

Installation Instructions

To use the AdvancedDataGrid you need to install AdvancedDataGrid.mxp with Macromedia Flash 2004. To install the AdvancedDataGrid.mxp extension you should use the "Macromedia Extension Manager" (this tool can be downloaded from www.macromedia.com). After completing the installation process, you can find the AdvancedDataGrid component in "Components Panel".

DataGrid Extensions API

NOTE: If at any point you need to use:

```
new DataGridColumn();
```

...then you MUST include this additional library:

```
import tufat.com.controls.gridclasses.DataGridColumn;
```

"gridObj" = the datagrid object reference

"i" = integer value (0 <= i < gridObj.columnCount)

- **Header background of the column. Sets the background color of the DataGrid header.**

Usage

```
gridObj.getColumnAt( i ).setStyleProperty("headerBG", COLOR);
```

Sample

```
gridObj.getColumnAt(0).setStyleProperty("headerBG", 0xFFCC55);
```

- **The "displayPrefix" displayed a string before the data, and "displayPostfix" displays a string immediately after the data.**

Usage

```
gridObj.getColumnAt( i ).setStyleProperty( "displayPrefix", prefix_string );
```

```
gridObj.getColumnAt( i ).setStyleProperty( "displayPostfix", postfix_string );
```

Sample

```
gridObj.getColumnAt(6).setStyleProperty( "displayPrefix", "$" );
```

```
gridObj.getColumnAt(6).setStyleProperty( "displayPostfix", ".00" );
```

- **Show/hide horizontal gridlines. Allows horizontal gridlines to be shown or hidden.**

Usage

```
gridObj.showHorizontalGridLines( boolean_value );
```

Sample

```
gridObj.showHorizontalGridLines( true );
```

- **Word wrap / autosize. "headerWordWrap" allows the text within the header to wrap. Allows the cell text to wrap (in a manner similar to how HTML text wraps in <table> tags).**

Usage

```
gridObj.setStyleProperty( "headerWordWrap", boolean_value );
```

```
gridObj.setStyleProperty( "WordWrap", boolean_value );
```

Sample

```
gridObj.setStyleProperty( "headerWordWrap", true );
```

```
gridObj.setStyleProperty( "WordWrap", true );
```

- **Hide header separator. Allows the header separator to be hidden for any column (the small bar which separates the headers).**

Usage

```
gridObj.getColumnAt( i ).setStyleProperty( "showHeaderSeparator", boolean_value );
```

Sample

```
gridObj.getColumnAt(1).setStyleProperty( "showHeaderSeparator", true );
```

- **Property which allows the user to specify the cell formatting for a particular column**

Usage

```
gridObj.getColumnAt( i ).setStyleProperty( "textFormat", TextFormat_value );
```

Sample

```
var tf = new TextFormat();
```

```
tf.italic = true;
```

```
tf.bold = true;
```

```
tf.underline = true;
```

```
tf.align = "center";
```

```
gridObj.getColumnAt(1).setStyleProperty( "textFormat", tf );
```

- **Preventing of grid header height autosizing (default if you use headerWordWrap property header height is autosizing).**

Usage

```
gridObj.setStyleProperty( "gridHeightAutosizing", boolean_value );
```

Sample

```
gridObj.setStyleProperty( "gridHeightAutosizing", false );
```

- **Manual header height. This allows the height of the header to be set to any specific values, in pixels.**

Usage

```
gridObj.setStyleProperty( "headerHeight", integer_value );
```

Sample

```
gridObj.setStyleProperty( "headerHeight", 85 );
```

- **Allows the user to insert HTML into the grid.**

Usage

```
gridObj.getColumnAt( i ).setStyleProperty( "htmlDrawing", boolean_value);
```

Sample

```
gridObj.getColumnAt(0).setStyleProperty( "htmlDrawing", true);
```

- **Property which allows the user to specify the cell formatting for a particular header**

Usage

```
gridObj.getColumnAt( i ).setStyleProperty( "headerTextFormat", TextFormat_value );
```

Sample

```
var tf = new TextFormat();
```

```
tf.italic = true;
```

```
tf.bold = true;
```

```
tf.underline = true;
tf.align = "center";
gridObj.getColumnAt(1).setStyleProperty( "headerTextFormat", tf );
```

- **Property that allows the user to specify the background color for an editable cell.**

Usage

```
gridObj.setStyleProperty( "editableBackgroundColor", COLOR);
```

Sample

```
gridObj.setStyleProperty( "editableBackgroundColor", 0xFFCC55);
```

- **adding .css styles to the Enhanced DataGrid component**

Usage

```
gridObj.setStyleProperty( "textStyleSheet", styleSheet_value );
```

Sample

```
var css_url = "html_styles.css";
// Create a new style sheet object
var style_sheet = new TextField.StyleSheet();
// Load CSS file and define onLoad handler:
style_sheet.load(css_url);
style_sheet.onLoad = function(ok) {
    if (ok) {
        // If the style sheet loaded without error,
        // then assign it to the text object,
        // and assign the HTML text to the text field.
        gridObj.setStyleProperty( "textStyleSheet", style_sheet );

    }else{trace("LOAD CSS ERROR");}
};
```

- **Vertical alignment for text in column**

Usage

```
gridObj.getColumnAt( i ).setStyleProperty( "valign", valign_value);
```

valign_value is element from set – “center”,”top”,”bottom”.

Sample

```
gridObj.getColumnAt(1).setStyleProperty( "valign", "center");
```

- **Set the background color of cell**

Usage

```
gridObj.setCellStyle( i, j, "background", COLOR );
```

Sample

```
gridObj.setCellStyle( 0, 0, "background", 0xFFFF00 );
```

- **Set the text color of a cell**

Usage

```
gridObj.setCellStyle( i, j, " textcolor", COLOR );
```

Sample

```
gridObj.setCellStyle( 0, 0, " textcolor", 0xFFFF00 );
```

- **Sort feature**

Usage

```
gridObj.sortOption = option;
```

option One or more numbers or names of defined constants, separated by the | (bitwise OR) operator, that change the behavior of the sort from the default. The following values are acceptable for *option*:

- 0 default DataGrid sort behavior
- 1 or Array.CASEINSENSITIVE
- 16 or Array.NUMERIC

Sample

```
gridObj.sortOption = Array.NUMERIC | Array.CASEINSENSITIVE;
```

Note: for NUMERIC sort be sure that your data has the Number type (123 – is the number type, “123” – is the string type).

Assigning a callback function to any cell.

```
var myListener = new Object();
myListener.cellFocusIn = function(event) {
    var cell = "(" + event.columnIndex + ", " + event.itemIndex + ")";
    var msg = "The cell at " + cell + " has gained focus";
    trace( msg );
};
myGrid.addEventListener("cellFocusIn", myListener);
```