Overview

The Extended Table class enables you to create a row containing objects that are logically grouped together and repeated. You provide statements in the SCL program to process these objects as an extended table of objects rather than as a single row of objects. The number of times these objects are repeated is determined when the program is executed. The term logical row describes the set of objects that is repeated. Extended tables can also be used as selection lists. That is, they can display rows of objects from which users can make one or more selections.

Parent:  
sashelp.fsp.widget.class

Class:  
sashelp.fsp.ExtTbl.class

Using Extended Tables

Flow of Control in Extended Tables

When an SCL program for an extended table runs, this process takes place:

1. The INIT section runs.
2. The GETROW section runs. Each row in the table is displayed.
3. The program waits for user input.
If a user modifies a row in the table, the PUTROW section runs for that row, the MAIN section runs, and the GETROW section runs again to display the table with the modified value. Then the program waits for more user input.

In the process, the GETROW section runs for all visible rows each time the PUTROW section runs, no matter how many rows the PUTROW section processes. In addition, the GETROW section runs when an extended table receives a _refresh method. An extended table receives a _refresh method in these cases:

- the frame is displayed initially
- users scroll (both vertically and horizontally)
- users enter the END command, if one of the rows contains a widget that is in error and the widget is not displayed (the widget is forced into view by setting its row to the top)
- these extended table methods are executed:
  - _erroron or _erroroff_ (if a row or widget name is passed in)
  - _needRefresh
  - _needRowRefresh
  - _selectRow or _unselectRow
  - _setMaxcol
  - _setMaxrow
  - _setMaxsel
  - _setToprow
  - _setViscol
  - _setVisrow
  - _unselectAll

For some of these methods, GETROW runs only if the method affects the display. See the individual methods for more information.

Because you can create a frame with multiple extended tables, the GETROW section does not always execute after the MAIN section runs. If GETROW were to run every time the MAIN section ran, modifying a widget outside of the extended tables would cause all GETROW sections in the table to run, whether they relied on the modified object or not. Potentially, this could slow down an application, especially if the GETROW section performed extensive work.

You can override this behavior using the _needRefresh method defined in the Widget class. The _needRefresh method allows the table to receive a _refresh method. You can also use the Extended Table method, _needRowRefresh.

---

**Automatic Instance Variables**

Two special instance variables, _currow and _curcol, are added to the extended table object and all objects within the extended table. These two instance variables are marked as “automatic” so that when the extended table's _getRowLabel or _putRowLabel method, or an object's _objectLabel or _select method run, these variables are automatically initialized to the correct values.

---

**Methods**

Methods specific to the Extended Table class are described here. Inherited methods are described in the Object class and the Widget class.
**CAUTION:**
Do not use CALL SEND to send methods to widgets in extended tables. As a general rule, you should not use CALL SEND to send methods to widget objects in extended tables because the mapping from object name to object identifier that CALL NOTIFY does depends on the value of the current row, which is stored in the reserved variable _currow.

---

**Dictionary**

___

**_endpage**

Signals the end of a page in a dynamic table and stops the processing of the GETROW section

**Syntax**

CALL NOTIFY (extended-table-name, '_endpage');

**Details**

_endpage indicates that the desired number of rows has been displayed. This method is used in the GETROW section.

**Example**

This example reads text from an external file into a table named TABLE. If the read fails, it signifies the end of the table. If the first character of the text is the hexadecimal character ‘C’ (form feed), it signifies a new page. This code is in the GETROW section. Additional code is needed to position the file pointer based on _currow.

    GETROW:
    if (fread(fid)^=0) then call notify
       ('table','_endtable_');
    else
       do;
          fget(fid,c,1);
          if (c='0C') then
             call notify('table',
                          '_endpage_');
       end;
    return;

---

**_endtable**

Signals the end of a dynamic table and stops the processing of the GETROW section
Syntax
CALL NOTIFY (extended-table-name, '_endtable');

Details
_endtable indicates that the maximum number of rows has been displayed. This method is used in the GETROW section.

Example
This example fetches observations from a data set and signals the end of the table when an error is encountered or the end of the data set is reached. Note that the _endtable method is not called for warnings such as a record locked by another process.

GETROW:
rc = fetchobs( dsid, _currow_);
if ( rc > 0 or rc = -1) then do;
call notify( 'table', '_endtable' );
if ( rc ^= -1 ) then
  _msg_ = 'Error encountered reading data set';
end;
return;

_ERROROFF

Turns off the ERROR attribute for a widget in a table

Overrides the _erroroff method for the Widget class.

Syntax
CALL NOTIFY (extended-table-name, '_erroroff', row<, widget-name>);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>row</td>
<td>N</td>
<td>the row that contains the widget. If row is 0, all rows up to the current maximum are processed.</td>
</tr>
<tr>
<td>widget-name</td>
<td>C</td>
<td>the name of the widget. If widget-name is blank, all widgets in the row are processed.</td>
</tr>
</tbody>
</table>

Details
In addition to inheriting the functionality of the _erroroff method for the Widget class, _erroroff for extended tables enables you to initialize or change the ERROR attribute
for widgets outside the GETROW and PUTROW sections of your program. If row or
widget-name name is specified, _erroroff causes the GETROW section of your SCL
program to execute when the extended table receives a _refresh.

Examples

See the examples for the _erroron method.

>Erroron

Turns on the ERROR attribute for a widget in a table

Overrides  This method overrides the _erroron method for the Widget class.

Syntax

CALL NOTIFY (extended-table-name, '_erroron', row<, widget-name>);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>row</td>
<td>N</td>
<td>the row that contains the widget. If row is 0, all rows up to the current maximum are processed.</td>
</tr>
<tr>
<td>widget-name</td>
<td>C</td>
<td>the name of the widget. If widget-name is blank, all widgets in the row are processed.</td>
</tr>
</tbody>
</table>

Details

In addition to inheriting the functionality from the _erroron method for the Widget class, _erroron for extended tables enables you to initialize or change the ERROR attribute for widgets outside the GETROW and PUTROW sections of your program. If row or widget-name name is specified, _erroron causes the GETROW section of your SCL program to execute when the extended table receives a _refresh.

In order for the MAIN section to execute after the _erroron method, you must specify the CONTROL ERROR statement at the beginning of the SCL program.

Examples

These examples show different ways to turn on the ERROR attribute:

- turn on the ERROR attribute for OBJ1 on row 5:
  ```
  MAIN:
  call notify('table','_erroron',5,'obj1');
  return;
  ```

- turn on the ERROR attribute for OBJ1 on all rows:
  ```
  MAIN:
  call notify('table','_erroron',0,'obj1');
  return;
  ```

- turn on the ERROR attribute for all widgets on row 12:
MAIN:
    call notify('table','_erroron',12);
return;

 turn on the ERROR attribute for all widgets on all rows:
MAIN:
    call notify('table','_erroron',0);
return;

_getErrors

Returns the identifier for an SCL list containing the rows or objects in error

Syntax

CALL NOTIFY (extended-table-name, '_getErrors', out-error-listid<, row>);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>out-error-listid</td>
<td>N</td>
<td>an identifier for the SCL list containing either the rows or objects in error</td>
</tr>
<tr>
<td>row</td>
<td>N</td>
<td>the row to examine for objects in error</td>
</tr>
</tbody>
</table>

Details

If row is specified, the list contains the names of the objects in error in the specified row; otherwise, the list contains the rows that are in error.

Example

This example displays the names of all objects in error in row 2:

    call notify('table','_getErrors',
                errlist,2);
    call putlist(errlist,'Names of objects',1);

This example displays the list of rows in error in TABLE:

    call notify('table','_getErrors',
                errlist);
    call putlist(errlist,'List of rows in error',1);

_getHscroll

Returns the horizontal scrolling unit
## Syntax

CALL NOTIFY (extended-table-name, '_getHscroll', out-unit<, out-num-units>);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>out-unit</td>
<td>C</td>
<td>the scrolling unit;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'PAGE'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'HALF'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'COLUMN'</td>
</tr>
</tbody>
</table>
| out-num-units  | N    | the number of columns if the scrolling unit is 'COLUMN'

## Example

_getHscroll returns the horizontal scrolling unit and assigns it to the variable UNIT:

```plaintext
call notify('table','_get_hscroll_','unit');
```

## _getLastSel

Returns the row number of the most recently selected or deselected row

## Syntax

CALL NOTIFY (extended-table-name, '_getLastSel', out-row-number, out-is-selected);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>out-row-number</td>
<td>N</td>
<td>the row number of the last row selected or deselected</td>
</tr>
<tr>
<td>out-is-selected</td>
<td>N</td>
<td>a value indicating the status of the row:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>^=0 currently selected</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 not selected</td>
</tr>
</tbody>
</table>

## Example

_getLastSel returns the number of the last row selected or deselected by the user:

```plaintext
call notify('table','_get_last_sel_','row','issel');
if (issel) then
put 'The last selected row was'
```
---

### _getMaxcol

Returns the maximum number of columns in the table

---

**Syntax**

CALL NOTIFY (extended-table-name, '_getMaxcol', out-maxcol);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>out-maxcol</td>
<td>N</td>
<td>the maximum number of columns in the table</td>
</tr>
</tbody>
</table>

**Details**

The maximum number of columns is initially set to the maximum size text-entry object that can fit in a table. The ratio of this value to the number of visible columns determines the size of the thumb in the horizontal scroll bar. See also _getViscol in this class.

---

### _getMaxrow

Returns the maximum number of rows in the table

---

**Syntax**

CALL NOTIFY (extended-table-name, '_getMaxrow', out-maxrow);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>out-maxrow</td>
<td>N</td>
<td>the maximum number of rows (negative, if dynamic table)</td>
</tr>
</tbody>
</table>

**Example**

:getMaxrow returns the maximum row in the variable MAXROW:

```plaintext
    call notify('table', '_get_maxrow',
                maxrow);
    if ( maxrow < 0 ) then
```

---
_getMaxsel

Returns the maximum number of selections allowed in a table

Syntax
CALL NOTIFY (extended-table-name, '_getMaxsel', out-maxsel);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>out-maxsel</td>
<td>N</td>
<td>the maximum number of selections allowed</td>
</tr>
</tbody>
</table>

_getNselect

Returns the number of rows currently selected in the table

Syntax
CALL NOTIFY (extended-table-name, '_getNselect', out-selections);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>out-selections</td>
<td>N</td>
<td>the number of rows currently selected.</td>
</tr>
</tbody>
</table>

_getToprow

Returns the number of the row displayed at the top of the table

Syntax
CALL NOTIFY (extended-table-name, '_getToprow', out-toprow);
_getViscol

Returns the number of columns that are currently visible

Syntax

CALL NOTIFY (extended-table-name, '_getViscol', out-viscol);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>out-viscol</td>
<td>N</td>
<td>the number of table columns that are visible</td>
</tr>
</tbody>
</table>

Details

The value returned by the _getViscol method is used when a table is scrolled horizontally by PAGE or HALF. The ratio of the visible columns to the maximum number of columns determines the size of the thumb in the horizontal scroll bar. See _setHscroll in this class.

_getVisrow

Returns the number of rows that are currently visible

Syntax

CALL NOTIFY (extended-table-name, '_getVisrow', out-visrow);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>out-visrow</td>
<td>N</td>
<td>the number of table rows that are visible</td>
</tr>
</tbody>
</table>

Details

The visible rows are the number of logical rows that can be displayed in the region.
_getVscroll

Returns the vertical scrolling unit

Syntax

CALL NOTIFY (extended-table-name, '_getVscroll', out-unit<, out-num-units>);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>out-unit</td>
<td>C</td>
<td>the scrolling unit:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'MAX'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'PAGE'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'HALF'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'COLUMN'</td>
</tr>
</tbody>
</table>
| out-num-units    | N    | the number of logical rows if the scrolling unit is 'ROW'

_getWidgets

Returns the identifier for an SCL list containing all widgets in the current row

Syntax

CALL NOTIFY (extended-table-name, '_getWidgets', out-id);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| out-id   | N    | the identifier for the SCL list containing widget identifiers

Details

Use _getWidgets only in an SCL program's GETROW or PUTROW sections. This method can be used to send methods to objects in the table.

Example

_getWidgets returns the list of widgets in the third row of the table and changes the border color of all widgets to red:

GETROW:
    rc = fetchobs (dsid, _currow_);
    if (_currow_=3) then
    do;
call notify('table', '_get_widgets_', widgets);
do i=1 to listlen(widgets);
    obj=getiteml(widgets,i);
    call send(obj, '_set_border_color_', 'red');
end;
end;
return;

__getRow

Provides a model for a user-defined method that performs the actions that would otherwise be included in the GETROW section of the SCL program

Details

The __getRow method, which is normally used when subclassing extended tables, is not called by SCL code; it is called by the extended table object after the __getRowLabel method runs. The __getRow method establishes and sets the value of objects in an extended table. The value of attributes set in the GETROW section are not retained when users scroll the table, so you must conditionally assign or deassign attributes. This method is provided for compatibility only. When subclassing an extended table, use the __getRowLabel method. For more information on creating a subclass, see SAS/AF online help.

__getRowLabel

Runs the GETROW section of the SCL program

Details

The __getRowLabel method, which is normally used when subclassing extended tables, is not called by SCL code; it is called by the extended table object to establish and set the values of objects in an extended table. By default, the __getRowLabel method runs the GETROW section of the SCL program.

Example

This method can be overridden if you need to perform some pre- or post-processing with respect to the GETROW section. For example, if you want to process some information both before and after the GETROW section of the SCL program runs, you can override the __getRowLabel method of the Extended Table class and write your method like this:
length _method_ $ 40;

GETROW: method;
 /* perform preprocessing */
 /* some SCL statements here; */

 /* run the GETROW section of the SCL
 program */
call super( _self_, _method_ );

 /* perform post-processing */
 /* more SCL statements here; */

demethod;

__hscroll

Scrolls the table horizontally

Syntax

CALL NOTIFY (extended-table-name, '_hscroll', in-unit, in-num-units);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>in-unit</td>
<td>C</td>
<td>the scrolling unit:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'MAX'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'PAGE'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'HALF'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'COLUMN'</td>
</tr>
<tr>
<td>in-num-units</td>
<td>N</td>
<td>the number of units to scroll. To scroll right, specify a positive number. To scroll left, specify a negative number. The default scroll amount is used if in-unit is not specified. See _setHscroll in this class.</td>
</tr>
</tbody>
</table>

Details

Because an extended table is not itself horizontally scrollable, you cannot place objects beyond the boundaries of the extended table's container box. The SCL program provides the scrolling functionality. When users scroll the table, the global variable _curcol is updated, based on the scroll amount, and contains the value of the leftmost column in the table. It is then up to the application to set values of objects correctly, based on the value of _curcol.

Examples

This example scrolls TABLE four pages to the right:
call notify('table','_hscroll_','page',4);

This example scrolls TABLE eight columns to the left:

    call notify('table','_hscroll_','column',-8);

This example scrolls TABLE all the way to the left:

    call notify('table','_hscroll_','max',-1);

---

_issel

Reports whether a specified row is selected

Syntax

CALL NOTIFY (extended-table-name, '_issel', in-row-number, status);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>in-row-number</td>
<td>N</td>
<td>the number of the row to check for selected status</td>
</tr>
<tr>
<td>status</td>
<td>N</td>
<td>a value indicating whether the row is selected:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>^=0 selected</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 not selected</td>
</tr>
</tbody>
</table>

Example

_issel reports whether the third row in the table named TABLE is currently selected:

    call notify('table','_issel_','3',issel);
    if (issel) then
        put "The third row is currently selected";
    else
        put "The third row is NOT currently selected";

---

_needRowRefresh

Marks a row for a later refresh operation

Syntax

CALL NOTIFY (extended-table-name, '_needRowRefresh', row);
Extended Table Class

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>row</td>
<td>N</td>
<td>the row that needs a refresh</td>
</tr>
</tbody>
</table>

**Details**

This method is only available if Putrow Options in the Attributes window is set to Run getrow for all modified rows or Do not run getrow. When either of these options is set, the number of GETROW refreshes is limited. However, there are times when a row needs to be refreshed but the process is prevented by these options.

For example, suppose an extended table consists of protected text-entry objects. The table is being used for displaying information so none of the rows can be modified by the user. In this case, the option, Do not run getrow is enabled. However, if the information being displayed in the table is retrieved from a data set and the application modifies a specific observation in the data set, then the row corresponding to that observation needs to be refreshed.

Refreshing of the row does not occur immediately but rather occurs when the table receives a _refresh method. See “Flow of Control in Extended Tables” on page 1227 for more information on _refresh.

**See Also**

_needRefresh in the Widget class.

---

**_putrow**

Provides a model for a user-defined method that performs the actions that would otherwise be included in the PUTROW section of the SCL program

**Details**

The _putrow method, which is normally used when subclassing extended tables, is not called by SCL code; it is called by the extended table object after the _putrowLabel method runs. The _putrow method is called whenever an object is modified or selected.

This method is provided for compatibility only. When subclassing an extended table, use the _putrowLabel method. For more information on creating a subclass, see SAS/AF online help.

---

**_putrowLabel**

Runs the PUTROW section of the SCL program

**Details**

The _putrowLabel method, which is normally used when subclassing extended tables, is not called by SCL code; it is called by the extended table object whenever an object is
modified or selected in an extended table. By default, the _putrowLabel method runs
the PUTROW section of the SCL program.

Example

This method can be overridden if you need to perform some pre- or post-processing
with respect to the PUTROW section. For example, if you want to process some
information both before and after the PUTROW section of the SCL program runs, you
can override the _putrowLabel method of the Extended Table class and write your
method similar to this:

```
length _method_ $ 40;

PUTROW: method;
    /* perform preprocessing */
    /* some SCL statements here; */

    /* run the PUTROW section of the
    SCL program */
    call super( _self_, _method_ );

    /* perform post-processing */
    /* more SCL statements here; */

endmethod;
```

___

**_selected**

Returns the number of the \(n\)th selected row

----------

**Syntax**

CALL NOTIFY (extended-table-name, '_selected', in-nth-selection, out-row-number);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>in-nth-selection</td>
<td>N</td>
<td>the (n)th row selected</td>
</tr>
<tr>
<td>out-row-number</td>
<td>N</td>
<td>row number of the (n)th selected row</td>
</tr>
</tbody>
</table>

**Example**

This example returns the row number for the next-to-last row selected by the user.
To do this, it first uses the _getNselect method to retrieve the total number of selected
rows. The _selected method uses this number to calculate the next-to-last selected row
(NUM_SELECTED-1), for which it determines the row number and assigns it to the
variable ROW_NUM.

```
call notify('table', '_get_nselect_',
            num_selected);
```
if (num_selected > 2) then do;
call notify('table','_selected_',
    num_selected-1,row_num);
put "The next to last row selected is
    row: " row_num;
end;

__selectRow

Changes the select status of a specified table row

Syntax

CALL NOTIFY (extended-table-name, '_selectRow', in-row-number<, action>);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>in-row-number</td>
<td>N</td>
<td>the row to select or deselect</td>
</tr>
<tr>
<td>action</td>
<td>C</td>
<td>action to apply to the row:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'ON' selects the row (the default)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'OFF' deselects the row</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'TOGGLE' deselects the row if it is selected, and selects the row if it deselected</td>
</tr>
</tbody>
</table>

Details

If the row was not previously selected, _selectRow causes the GETROW section of your SCL program to execute when the extended table receives a _refresh.

__setHscroll

Assigns the unit by which a table scrolls horizontally

Syntax

CALL NOTIFY (extended-table-name, '_setHscroll', in-unit<, in-num-units><, in-num-units>);
### Argument Type Description

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>in-unit</td>
<td>C</td>
<td>the scrolling unit; 'MAX', 'PAGE', 'HALF', 'COLUMN'</td>
</tr>
<tr>
<td>in-num-units</td>
<td>N</td>
<td>number of units if the scrolling unit is 'COLUMN'</td>
</tr>
</tbody>
</table>

#### Details

The horizontal scrolling unit assigned by `_setHscroll` determines the amount by which to scroll when users click either side of the thumb inside the scroll bar.

Because an extended table is not itself horizontally scrollable, you cannot place objects beyond the boundaries of the extended table's container box. The SCL program provides the scrolling functionality. When users scroll the table, the global variable `_curcol` is updated, based on the scroll amount, and contains the value of the leftmost column in the table. It is then up to the application to set values of objects correctly, based on the value of `_curcol`.

---

### `_setMaxcol`

**Assigns the maximum number of table columns**

---

#### Syntax

```sql
CALL NOTIFY (extended-table-name, '_setMaxcol', in-maxcol);
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>in-maxcol</td>
<td>N</td>
<td>the maximum number of columns that can be displayed by scrolling right</td>
</tr>
</tbody>
</table>

#### Details

The ratio of the maximum number of columns to the number of visible columns determines the size of the thumb in the horizontal scroll bar. See `_getViscol` in this class.

`_setMaxcol` causes the GETROW section of your SCL program to execute in the following cases:

- the new maximum column is less than the current column
- the new maximum column affects the current columns that are displayed.
**_setMaxrow**

Assigns the maximum number of table rows

___________

**Syntax**

CALL NOTIFY (extended-table-name, '_setMaxrow', in-maxrow);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>in-maxrow</td>
<td>N</td>
<td>the maximum row</td>
</tr>
</tbody>
</table>

**Details**

This method is for non-dynamic tables. The ratio of the maximum number of rows to the visible rows determines the size of the thumb in the vertical scroll bar. See _getVisrow in this class.

_setMaxrow causes the GETROW section of your SCL program to execute in the following cases:

- the new maximum row is less than the current top row
- the new maximum row affects the current rows that are displayed.

**Example**

_setMaxrow assigns the maximum number of rows that can be displayed in TABLE based on the number of observations in a data set:

```splus
dsid=open('work.temp');
nobs=attrn(dsid, 'nobs');
call notify('table','_set_maxrow_ ',nobs);
```

**_setMaxsel**

Assigns the maximum number of selections allowed in a table

___________

**Syntax**

CALL NOTIFY (extended-table-name, '_setMaxsel', in-maxsel);
### Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>in-maxsel</td>
<td>N</td>
<td>the maximum number of selections</td>
</tr>
</tbody>
</table>

### Details

If you specify a value for `in-maxsel` that is less than the current number of selections, the rows after `in-maxsel` are deselected and the GETROW section runs when the extended table receives a `_refresh`.

---

## `_setToprow`

Assigns a row to display as a table’s top row

### Syntax

```sql
CALL NOTIFY (extended-table-name, '_setToprow', in-toprow);
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>in-toprow</td>
<td>N</td>
<td>the number of the table's top row</td>
</tr>
</tbody>
</table>

### Details

`_setToprow` causes the GETROW section of your SCL program to execute if you specify a value for `in-toprow` that is different from the current top row of the extended table.

### Example

`_setToprow` scrolls the fifth row to the top of the table named TABLE:

```sql
call notify('table', '_set_toprow', 5);
```

---

## `_setViscol`

Assigns the number of columns that are visible

### Syntax

```sql
CALL NOTIFY (extended-table-name, '_setViscol', in-viscol);
```
Extended Table Class

**_setViscol**

Assigns the number of visible columns. This value cannot be greater than the maximum number of columns. See _setMaxcol in this class.

**Details**

_setViscol causes the GETROW section of your SCL program to execute if you specify a value for in-viscol that is different from the current number of visible columns. The ratio of the visible columns to the maximum number of columns determines the size of the thumb in the horizontal scroll bar.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>in-viscol</td>
<td>N</td>
<td>the number of visible columns.</td>
</tr>
</tbody>
</table>

**_setVisrow**

Assigns the number of rows that are visible in a table.

**Syntax**

CALL NOTIFY (extended-table-name, '_setVisrow', in-visrow);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>in-visrow</td>
<td>N</td>
<td>the number of rows visible in a</td>
</tr>
</tbody>
</table>

**Details**

The number of visible rows assigned by _setVisrow is the number of logical rows that can be displayed in the region. _setVisrow causes the GETROW section of your SCL program to execute if you specify a value for in-visrow that is different from the current number of visible rows.

**_setVscroll**

Assigns the unit by which a table scrolls vertically.

**Syntax**

CALL NOTIFY (extended-table-name, '_setVscroll', in-unit<, in-num-units>);
### Argument Type Description

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>in-unit</td>
<td>C</td>
<td>the scrolling unit:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'MAX'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'PAGE'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'HALF'</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'ROW'</td>
</tr>
<tr>
<td>in-num-units</td>
<td>N</td>
<td>the number of units if the scrolling unit is 'ROW'.</td>
</tr>
</tbody>
</table>

### Details

The vertical scroll unit assigned by `_setVscroll` determines the amount by which to scroll when users click either side of the thumb inside the scroll bar. When a user scrolls the table, the global variable `_currow` is updated to the number of the row displayed at the top of the table, based on the scroll amount.

---

### `_unselectAll`

Unselects all rows in the table and executes the GETROW section of the program

#### Syntax

```plaintext
CALL NOTIFY (extended-table-name, '_unselectAll');
```

### `_unselectRow`

Unselects the _n_ th row of the table

#### Syntax

```plaintext
CALL NOTIFY (extended-table-name, '_unselectRow', in-row-number);
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>in-row-number</td>
<td>N</td>
<td>the number of the row to deselect</td>
</tr>
</tbody>
</table>

#### Details

 `_unselectRow` performs the same function as the `_selectRow` method with the OFF parameter. If the row was previously selected, `_unselectRow` causes the GETROW section to execute when the extended table receives a `_refresh`. 
### _vscroll

Scrolls the table vertically

---

**Syntax**

CALL NOTIFY (extended-table-name, '_vscroll', <in-unit>, in-num-units>);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>in-unit</td>
<td>C</td>
<td>the scrolling unit:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>‘MAX’</td>
</tr>
<tr>
<td></td>
<td></td>
<td>‘PAGE’</td>
</tr>
<tr>
<td></td>
<td></td>
<td>‘HALF’</td>
</tr>
<tr>
<td></td>
<td></td>
<td>‘ROW’</td>
</tr>
<tr>
<td>in-num-units</td>
<td>N</td>
<td>the number of units (positive, for scroll down, negative, for scroll up). The default scroll amount is used if in-unit is not specified. See _setVscroll in this class.</td>
</tr>
</tbody>
</table>

**Example**

This statement scrolls TABLE up eight rows:

```plaintext
call notify('table', '_vscroll', 'row', -8);
```

This statement scrolls the table to the top:

```plaintext
call notify('table', '_set_vscroll', 'max', -1);
```

You can also scroll the table to the top using the _setToprow method.