 line offset$8; endmethod;
```

Argument | Type | Description
---|---|---
event | C | specifies the event code that modified the line
line | N | specifies the line number of the affected line
offset | N | specifies the character offset of the affected character

**Details**
The _feedback method is run for every character that is input when the Keystroke Feedback mode is chosen from the attribute window, or when the _setMode method is called with the parameter ‘ALWAYS’. See _setMode in this class.

The _feedback method is run for every carriage return regardless of mode settings.

Do not call this method from the SCL code using _callNotify or _callSend. Define a method block and use it to override the _feedback method. The _feedback method will be called automatically. The following events are then sent to this method:

**Table 66.1**

<table>
<thead>
<tr>
<th>Possible Events</th>
<th>Value is...</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRINTABLE</td>
<td>a printable character was typed in</td>
</tr>
<tr>
<td>RETURN</td>
<td>a Carriage Return or New Line was entered</td>
</tr>
<tr>
<td>DELWORD</td>
<td>a word was deleted</td>
</tr>
<tr>
<td>DELPREVWORD</td>
<td>the previous word was deleted</td>
</tr>
</tbody>
</table>
Possible Events | Value is...
---|---
DELCHAR | a character was deleted
DELPREVCHAR | the previous character was deleted
DELTOEOL | all characters to the end of the line were deleted
DELTOBOL | all characters to the beginning of the line were deleted
DELLINE | the line was deleted
ENTER | massive changes were made to the object in one operation

**Example**

This example uses the _feedback method to test input against a list of color names. If a match is found, the entered value is replaced with the found value. The portion of text that is typed in is marked so that the next keystroke replaces that text. For example, if you type ‘R’, it displays ‘RED.’ If you then type an ‘O’ following the ‘R’, it displays ‘ROSE’.

```plaintext
length v $ 200;

fb: method event $ 20 line offset 8;
call send(_self_, "_getText", v);
len = length(v);
if (len = 0) then return;
if (event ^= "PRINTABLE") then return;
/* get the list of valid values */
l = getiteml(_SELF_, "LIST");
rc = searchc(l, v, 1, 1, 'y', 'y');
if (rc > 0) then do;
v =getitemc(l, rc);
call send(_SELF_, "_setText", v);
call send(_SELF_, "_setSelect", 1,
        len+1, 1, 200);
end;
endmethod;

/* _init method sets up the list that */
/* is used to hold all possible */
/* valid selections */

init: method;
call super(_SELF_, "_init");
l = makelist();
rc = insertc(l, "BLACK", -1);
rc = insertc(l, "WHITE", -1);
rc = insertc(l, "RED", -1);
rc = insertc(l, "GREEN", -1);
rc = insertc(l, "BLUE", -1);
rc = insertc(l, "PURPLE", -1);
rc = insertc(l, "VIOLET", -1);
rc = insertc(l, "ORANGE", -1);
rc = insertc(l, "YELLOW", -1);
rc = insertc(l, "PINK", -1);
rc = insertc(l, "CYAN", -1);
```
rc = insertc(l, "MAGENTA", -1);
rc = insertc(l, "BROWN", -1);
rc = insertc(l, "GOLD", -1);
rc = insertc(l, "LIME", -1);
rc = insertc(l, "GRAY", -1);
rc = insertc(l, "LILAC", -1);
rc = insertc(l, "MAROON", -1);
rc = insertc(l, "ROSE", -1);
rc = setniteml(_SELF_, l, "LIST");

/* force _feedback method to run all the time */
call send(_SELF_, "_setMode", "ALWAYS");
endmethod;

/* Don’t forget to cleanup during the _term method */
term: method;
l = getniteml(_SELF_, "LIST");
rc = dellist(l);
call super(_SELF_, "_term");
endmethod;

_getAutoflow

Determines if automatic text wrapping is on or off

Syntax

CALL NOTIFY (ext_text_name, ‘_getAutoflow’, value);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>C</td>
<td>returns the current status of autoflow:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>’Y’      autoflow is on</td>
</tr>
<tr>
<td></td>
<td></td>
<td>’N’      autoflow is off</td>
</tr>
</tbody>
</table>

_getCaps

Determines if automatic capitalization is on or off
Syntax
CALL NOTIFY (ext_text_name, '_getCaps', value);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>C</td>
<td>returns the status of capitalization:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'Y' capitalization is on</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'N' capitalization is off</td>
</tr>
</tbody>
</table>

_getCurcol

Returns the horizontal position of the cursor within the extended text entry object

Syntax
CALL NOTIFY (ext_text_name, '_getCurcol', offset);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>offset</td>
<td>N</td>
<td>returns the number of columns (characters) from the left side of the object</td>
</tr>
</tbody>
</table>

_getCurrow

Returns the line number of the cursor within the extended text entry object

Syntax
CALL NOTIFY (ext_text_name, '_getCurrow', line);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>line</td>
<td>N</td>
<td>returns the line number of the cursor position</td>
</tr>
</tbody>
</table>

_getHscroll

Returns the horizontal scroll unit and optionally the number of units to scroll
**Syntax**

CALL NOTIFY (ext_text_name, '_getHscroll', unit, <num-units>);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unit</td>
<td>C</td>
<td>returns the scrolling unit:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'PAGE' number of visible columns</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'HALF' half the number of visible columns.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'MAX' to the number of the maximum columns minus the visible columns (See _getMaxcol in this class.)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'COLUMN' amount specified by num-units.</td>
</tr>
<tr>
<td>num-units</td>
<td>N</td>
<td>number of units to scroll</td>
</tr>
</tbody>
</table>

**See Also**

_hscroll

---

**_getLine**

Returns the text of the specified line

**Syntax**

CALL NOTIFY (ext_text_name, '_getLine', text-string, line-num);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>text-string</td>
<td>C</td>
<td>returns the value of the line of the object</td>
</tr>
<tr>
<td>line-num</td>
<td>N</td>
<td>specifies the line number to return</td>
</tr>
</tbody>
</table>

---

**_getMaxcol**

Returns the maximum number of displayable columns allowed in the extended text field

**Syntax**

CALL NOTIFY (ext_text_name, '_getMaxcol', cols);
**_getMaxrow_**

Returns the maximum number of displayable rows

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rows</td>
<td>N</td>
<td>returns the maximum number of displayable rows</td>
</tr>
</tbody>
</table>

**Details**

The `_getMaxcol` method returns the value from the **Line Length** field that you can set in the Extended Text Entry Attributes window. The maximum number of displayable columns is not necessarily the same as the number of columns visible at one time in the extended text entry. Instead, it is the maximum number of columns that the extended text entry will hold. Scrolling the extended text entry may be necessary to view all displayable columns.

**See Also**

`_getViscol`

---

**_getMaxrow_**

Returns the maximum number of displayable rows

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rows</td>
<td>N</td>
<td>returns the maximum number of displayable rows</td>
</tr>
</tbody>
</table>

**Details**

The maximum number of displayable rows is not necessarily the same as the number of rows visible at one time in the extended text entry. Instead, it is the maximum number of lines the extended text entry will hold. Scrolling the extended text entry may be necessary to view all displayable rows. If unlimited rows are specified in the attribute window, `_getMaxrow` returns -1.

**See Also**

`_getVisrow`.

---

**_getMode_**

Returns the current setting of the Keystroke Feedback mode

**Syntax**

CALL NOTIFY (ext_text_name, '_getMode', text-string);
Extended Text Entry Class

### _getText

**Argument Type Description**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>text-string</td>
<td>C</td>
<td>'ALWAYS' _feedback method runs for every modification of the object</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'NONE' _feedback method never runs</td>
</tr>
</tbody>
</table>

#### Syntax

CALL NOTIFY (ext_text_name, '_getText', text-string);

### _getSelect

**Returns the position of the currently marked area**

#### Syntax

CALL NOTIFY (ext_text_name, '_getSelect', start-line, start-offset, end-line, end-offset);

**Argument Type Description**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>start-line</td>
<td>N</td>
<td>returns the beginning line of the current marked area</td>
</tr>
<tr>
<td>start-offset</td>
<td>N</td>
<td>returns the offset of the first character in the marked area</td>
</tr>
<tr>
<td>end-line</td>
<td>N</td>
<td>returns the last line of the current marked area</td>
</tr>
<tr>
<td>end-offset</td>
<td>N</td>
<td>returns the offset of the last character in the marked area</td>
</tr>
</tbody>
</table>

**Details**

Marked areas in the extended text entry are only line marks; that is, they span the entire line.

### _getText

**Returns the text of the first line of the extended text entry field**

**Inherited** from Widget

#### Syntax

CALL NOTIFY (ext_text_name, '_getText', text-string);
**Argument Type Description**

| text-string  | C                           | returns the text of the first line |

**Details**
The `_getText` method is a shorthand method for the `_getLine` method with a parameter of 1.

### _getToprow

**Returns the line number of the first visible row**

---

**Syntax**

CALL NOTIFY (ext_text_name, `_getToprow`, line);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>line</td>
<td>N</td>
<td>returns the line number of the first row of the currently displayed rows</td>
</tr>
</tbody>
</table>

### _getValue

**Returns every line of text in the extended text entry field**

**Inherited** from Widget

---

**Syntax**

CALL NOTIFY (ext_text_name, `_getValue`, list);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>list</td>
<td>N</td>
<td>returns the identifier of an SCL list that contains every line of text</td>
</tr>
</tbody>
</table>

**Details**
The `_getValue` method is equivalent to calling the `_getLine` method for each line.
_getViscol

Returns the number of columns that are visible at one time

Syntax
CALL NOTIFY (ext_text_name, '_getViscol', columns);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>columns</td>
<td>N</td>
<td>returns the number of columns that can be displayed at one time. This number is equivalent to the character width of the region.</td>
</tr>
</tbody>
</table>

_getVisrow

Returns the number rows that are visible at one time using the current font height

Syntax
CALL NOTIFY (ext_text_name, '_getVisrow', rows);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rows</td>
<td>N</td>
<td>returns the number of rows that can be displayed at one time. This number is equivalent to the character height of the region.</td>
</tr>
</tbody>
</table>

See also
_vscroll.

_getVscroll

Returns the vertical scrolling unit and optionally the number of units to scroll at once

Syntax
CALL NOTIFY (ext_text_name, '_getVscroll', unit, <num-units>);
### Argument Type Description

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unit</td>
<td>C</td>
<td>returns the scrolling unit:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'PAGE' number of visible columns</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'HALF' half the number of visible columns (default)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'MAX' to the number of the maximum columns minus the visible columns</td>
</tr>
<tr>
<td></td>
<td></td>
<td>See _getMaxcol in this class.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'ROW' amount specified by num-units</td>
</tr>
<tr>
<td>num-units</td>
<td>N</td>
<td>number of units to scroll</td>
</tr>
</tbody>
</table>

### _hscroll

Scrolls the extended text entry horizontally by a specified number of units

#### Syntax

CALL NOTIFY (ext_text_name, '_hscroll'<, unit><, num-units>);

### Details

If you call the _hscroll method from the INIT label, you must precede it with the _refresh method to update the display of the extended text entry object.

### See Also

_getHscroll and _setHscroll.

### _paste

Paste the contents of current paste buffer into the object at the current cursor location.
Syntax

CALL NOTIFY (ext_text_name, '_paste', action);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>C</td>
<td>defines how to paste:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'INSERT' inserts the paste buffer into the object</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'REPLACE' overlays the current contents of the object with the paste buffer (default)</td>
</tr>
</tbody>
</table>

_pasteList

Pastes the contents of a list into the object at the current cursor location

Syntax

CALL NOTIFY (ext_text_name, '_pasteList', action, list);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>C</td>
<td>specifies how to paste:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'INSERT' inserts the contents of the list into the object</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'REPLACE' overlays the current contents of the object with the contents of the list</td>
</tr>
<tr>
<td>list</td>
<td>N</td>
<td>specifies the identifier of an SCL list that contains the lines to paste</td>
</tr>
</tbody>
</table>

_pasteText

Pastes the contents of the string into the object at the current cursor location

Syntax

CALL NOTIFY (ext_text_name, '_pasteText', action, string);
### Argument Type Description

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>C</td>
<td>defines how to paste:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'INSERT' inserts the text into the object</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'REPLACE' overlays the current contents of the object with the text</td>
</tr>
<tr>
<td>string</td>
<td>C</td>
<td>string to be pasted</td>
</tr>
</tbody>
</table>

---

### `_protect`

Sets the level of protection

#### Syntax

```c
CALL NOTIFY (ext_text_name, '_protect' , mode);
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mode</td>
<td>C</td>
<td>specifies the protection mode:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'SCROLLABLE' the extended text entry captures events, but it does not change the value of the extended text entry in any way.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>missing the extended text entry has the same protection as all other widgets: namely, it cannot be tabbed, and all events occurring in the extended text entry’s region are passed on to the parent region</td>
</tr>
</tbody>
</table>

#### Details

Use SCROLLABLE mode if the displayable text is larger than the region in which it is to display. Users can tab to the extended text entry and use the arrow keys or scroll bars to scroll the text in the viewport, but they cannot change the contents.

---

### `_setAutoflow`

Turns automatic text wrapping on or off

#### Syntax

```c
CALL NOTIFY (ext_text_name, '_setAutoflow', value);
```
**Argument Type Description**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>C</td>
<td>specifies the autoflow:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'Y' turns on autoflow</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'N' turns off autoflow</td>
</tr>
</tbody>
</table>

**Details**

If you type more characters than fit on one line without pressing RETURN, the line automatically wraps to the next line. If autoflow is off, the wrap automatically occurs at the end of the line, and the next character is placed at the beginning of the next line, regardless of word spacing. If autoflow is on, it attempts to move the last word typed to the next line by moving all characters after the last space on the line to the next line. If there is no space on the line, autoflow has no effect.

---

**setCaps**

Sets automatic capitalization on or off

---

**Syntax**

CALL NOTIFY (ext_text_name, '_setCaps', value);

---

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>C</td>
<td>specifies the capitalization</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'Y' turns capitalization on</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'N' turns capitalization off</td>
</tr>
</tbody>
</table>

---

**setColor**

Assigns the color of the text

---

**Syntax**

CALL NOTIFY (ext_text_name, '_setColor', color-name);
_setFont

Assigns a font for the text

**Syntax**

CALL NOTIFY (ext_text_name, '_setFont', font-list-id);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>color-name</td>
<td>C</td>
<td>specifies the color of the text using the specified valid RGB or SAS color name</td>
</tr>
<tr>
<td>font-list-id</td>
<td>N</td>
<td>specifies the identifier of an SCL list that contains the font information</td>
</tr>
</tbody>
</table>

**Details**

The value passed to the _setFont method can come from the value of a font selector widget, or from a _getFont method in a widget which uses a font list (such as the extended text entry itself). The font list must be in the format returned by the FONTSEL() function in SCL. Pass in an empty list, or 0, to use the default SAS font.

**Example**

The CALL NOTIFY statement in the following example sets the font for obj1 to the values defined in the list font1:

```sas
init:
  font1=makelist();
  setnitemn(font1,2,'style');
  setnitemn(font1,11,'weight'); he
  call notify('obj1','_set_font_',font1);
return;
```

__setHscroll

Assigns the unit, and optionally the number of units, to scroll an extended text entry horizontally

**Syntax**

CALL NOTIFY (ext_text_name, '_setHscroll', unit<num-units>);
Extended Text Entry Class

_setLine

**Argument** | **Type** | **Description**
---|---|---
unit | C | specifies the scrolling unit:
'PAGE' | number of visible columns
'HALF' | half the number of visible columns (the default)
'MAX' | to the number of the maximum columns minus the visible columns. See _getMaxcol in this class.
'COLUMN' | amount specified by num-units
num-units | N | specifies the number of units to scroll; the default is 1

**See Also**
_hscroll,

_setJustify

Assigns justification to the text

---

**Syntax**

CALL NOTIFY (ext_text_name, '_setJustify', style);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| style | C | specifies the justification style:
'CENTER'
'LEFT' (default)
'RIGHT'
'NONE'

_setLine

Assigns text to the specified line

---

**Syntax**

CALL NOTIFY (ext_text_name, '_setLine', text-string, line-num);
### _setMaxcol

Specifies the maximum number of columns allowed on one line of the extended text field.

**Syntax**

CALL NOTIFY (ext_text_name, '_setMaxcol', cols);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cols</td>
<td>N</td>
<td>specifies the maximum number of columns allowed on a line; cols cannot exceed 200</td>
</tr>
</tbody>
</table>

**Details**

If the maximum length is greater than the display length of the field, you can scroll left and right.

### _setMaxrow

Assigns the maximum number of rows in the extended text entry object.

**Syntax**

CALL NOTIFY (ext_text_name, '_setMaxrow', rows);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rows</td>
<td>N</td>
<td>specifies the maximum number of rows</td>
</tr>
</tbody>
</table>

**Details**

You can set the number of rows to virtually infinite if rows is set to 0 or . (missing).
### _setMode

**Sets the Keystroke Feedback mode**

#### Syntax

CALL NOTIFY (ext_text_name, '_setMode', string);

#### Argument Type Description

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>C</td>
<td>specifies the Keystroke Feedback mode:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'ALWAYS' runs the _feedback method for every modification made to the object</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'NONE' causes _feedback not to run at all</td>
</tr>
</tbody>
</table>

### _setSelect

**Specifies the position of a marked area**

#### Syntax

CALL NOTIFY (ext_text_name, '_setSelect', start-line, start-offset, end-line, end-offset);

#### Argument Type Description

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>start-line</td>
<td>N</td>
<td>specifies the beginning line of the current marked area</td>
</tr>
<tr>
<td>start-offset</td>
<td>N</td>
<td>specifies the offset of the first character in the marked area</td>
</tr>
<tr>
<td>end-line</td>
<td>N</td>
<td>specifies the last line of the current marked area</td>
</tr>
<tr>
<td>end-offset</td>
<td>N</td>
<td>specifies the offset of the last character in the marked area</td>
</tr>
</tbody>
</table>

#### Details

The _setSelect method sets the current marked area as if the user had dragged out a mark. Marked areas in the extended text entry are only line marks; that is, they span the entire line.

### _setText

**Assigns text to the first line of the extended text entry**
**Inherited** from Widget

---

**Syntax**

CALL NOTIFY (ext_text_name, 'setToprow', string);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>C</td>
<td>specifies the value of the first line of the object</td>
</tr>
</tbody>
</table>

---

**_setToprow**

Scrolls the specified row to the top

---

**Syntax**

CALL NOTIFY (ext_text_name, '_setToprow', rownum);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rownum</td>
<td>N</td>
<td>specifies the number of the row to scroll to the top</td>
</tr>
</tbody>
</table>

---

**Details**

If you call the _setToprow method from the INIT label, you must precede it with the _refresh method to update the display of the extended text entry.

---

**_setValue**

Assigns text to each line of the object

**Inherited** from Widget

---

**Syntax**

CALL NOTIFY (ext_text_name, '_setValue', list);
## Extended Text Entry Class

### Argument Type Description

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>list</td>
<td>N</td>
<td>specifies the identifier of an SCL list that contains character items for each line</td>
</tr>
</tbody>
</table>

### Details

The `_setValue` method is equivalent to calling the `_setLine` method for each line. If you specify fewer lines in your list than currently exist with data in the extended text entry, then you will lose the extra lines in the extended text entry.

---

### `_setVscroll`

Assigns the unit, and optionally the number of units, to scroll an extended text entry vertically

#### Syntax

CALL NOTIFY (ext_text_name, '_setVscroll', unit, <num-units>);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unit</td>
<td>C</td>
<td>specifies the scrolling unit:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'PAGE' number of visible rows (the default)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'HALF' half the number of visible rows</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'MAX' to the number of the maximum rows minus the visible rows. See <code>_getMaxrow</code> in this class.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'ROW' amount specified by num-units.</td>
</tr>
<tr>
<td>num-units</td>
<td>N</td>
<td>specifies the number of units to scroll; the default is 1</td>
</tr>
</tbody>
</table>

#### See also

_vscroll.

---

### `_snugFit`

Sets the snug fit option

#### Syntax

CALL NOTIFY (object-name, '_snugFit');
**Details**

The _snugFit method snaps the region to the size of the text within the object.

---

**_vscroll**

**Scrolls the extended text entry vertically by a specified number of units**

**Syntax**

CALL NOTIFY (ext_text_name, '_vscroll', <unit>, <num-units>);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unit</td>
<td>C</td>
<td>specifies the scrolling unit:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'PAGE'   number of visible rows</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'HALF'   half the number of visible rows</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'MAX' to the number of the maximum rows minus the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>visible number of rows. See _getMaxrow in this</td>
</tr>
<tr>
<td></td>
<td></td>
<td>class.</td>
</tr>
<tr>
<td>num-units</td>
<td>N</td>
<td>amount specified by num-units</td>
</tr>
</tbody>
</table>

**Details**

If you call the _vscroll method from the INIT label, you must precede it with the _refresh method to update the display of the extended text entry.

```plaintext
_setText
  | Set the first line to the value of string
_getText
  | Get the value of the first line
_setLine
  | Set the specified line to the value of string
_getLine
  | Get the value of the specified line
_clear
  | Set all lines to blank
```
_setMaxrow
  | Set the maximum number of displayable rows
_getMaxrow
  | Returns the maximum number of displayable rows
setMaxcol
  | Set the maximum number of displayable columns
getMaxcol
  | Returns the maximum number of displayable columns
getCurrow
  | Get the line number of the cursor position
getCurcol
  | Get the offset of the cursor position
setValue
  | Set each line for a list
getValue
  | Return each line in list.
setFont
  | Set the font of the extended text entry object
setColor
  | Set the color of the extended text entry object
feedback
  | Method is run for every character or enter key
setMode
  | Set the event feedback mode
getMode
  Returns the current event feedback mode
getSelect
  | Returns the corners of the marked text
setSelect
  | Sets the defined area as marked
paste
  | Pastes the current paste buffer into the object
pasteText
  | Pastes the contents of a string into the object
pasteList
  | Pastes the contents of the list into the object
_cursor
  | Sets the current cursor location
_clearMarks
  | Erases any existing marks
_setJustify
  | Sets the current justifications style
_getVisrow
  | Returns the number of currently visible rows
_getViscol
  | Returns the number of currently visible columns
_getToprow
  | Returns the line number of the first visible row
_setAutoflow
  | Sets autoflow on or off
_getAutoflow
  | Returns the current value for autoflow
_setCaps
  | Sets caps on or off
_getCaps
  | Returns the current value for caps
_snugFit
  | Sets the snug fit option on or off
_hscroll
  | Scrolls horizontally using current units and amount
_getHscroll
  | Returns the current value for hscroll units and amount
_setHscroll
  | Sets the hscroll units and amount
_vscroll
  | Scrolls vertically using current units and amount
_getVscroll
  | Returns the current value for vscroll units and amount
_setVscroll
  | Sets the vscroll units and amount
_setToprow
  | Moves the specified row to the top of the viewport
_hscroll
 Scrolls the extended text entry horizontally

_setHscroll
 Assigns the unit by which an extended text entry can scroll horizontally

_getHscroll
 Returns the horizontal scroll unit

_vscroll
 | Scrolls the extended text entry vertically

_setVscroll
 | Assigns the unit by which an extended text entry can scroll vertically

_getVscroll
 | Returns the vertical scroll unit

_protect
 | Sets one of two protection modes

See Also

_setVscroll and _getVscroll.