API Reference

by Flexmonster

1. API reference

Flexmonster Pivot Table & Charts Component has convenient full-functional JavaScript API to embed the component into web applications.

This API documentation describes objects, calls and events for version 2.3.

In case you are looking for documentation from version 2.2, it is available here: Documentation 2.2(https://s3.amazonaws.com/flexmonster/2.2/docs/Flexmonster-Documentation-2.2.pdf) and API Reference 2.2(https://s3.amazonaws.com/flexmonster/2.2/docs/Flexmonster-API-Reference-2.2.pdf) .

The following example illustrates how to embed the component via initial jQuery call:

```
<div id="pivotContainer">The component will appear here</div>
<script src="flexmonster/lib/jquery.min.js"></script>
<script src="flexmonster/flexmonster.js"></script>

<script>
    var pivot = $("#pivotContainer").flexmonster({
        toolbar: true,
        licenseKey: "XXXX-XXXX-XXXX-XXXX-XXXX"
    });
</script>
```

Open the example on JSFiddle(http://jsfiddle.net/flexmonster/L54jrsp5/) or read more about embedding Flexmonster Component: $.flexmonster(/api/flexmonster) .

List of objects

Report Object(/api/report-object/)	contains all the possible aspects of pivot tables and pivot charts configuration
Slice Object(/api/slice-object/)	defines what data subset from the data source is going to be shown in the report
Options Object(/api/options-object/)	used to specify appearance and functionality available for customers
Format Object(/api/format-object/)	defines the way how numeric values are formatted in the component
Conditional Format Object(/api/conditional-format-object/)	describes conditional formatting rules
Cell Object(/api/cell-object/)	contains information about the cell

List of API calls:

addCalculatedMeasure(/api/addcalculatedmeasure/)	adds calculated measure
addCondition(/api/addcondition/)	adds a conditional formatting rule
addJSON(/api/addjson/)	sets JSON data source

clear(/api/clear/)	clears the component’s data and view
clearFilter(/api/clearfilter/)	clears the filter applied previously to the specified hierarchy
clearXMLACache(/api/clearxmlacache/)	requests Microsoft Analysis Services to clear the cache
closeFieldsList(/api/closefieldslist/)	closes Fields List
collapseAllData(/api/collapsealldata/)	collapses all nodes and drills up all levels of all hierarchies
collapseData(/api/collapsedata/)	collapses all nodes of the specified hierarchy
connectTo(/api/connectto/)	connects to the data source without cleaning the report
customizeCell(/api/customizecell/)	allows customizing of separate cells
dispose(/api/dispose/)	prepares the pivot table instance to be deleted with the browser’s garbage collection
expandAllData(/api/expandalldata/)	expands all nodes and drills down all levels of all hierarchies
expandData(/api/expanddata/)	expands all nodes of the specified hierarchy
exportTo(/api/exportto/)	exports grid or chart to CSV, HTML, PDF, Image or Excel format
fullScreen(/api/fullscreen/)	switches the component to full-screen mode
getAllConditions(/api/getallconditions/)	returns a list of conditional formatting rules of the report
getAllHierarchies(/api/getallhierarchies/)	returns a list of all available hierarchies
getAllMeasures(/api/getallmeasures/)	returns a list of all available measures
getCell(/api/getcell/)	returns information about cell by row and column indexes
getColumns(/api/getcolumns/)	returns a list of hierarchies selected in the report slice for columns
getCondition(/api/getcondition/)	returns a conditional formatting rule by id
getData(/api/getdata/)	retrieves the data that pivot instance is showing or from the selected slice
getFilter(/api/getfilter/)	returns the filtered members for the specified hierarchy
getFilterProperties(/api/getfilterproperties/)	returns the filter properties set for the specified hierarchy
getFormat(/api/getformat/)	returns Format Object(/api/format-object/) of a default number format or the number format for the specified measure
getMeasures(/api/getmeasures/)	returns a list of the selected measures in the report
getMembers(/api/getmembers/)	returns a list of members for the specified hierarchy
getOptions(/api/getoptions/)	returns Options Object(/api/options-object/) with component’s options
getPages(/api/getpages/)	returns a list of hierarchies selected in the report slice for pages (“Report Filter”)

getReport(/api/getreport/)	returns Report Object(/api/report-object/) which describes the current report
getRows(/api/getrows/)	returns a list of hierarchies selected in the report slice for rows
getSelectedCell(/api/getselectedcell/)	returns information about selected cell
getSort(/api/getsort/)	returns the sort type which is applied to the hierarchy
getXMLACatalogs(/api/getxmlacatalogs/)	obtains a list of all available catalogs on a given data source
getXMLACubes(/api/getxmlacubes/)	obtains a list of all available cubes on a given data source
getXMLADataSources(/api/getxmladatasources/)	obtains a list of all data sources by given URL for XMLA connect
getXMLAProviderName(/api/getxmlaprovidername/)	returns dataSourceType for given proxyUrl
load(/api/load/)	loads report JSON file from the specified URL
off(/api/off/)	removes JS handlers for specified event
on(/api/on/)	sets a JS function for the specified event
open(/api/open/)	opens local report file
openFieldsList(/api/openfieldslist/)	opens Fields List
print(/api/print/)	prints the content of the grid or chart via OS print manager
refresh(/api/refresh/)	redraws the component
removeAllCalculatedMeasures(/api/removeallcalculatedmeasures/)	removes all calculated measures
removeAllConditions(/api/removeallconditions/)	removes all conditional formatting rules
removeCalculatedMeasure(/api/removecalculatedmeasure/)	removes the calculated measure by measure unique name
removeCondition(/api/removecondition/)	removes the conditional formatting rule by id
removeSelection(/api/removeselection/)	removes a selection from cells on the grid
runQuery(/api/runquery/)	runs a query with specified Slice Object(/api/slice-object/) and displays the result data
save(/api/save/)	saves your current report to a specified location
setBottomX(/api/setbottomx/)	sets the Bottom X filter for the specified hierarchy and measure
setFilter(/api/setfilter/)	sets the filter for the specified hierarchy
setFormat(/api/setformat/)	sets a default number format or the number format for the specified measure
setOptions(/api/setoptions/)	sets the component’s options

setReport(/api/setreport/)	sets a report to be displayed in the component
setSort(/api/setsort/)	sets the sort type to the specified hierarchy
setTopX(/api/settopx/)	sets the Top X filter for the specified hierarchy and measure
showCharts(/api/showcharts/)	switches to the charts view and shows the chart of the specified type
showGrid(/api/showgrid/)	switches to the grid view
showGridAndCharts(/api/showgridandcharts/)	switches to the grid and charts view and shows the chart of the specified type
sortValues(/api/sortvalues/)	sorts values in a specific row or column in the pivot table
updateData(/api/updatedata/)	updates data for the report without cleaning the report

List of events:

reportcomplete(/api/reportcomplete/)	triggered when the operations can be performed with the component (data source file or OLAP structure was loaded successfully and the grid/chart was rendered)
beforetoolbarcreated(/api/beforetoolbarcreated/)	triggered before the creation of Toolbar
cellclick(/api/cellclick/)	triggered when a cell is clicked on the grid
celldoubleclick(/api/celldoubleclick/)	triggered when a cell is double clicked on the grid
datachanged(/api/datachanged/)	triggered after the user edits data
dataerror(/api/dataerror/)	triggered when some error occurred during the loading of data
datafilecancelled(/api/datafilecancelled/)	triggered when Open file dialog was opened and customer clicks Cancel button
dataloaded(/api/dataloaded/)	triggered when the component loaded data
exportcomplete(/api/exportcomplete/)	triggered after the export file was generated successfully
exportstart(/api/exportstart/)	triggered when the export of grid or chart starts
fieldslistclose(/api/fieldslistclose/)	triggered when the built-in Fields List is closed
fieldslistopen(/api/fieldslistopen/)	triggered when the built-in Fields List is opened
filteropen(/api/filteropen/)	triggered when a filter icon is pressed
loadingdata(/api/loadingdata/)	triggered when data starts loading from from local or remote CSV, JSON or after report was loaded
loadinglocalization(/api/loadinglocalization/)	triggered when localization file starts loading
loadingolapstructure(/api/loadingolapstructure/)	triggered when structure of OLAP cube starts loading
loadingreportfile(/api/loadingreportfile/)	triggered when a report file started loading
localizationerror(/api/localizationerror/)	triggered when some error appeared while loading localization file

localizationloaded(/api/localizationloaded/)	triggered when localization file was loaded
olapstructureerror(/api/olapstructureerror/)	triggered when some error appeared while loading OLAP structure
olapstructureloaded(/api/olapstructureloaded/)	triggered when OLAP structure was loaded
openingreportfile(/api/openingreportfile/)	triggered when customer uses menu Open -> Local report or API method open()(/api/open/) is called
printcomplete(/api/printcomplete/)	triggered after OS print manager was closed
printstart(/api/printstart/)	triggered when print(options:Object)(/api/print/) method was called or Export > Print was selected in the Toolbar
querycomplete(/api/querycomplete/)	triggered after the data query was complete
queryerror(/api/queryerror/)	triggered if some error occurred while running the query
ready(/api/ready/)	triggered when the component's initial configuration completed and the component is ready to receive API calls
reportchange(/api/reportchange/)	triggered when a report is changed in the component
reportfilecancelled(/api/reportfilecancelled/)	triggered when customer uses menu Open -> Local report and clicks Cancel button
reportfileerror(/api/reportfileerror/)	triggered when some error occurred during the loading of the report file
runningquery(/api/runningquery/)	triggered before data query is started
update(/api/update/)	triggered when the change occurred in the component

1.1. $.flexmonster

$("#pivotContainer").flexmonster({
componentFolder: String,
global: ReportObject(/api/report-object/)
,
width: Number,
height: Number,
report: ReportObject(/api/report-object/)
| String,
toolbar: Boolean,
customizeCell: Function,
licenseKey: String
})

[starting from version: 2.3]

Embeds the component into the HTML page.
As a parameter jQuery call gets #pivotContainer – id of the HTML element you would like to have as a container for the component.
This method allows you to insert the component into your HTML page and to provide it with all necessary information for the initialization. This is the first API call you need to know.

Starting from version 2.3 you can preset options for all reports using global object.

Note: Please do not forget to import jQuery and flexmonster.js before you start working with it.

Parameters

- componentFolder – URL of the component's folder which contains all necessary files. Also, it is used as a base URL for report files, localization files, styles and images. The default value for componentFolder is flexmonster/.
- global – object that allows you to preset options for all reports. These options can be overwritten for concrete reports. Object structure is the same as for Report Object(/api/report-object/) .
- width – width of the component on the page (pixels or percent). The default value for width is 100%.
- height – height of the component on the page (pixels or percent). The default value for height is 500.
- report – property to set a report. It can be inline Report Object(/api/report-object/) or URL to report JSON. XML reports are also supported in terms of backward compatibility.
- toolbar – parameter to embed the toolbar or not. Default value is false – without the toolbar.
- customizeCell – function that allows customizing of separate cells.
- licenseKey – the license key.

Event handlers can also be set as properties for the jQuery call. Check the list here(/api/events/) .

All the parameters are optional. If you run $("#pivotContainer").flexmonster() – empty component without the toolbar will be added with the default width and height.

Returns

Object, the reference to the embedded pivot table. If you want to work with multiple instances on the same page use these objects. All API calls are available through them.

After initialization, you can obtain an instance reference of the created component by selector as following: var pivot = $("#pivot").data("flexmonster");

Examples

1) Add the component instance to your web page without toolbar:

```
<div id="pivotContainer">The component will appear here</div>
<script src="flexmonster/lib/jquery.min.js"></script>
<script src="flexmonster/flexmonster.js"></script>

<script>
 var pivot = $("#pivotContainer").flexmonster({
  report: {
   dataSource: {
    filename: "data.csv"
   }
  },
  licenseKey: "XXXX-XXXX-XXXX-XXXX-XXXX"
 });
</script>
```

Check out the example on JSFiddle(http://jsfiddle.net/flexmonster/pc8nzn5z/)
.

2) Add the component with toolbar:

```
<div id="pivotContainer">The component will appear here</div>
<script src="flexmonster/lib/jquery.min.js"></script>
<script src="flexmonster/flexmonster.js"></script>

<script>
 var pivot = $("#pivotContainer").flexmonster({
  toolbar: true,
  report: {
   dataSource: {
    filename: "data.csv"
   }
  },
  licenseKey: "XXXX-XXXX-XXXX-XXXX-XXXX"
 });
</script>
```

Try the example on JSFiddle(http://jsfiddle.net/flexmonster/z9ugmt7c/)
.

3) Get the component instance by selector:

```
<div id="pivot">The component will appear here</div>
<script src="flexmonster/lib/jquery.min.js"></script>
<script src="flexmonster/flexmonster.js"></script>

<button onclick="getRefBySelector()">Get Reference</button>

<script type="text/javascript">
    $("#pivot").flexmonster({
        licenseKey: "XXXX-XXXX-XXXX-XXXX-XXXX",
        width: "100%",
        height: 350,
        toolbar: true
    });

    function getRefBySelector() {
        var pivot = $("#pivot").data("flexmonster");
        pivot.setReport({
            dataSource: {
                filename: "http://www.flexmonster.com/download/data.csv"
            }
        });
    }
</script>
```

Check out the example on JSFiddle(http://jsfiddle.net/flexmonster/xsy3d9xz/)
.

4) Add and operate with multiple instances:

```
<div id="firstPivotContainer">The component will appear here</div>
<div id="secondPivotContainer">The component will appear here</div>
<script src="flexmonster/lib/jquery.min.js"></script>
<script src="flexmonster/flexmonster.js"></script>

<script>
 var pivot1 = $("#firstPivotContainer").flexmonster({
  toolbar: true,
  report: {
   dataSource: {
    filename: "data.csv"
   }
  },
  licenseKey: "XXXX-XXXX-XXXX-XXXX-XXXX"
 });

 var pivot2 = $("#secondPivotContainer").flexmonster({
  toolbar: true,
  report: {
   dataSource: {
    filename: "data2.csv"
   }
  },
  licenseKey: "XXXX-XXXX-XXXX-XXXX-XXXX"
 });
</script>

<button onclick="javascript: swapReports()">Swap Reports</button>
<script>
 function swapReports() {
  var report1 = pivot1.getReport();
  var report2 = pivot2.getReport();

  pivot1.setReport(report2);
  pivot2.setReport(report1);
 }
</script>
```

Try the example on JSFiddle(http://jsfiddle.net/flexmonster/1dLjhu0s/)
.

5) Set event handler via $("#pivotContainer").flexmonster():

```
var pivot = $("#pivotContainer").flexmonster ({
 toolbar: true,
 report: {
  dataSource: {
```

```
   filename: "data.csv"
  }
 },
 licenseKey: "XXXX-XXXX-XXXX-XXXX-XXXX",
 ready : function () {
  console.log("The component was created");
 }
});
```

Open this example on JSFiddle(http://jsfiddle.net/flexmonster/9xzsrymg/)
.

6) How to use customizeCell:

```
$("#pivot-container").flexmonster({
   customizeCell: function(html, data) {
  // change html
  return html;
 },
 ...
});
```

Check out the full example on JSFiddle(http://jsfiddle.net/flexmonster/q1gtwj48/)
.

See also

list of events(/api/events/)

2. Objects

Configure Flexmonster Component the way you want using various objects:

Report Object(/api/report-object/)	contains all the possible aspects of pivot tables and pivot charts configuration
Slice Object(/api/slice-object/)	defines what data subset from the data source is going to be shown in the report
Options Object(/api/options-object/)	used to specify appearance and functionality available for customers
Format Object(/api/format-object/)	defines the way how numeric values are formatted in the component
Conditional Format Object(/api/conditional-format-object/)	describes conditional formatting rules
Cell Object(/api/cell-object/)	contains information about the cell

2.1. Report Object

All the possible aspects of pivot tables and pivot charts configuration can be set via report object. The component supports saving reports and loading of previously saved ones.

Report object has the following properties:

- dataSource – Object. Contains information about data source:
  - browseForFile – Boolean. Defines whether you want to load CSV file from the local file system (true) or not (false). Default value is false. Only for CSV and JSON data source type.
  - catalog – String. The data source catalog name of the OLAP data source (only for "microsoft analysis services", "mondrian", "iccube" data source types).
  - cube – String. Given catalog's cube's name of the OLAP data source (only for "microsoft analysis services", "mondrian", "iccube" data source types).
  - data (starting from v2.2) – Property to set JSON data if it is already on the page. See example in addJSON()(/api/addJSON/)
    .
  - dataSourceInfo – String. The service info of the OLAP data source (only for "microsoft analysis services", "mondrian", "iccube" data source types).
  - dataSourceType – String. Type of data source. The component supports the following types: "microsoft analysis services", "mondrian", "iccube", "csv", "ocsv" and "json".
  - fieldSeparator – String. Defines specific fields separator to split CSV row (only for CSV data source type). There is no need to define it if CSV fields are separated by , or ;. This property is used if another char separates fields. For example, if you use TSV, where tab char is used to separate fields in row, fieldSeparator parameter should be defined as "\t".
  - filename – String. The URL to CSV or JSON file or to server-side script which generates CSV data or JSON data (only for CSV and JSON data source type).
  - ignoreQuotedLineBreaks (starting from v2.1) – Boolean. Indicates whether the line breaks in quotes will be ignored (true) in CSV files or not (false). Default value is true, which makes CSV parsing faster. Set it to false only if your data source has valuable for you line breaks in quotes. Please note that this might slow down CSV parsing a little bit.
  - proxyUrl – String. The path to proxy URL to the OLAP data source, such as Microsoft Analysis Services, Mondrian, icCube (only for "microsoft analysis services", "mondrian", "iccube" data source types).
  - recordsetDelimiter – String. Defines which char is used in CSV to denote the end of CSV row (only for CSV data source type). Default value is "?".
  - subquery – String. The parameter to set the server-side filter which helps to decrease the size of the response from the OLAP cube (only for "microsoft analysis services" and "iccube" data source types). For example, to show reports only for one specific year set subquery the following way: "subquery": "select {[Delivery Date].[Calendar].[Calendar Year].&[2008]} on columns from [Adventure Works]".

- slice – Slice Object(/api/slice-object/)
  . There you can define fields that go to rows, go to columns and go to measures, add filtering, sorting, report filtering, expands and drills.

- options – Options Object(/api/options-object/)
  . Allows configuration of component's UI and functionality for customers.

- conditions – Array of Conditional Format Objects(/api/conditional-format-object/)
  .

- formats – Array of Format Objects(/api/format-object/)
  .

- tableSizes – Object. Contains information about table sizes:
  - columns – Array of objects. Each object is used to save and restore width of some column in report:
    - width – Number. Column width in pixels.
    - idx – Number. Index of the column, starts from 0.
    - tuple – Array. Consists of unique names that identifies the column in the table based on data in it.
    - measure(optional) – String. Measure unique name (this property is defined only if “[Measures]” is selected for columns in the slice). measure is used with tuple and is not set when idx is used.

    Please note, it’s necessary to use either idx or tuple, not both.

  - rows – Array of objects. Each object is used to save and restore height of some row in report:
    - height – Number. Row height in pixels.
    - idx – Number. Index of the row, starts from 0.
    - tuple – Array. Consists of unique names that identifies the row in the table based on data in it.
    - measure(optional) – String. Measure unique name (this property is defined only if “[Measures]” is selected for rows in the slice). measure is used with tuple and is not set when idx is used.

    Please note, it’s necessary to use either idx or tuple, not both.

- localization – Property to set a localization. It can be inline JSON or URL to localization JSON file.

You can get the report using getReport()(/api/getreport/)
API call. To set the report use setReport()(/api/setreport/)
API call.

Example

Example of report object:

```
{
    "dataSource": {
        "dataSourceType": "microsoft analysis services",
        "proxyUrl": "http://olap.flexmonster.com/olap/msmdpump.dll",
        "dataSourceInfo": "WIN-IA9HPVD1RU5",
        "catalog": "Adventure Works DW Standard Edition",
        "cube": "Adventure Works",
        "binary": false
    },
    "slice": {
        "rows": [
            {
                "uniqueName": "[Geography].[Geography]",
                "levelName": "[Geography].[Geography].[State-Province]",
                "filter": {
                    "members": [
                        "[Geography].[Geography].[City].&[Malabar]&[NSW]",
                        "[Geography].[Geography].[City].&[Lavender Bay]&[NSW]",
                        "[Geography].[Geography].[City].&[Matraville]&[NSW]",
                        "[Geography].[Geography].[City].&[Milsons Point]&[NSW]"
```

```
                ],
                "negation": true
            },
            "sort": "asc"
        },
        {
            "uniqueName": "[Sales Channel].[Sales Channel]",
            "sort": "asc"
        }
    ],
    "columns": [
        {
            "uniqueName": "[Measures]"
        }
    ],
    "measures": [
        {
            "uniqueName": "[Measures].[Reseller Order Count]",
            "aggregation": "none",
            "active": true,
            "format": "29mvnel3"
        }
    ],
    "expands": {
        "expandAll": false,
        "rows": [
            {
                "tuple": [
                    "[Geography].[Geography].[City].&[Lane Cove]&[NSW]"
                ]
            }
        ]
    },
    "drills": {
        "drillAll": false,
        "rows": [
            {
                "tuple": [
                    "[Geography].[Geography].[Country].&[Australia]"
                ]
            },
            {
                "tuple": [
                    "[Geography].[Geography].[State-Province].&[NSW]&[AU]"
                ]
            },
            {
                "tuple": [
                    "[Geography].[Geography].[City].&[Darlinghurst]&[NSW]"
                ]
            }
        ]
    }
},
"options": {
```

```
    "viewType": "grid",
    "grid": {
        "type": "compact",
        "title": "",
        "showFilter": true,
        "showHeaders": true,
        "fitGridlines": false,
        "showTotals": true,
        "showGrandTotals": "on",
        "showExtraTotalLabels": false,
        "showHierarchies": true,
        "showHierarchyCaptions": true,
        "showReportFiltersArea": true,
        "pagesFilterLayout": "horizontal"
    },
    "chart": {
        "type": "bar",
        "title": "",
        "showFilter": true,
        "multipleMeasures": false,
        "oneLevel": false,
        "autoRange": false,
        "reversedAxes": false,
        "showLegendButton": false,
        "showAllLabels": false,
        "showMeasures": true,
        "showOneMeasureSelection": true,
        "showWarning": true,
        "activeMeasure": ""
    },
    "configuratorActive": true,
    "configuratorButton": true,
    "configuratorMatchHeight": false,
    "showAggregations": true,
    "showCalculatedValuesButton": true,
    "editing": false,
    "drillThrough": true,
    "showDrillThroughConfigurator": true,
    "sorting": "on",
    "datePattern": "dd/MM/yyyy",
    "dateTimePattern": "dd/MM/yyyy HH:mm:ss",
    "saveAllFormats": false,
    "showDefaultSlice": true,
    "showEmptyData": false,
    "defaultHierarchySortName": "asc",
    "selectEmptyCells": true,
    "showOutdatedDataAlert": false
},
"conditions": [
    {
        "formula": "if(#value < 40, 'trueStyle')",
        "trueStyle": {
            "backgroundColor": "#FFCCFF",
            "color": "#000033",
            "fontFamily": "Arial",
```

```
                "fontSize": 12
            },
            "falseStyle": {}
        }
    ],
    "formats": [
        {
            "name": "29mvnel3",
            "thousandsSeparator": " ",
            "decimalSeparator": ".",
            "decimalPlaces": -1,
            "maxDecimalPlaces": -1,
            "maxSymbols": 20,
            "currencySymbol": "$",
            "currencySymbolAlign": "left",
            "nullValue": "",
            "infinityValue": "Infinity",
            "divideByZeroValue": "Infinity",
            "textAlign": "right",
            "isPercent": false
        }
    ],
    "tableSizes": {
        "columns": [
            {
                "tuple": [],
                "measure": "[Measures].[Reseller Order Count]",
                "width": 182
            }
        ]
    },
    "localization": "loc-ua.json"
}
```

2.2. Slice Object

Slice is a definition of what data subset from the data source is going to be shown in the report. This object has the following properties:

- columns - Array of objects. A list of hierarchies selected in the report slice for columns. Each object can have the following properties:
    - uniqueName - String, hierarchy unique name.
    - caption (optional) - String, hierarchy caption;
    - dimensionName (optional) - String, dimension name;
    - filter (optional) - object with the information on filtering:
        - members - Array of hierarchy's members to be reflected/shown according to the applied filter;
        - negation - Boolean. Represents the filter on hierarchy's members. It tells the component to show the members of hierarchy specified in items (true) or to show all the members of hierarchy except the items (false).
        - measure - Represents the filter on values. The name of the measure on which Top X or Bottom X filter will be applied.
        - quantity - Represents the filter on values. Number of elements to choose for the Top X filter

if type is 'top' or for the Bottom X filter if type is 'bottom'.
        - type - Represents the filter type applied to the hierarchy. It can be: 'none' - filter is not applied to the hierarchy, 'members' - the filter on hierarchy's members is applied, 'top' - the filter Top X is applied on values, 'bottom' - the filter Bottom X is applied on values.
    - levelName (optional) - String. Used to specify the level of the hierarchy that is shown on the grid.
    - sort (optional) - String, sorting type for the members ("asc", "desc" or "unsorted");
    - sortOrder (optional) - Array. Using this property you can set custom sorting for hierarchy members. You can specify sortOrder the following way: [“member_1”, “member_2”, etc.].
- drills (optional) - Object. Stores the information about drill-downs in multilevel hierarchies.
    - drillAll (optional) - Boolean. Indicates whether all levels of all hierarchies in slice will be drilled down (true) or drilled up (false).
    - columns (optional) - Array of objects. It is used to save and restore drilled down columns.
    - rows (optional) - Array of objects. It is used to save and restore drilled down rows.
- expands (optional) - Object. Stores the information about expanded nodes.
    - expandAll (optional) - Boolean. Indicates whether all nodes in the data tree will be expanded (true) or collapsed (false) on the grid and on charts.
    - columns (optional) - Array of objects. It is used to save and restore expanded columns.
    - rows - Array of objects. It is used to save and restore expanded rows.
- measures - Array of objects. A list of selected measures and those which have different properties from default ones. Each object has the following properties:
    - uniqueName - String, measure unique name.
    - active (optional) - Boolean value that defines whether the measure will be in the list of available values but not selected (false) or will be selected for the report (true).
    - aggregation (optional) — String, unique name of aggregation that will be applied to the measure ("sum", "count", "distinctcount", "average", "product", "min", "max", "percent", "percentofcolumn", "percentofrow", "index", "difference", "%difference"). If it is calculated measure, it will be "none".
    - availableAggregations (optional) — Array of strings that represents the list of aggregation functions which can be applied to the current measure. If it is calculated measure, it will be [].
    - caption (optional) - String, measure caption.
    - formula (optional) - String that represents the formula that can contain the following operations: +, -, *, /; other measures can be addressed using measure unique name and aggregation function, for example sum("Price") or max("Order"). Pivot supports the following aggregation functions for CSV, OCSV and JSON data sources: "sum", "count", "distinctcount", "average", "product", "min", "max", "percent", "percentofcolumn", "percentofrow", "index", "difference", "%difference".
    - individual (optional) - Boolean. Defines whether the formula is calculated using raw values (true) or using aggregated values (false). Default value is false.
    - format (optional) - String, name of number formatting.
    - grandTotalCaption (optional) - String, measure grand total caption.
- memberProperties - Array of objects. Each object has the following properties:
    - levelName – String, hierarchy unique name.
    - properties - Array of member properties, which will be shown on the grid. Other available member properties can be accessed through the context menu.
- pages - Array of objects. A list of hierarchies selected in the report slice for pages (“Report Filter”). Each object has the following properties:
    - uniqueName - String. Hierarchy unique name.
    - caption (optional) - String, hierarchy caption;
    - dimensionName (optional) - String, dimension name;
    - filter (optional) - object with the information on filtering:
        - members - Array of hierarchy's members to be reflected/shown according to the applied filter;
        - negation - Boolean. It tells the component to show the members of hierarchy specified in items (true) or to show all the members of hierarchy except the items (false).
    - levelName (optional) - String. Used to specify the level of the hierarchy that is shown on the grid.
    - sort (optional) - String. Sorting type for the members ("asc", "desc" or "unsorted");
    - sortOrder (optional) - Array. Using this property you can set custom sorting for hierarchy members.

You can specify sortOrder the following way: [“member_1”, “member_2”, etc.].

- rows - Array of objects. A list of hierarchies selected in the report slice for rows. Each object can have the following properties:
    - uniqueName - String, hierarchy unique name.
    - caption (optional) - String, hierarchy caption;
    - dimensionName (optional) - String, dimension name;
    - filter (optional) - object with the information on filtering:
        - members - Array of hierarchy's members to be reflected/shown according to the applied filter;
        - negation - Boolean. Represents the filter on hierarchy's members. It tells the component to show the members of hierarchy specified in items (true) or to show all the members of hierarchy except the items (false).
        - measure - Represents the filter on values. The name of the measure on which Top X or Bottom X filter will be applied.
        - quantity - Represents the filter on values. Number of elements to choose for the Top X filter if type is 'top' or for the Bottom X filter if type is 'bottom'.
        - type - Represents the filter type applied to the hierarchy. It can be: 'none' - filter is not applied to the hierarchy, 'members' - the filter on hierarchy's members is applied, 'top' - the filter Top X is applied on values, 'bottom' - the filter Bottom X is applied on values.
    - levelName (optional) - String. Used to specify the level of the hierarchy that is shown on the grid.
    - sort (optional) - String, sorting type for the members ("asc", "desc" or "unsorted");
    - sortOrder (optional) - Array. Using this property you can set custom sorting for hierarchy members. You can specify sortOrder the following way: [“member_1”, “member_2”, etc.].
- sorting (optional) - Object. Defines the sorting for numbers in a specific row and/or column in the pivot table.
    - column - Object. Defines the sorting for numbers in a specific column. Object has 3 properties:
        - tuple - Array. Consists of unique names that identifies the column in the table based on data in it;
        - measure – String. Measure unique name on which sorting will be applied;
        - type - String, sorting type ("asc" or "desc").
    - row - Object. Defines the sorting for numbers in a specific row. Object has 3 properties:
        - tuple - Array. Consists of unique names that identifies the row in the table based on data in it;
        - measure – String. Measure unique name on which sorting will be applied;
        - type - String, sorting type ("asc" or "desc").

Change the slice using runQuery()(/api/runquery/)
API call. Get the slice among with other report parts using getReport()(/api/getreport/)
.

2.3. Options Object

Options object is used to specify appearance and functionality available for customers. It has the following properties:

- viewType - String. Type of view to show: "grid" or "charts" or "grid_charts" (starting from v1.9).
- grid – Object. Contains information about grid:
    - type - String. Selected grid type. The following grid types are supported: "compact", "classic" and "flat".
    - title - String. Title of the grid.
    - showFilter - Boolean. Indicates whether the opening columns/rows filter controls and page filter controls are visible (true) or not (false) on the grid. Default value is true.
    - showHeaders - Boolean. Indicates whether the spreadsheet headers are visible (true) or not (false).

    - fitGridlines - Boolean. Indicates whether the gridlines are shown for all cells (false) or only non-empty (true).
    - showTotals - Boolean. Indicates whether the totals are visible (true) or not (false).
    - showGrandTotals - String. Specifies how to show grand totals: in rows ("rows"), in columns ("columns"), in rows and columns ("on") or hide them ("off"). Default value is "on".
    - showHierarchies - Boolean. Specifies how to show drillable hierarchy cells on the grid: with link on the right (true) or with icon on the left (false). Default value is true.
    - showHierarchyCaptions - Boolean. Indicates whether the hierarchy captions are visible (true) or not (false) on the grid. Default value is true.
    - showReportFiltersArea (starting from v2.2) - Boolean. Indicates whether the reports filtering cells on the grid should be visible (true) or not (false). Default value is true.
    - pagesFilterLayout (starting from v2.3) - String. Allows to choose the layout for the report filters. Possible values: "horizontal" and "vertical". Default value is "horizontal".
    - drillthroughMaxRows (starting from v2.318) - Number. Allows setting the maximal number of rows for the MSAS Drill Through popup. Supported only via XMLA protocol. The default value is 1000.
- chart – Object. Contains information about charts:
    - type - String. Selected chart type. The following chart types are supported: "bar", "bar_h" (Horizontal Bar), "line", "scatter", "pie", "bar_stack" and "bar_line" (starting from v1.9).
    - title - String. Title of the chart.
    - showFilter (starting from v2.2) - Boolean. Indicates whether the opening columns and rows filter controls are visible (true) or not (false) on the charts. Default value is true.
    - multipleMeasures - Boolean. Starting from v1.9. Indicates whether to show multiple measures on charts. Default value is false.
    - oneLevel - Boolean. In a case of a drillable chart, defines whether the chart shows all nodes on the x-axis and the legend (false) or only the lowest expanded node on the x-axis and on the legend (true). Default value is false.
    - autoRange - Boolean. Indicates whether the range of values on the charts is selected automatically or not.
    - showLegendButton (starting from v2.2) - Boolean. Indicates whether the button to show/hide the legend on charts is visible. Default value is false which means that the legend is always visible, without the button that hides it.
    - showAllLabels - Boolean. Setting a value to true allows showing all the labels in Pie ?hart. If the value is false it will have the same behavior as it was before. Default value is false.
    - showMeasures (starting from v2.2) - Boolean. Hides all dropdowns on the top of charts if you want to show simple chart without controls or you want to save space. Default value is true - the dropdowns are visible by default, as it was in previous versions.
    - showOneMeasureSelection - Boolean. The default value is true, which means that the visibility of the measures dropdown on charts does not depend on the amount of measures in it. If the value is set to false, the measures dropdown on charts will be hidden if there is only one measure in the list and visible if there are two or more measures.
    - showWarning - Boolean. Indicates whether the warning are shown if data is too big for charts.
    - position - String. Position of charts related to the grid. It can be "bottom", "top", "left" or "right". Default value is "bottom".
    - activeMeasure - String. Selected measure on charts view.
    - pieDataIndex - String. Selected tuple index on the Pie chart.
- configuratorActive - Boolean. Indicates whether the Fields List is opened (true) or closed (false).
- configuratorButton - Boolean. Indicates whether the Fields List toggle button is visible (true) or not (false).
- configuratorMatchHeight (starting from v2.1) - Boolean. Indicates whether the Fields List will be the same height as the component (true) or it's height will be defined by its content amount (false). Default value is false.
- showAggregations (starting from v2.0) - Boolean. Indicates whether the aggregation selection control is visible (true) or not (false) for measures on Fields List. Default value is true.
- showCalculatedValuesButton (starting from v2.2) - Boolean. Controls the visibility of "Add calculated value" in Fields List. Default value is true.
- grouping - Boolean. Indicates whether grouping is enabled. Default value is false. This feature allows

customers to group chosen elements using filter window. For example, the customer has shops in different cities and wants to analyze sales information. It is possible to combine several cities in one group by geography or by the sales numbers, etc. Only for "ocsv", "csv" and "json" data source types.
- editing (starting from v2.1) - Boolean. Indicates whether the editing feature is enabled (true) or disabled (false) on the Drill Through popup for CSV, OCSV and JSON data sources. User will be able to double-click the cell and enter new value in it if the editing feature is enabled.
- drillThrough (starting from v2.1) - Boolean. Indicates whether the drill through feature is enabled (true) or disabled (false). User can drill through by double-clicking the cell with value. Drill through feature is available for all data sources except icCube. Default value is true.
- showDrillThroughConfigurator - Boolean. Indicates whether the Fields List toggle button is visible in Drill Through view. Default value is true.
- sorting (starting from v2.0) - String. Indicates whether the sorting controls are visible in rows ("rows"), in columns ("columns"), in rows and columns ("on" or true) on the grid cells or not visible ("off" or false). Default value is "on".
- datePattern - String. It is used to format "date string" date fields ("type":"date string" in JSON, "ds+" prefix in CSV). A default pattern string is dd/MM/yyyy.
- dateTimePattern - String. It is used to format "datetime" date fields ("type":"datetime" in JSON, "dt+" prefix in CSV). A default pattern string is dd/MM/yyyy HH:mm:ss.
- saveAllFormats - Boolean. If there are more than 5 formats defined, only the formats that are used for "active=true" measures will be saved in the report. In order to get saved all the formats, no matter how many of them you have and whether they are used for active measures or not, please set saveAllFormats property to true. Default value is false.
- showDefaultSlice (starting from v2.2) - Boolean. Defines whether the component selects a default slice for the report with empty slice (when nothing is set in rows, columns, pages and measures). If true, the first hierarchy from data goes to rows and the first measure goes to columns in the default slice. To avoid this default behavior, please set this property to false. Default value is true. Only for "csv", "ocsv" and "json" data source types.
- useOlapFormatting (optional) - Boolean. Indicates whether the values from data source will be formatted according to the format defined in the cube (true) or not (false). Default value is false.
- showMemberProperties - Boolean. Indicates whether the member properties for OLAP data source are available in the component (true) or not (false). Default value is false. This feature is only for "microsoft analysis services" and "mondrian" data source types.
- showEmptyData - Boolean. By default, if you have an empty CSV data source but the header is defined the component will show your slice with empty data cells. Set the value as false - the Component will show the “Data source is empty. Please check the CSV file.” message.
- defaultHierarchySortName (starting from v2.0) - String. Sorting type for hierarchies' members ("asc", "desc" or "unsorted"). Default value is "asc".
- selectEmptyCells (starting from v2.3) - Boolean. Indicates whether it is possible to select cells outside of the table. Default value is true.
- showOutdatedDataAlert - Boolean. Setting a value to true will show the warning to the user before automatic reloading of data from the cube. Default value is false which means there will be no warnings. Only for Flexmonster Accelerator.
- showAggregationLabels - Boolean. Indicates whether aggregation labels like "Total Sum of", "Sum of", etc. are shown in the columns/rows title. Default value is true.

Get Options Object via getOptions()(/api/getoptions/)
API call. Set this object via setOptions()(/api/setoptions/)
.

2.4. Format Object

Format object defines the way how numeric values are formatted in the component. It contains the following number format parameters:

- name – String. It identifies the format in the report, thus, it should be unique. The default is "", which means that this number format is a default one and it is applied to all the measures for which the specific number format is not set.
- thousandsSeparator – String. The default is " " (space).
- decimalSeparator – String. The default is ".".
- decimalPlaces – Number. The exact number of decimals to show in the fractional part of a number after the decimal separator. The default is -1, which means that the number will be shown as is.
- maxDecimalPlaces – Number. The maximum number of decimals to show in the fractional part of a number after the decimal separator. The default is -1, which means the number will be shown as is.
- maxSymbols – Number. The maximum number of symbols in a cell. The default is 20.
- currencySymbol – String. The symbol which is shown near the value (currency symbol, hours, percent, etc.). The default is "".
- currencySymbolAlign – String. The alignment of the currency symbol. It can be "left" or "right". The default is "left".
- isPercent – Boolean. It allows to format data as percentage. The behavior is the same as in Excel. The default is false. Set isPercent to true and numbers will be multiplied by 100 and % symbol will be added. For example, 0.56 will be changed to 56%. Please note, if % is set as currencySymbol, setting isPercent to true will not multiply numbers by 100.
- nullValue – String. It defines how to show null values in the grid. The default is "".
- infinityValue – String. It defines how to show infinity values in the grid. The default is "Infinity".
- divideByZeroValue – String. It defines how to show divided by zero values in the grid. The default is "Infinity".
- textAlign – String. The alignment of formatted values in cells on the grid: "right" or "left". The default is "right".

Set Format Object via setFormat()(/api/setformat/)
API call. Get this object using getFormat()(/api/getformat/)
.

2.5. Conditional Format Object

Conditional format object describes conditional formatting rules. It has the following properties:

- formula - IF statement with 3 arguments: IF(CONDITION, TRUE STYLE, FALSE STYLE), where the false style is optional. Condition can contain AND, OR, ==, !=, >, <, >=, <=, +, -, *, /, isNaN(), !isNaN(). #value is used to address the cell value in condition. Example: if(#value > 2, "trueStyle", "falseStyle").
- trueStyle - style object that will be applied to a cell if the condition for the cell value is met. Please note that for export to Excel and PDF colors should be set as hex color codes.
- falseStyle (optional) - style object. If it is set it will be applied to a cell if the condition for the cell value is not met.
- id (optional) - id of the conditional formatting rule. If id property is not set, the id for the rule will be set inside Pivot component.
- row (optional) - row index to which the condition should be applied.
- column (optional) - column index to which the condition should be applied.
- measure (optional) - measure unique name to which the condition should be applied.
- hierarchy (optional) - hierarchy unique name to which the condition should be applied.
- member (optional) - member unique name to which the condition should be applied.
- isTotal (optional) - Boolean. If it is not defined, the condition will be applied to all cells. If it is true, the condition will be applied to totals and subtotals cells only. If it is false, the condition will be applied to regular cells only.

To add new conditional formatting rule use addCondition()(/api/addcondition/)
API call.

2.6. Cell Object

Object which contains information about the cell. It has the following properties:

- columnIndex – Number. Index of the cell column.
- rowIndex – Number. Index of the cell row.
- label – String. Cell label.
- value – Number. Cell value.
- type – String. Type of the cell. Can be "header" or "value".
- isDrillThrough – Boolean. Indicates whether the cell is from the grid (false) or from the drill through pop-up (true).
- isTotal – Boolean. Identifies whether the cell contains total value.
- hierarchy – Object. Hierarchy connected with the cell.
- measure – Object. Measure connected with the cell.
- member – Object. Member connected with the cell.
- rows – Array of objects. Cell row tuple.
- columns – Array of objects. Cell column tuple.
- height – Number. Cell height in px.
- width – Number. Cell width in px.
- x – Number. Absolute X position of the cell on the page.
- y – Number. Absolute Y position of the cell on the page.

Use getCell()(/api/getcell/)
API call to get information about the cell by row and column indexes or getSelectedCell()(/api/getselectedcell/) to get the selected cell.

3. Methods

Flexmonster Pivot Table & Charts Component offers plenty of methods. All of them are available through the reference to each component instance created by flexmonster()(/api/flexmonster/)
API call.

List of API calls:

addCalculatedMeasure(/api/addcalculatedmeasure/)	adds calculated measure
addCondition(/api/addcondition/)	adds a conditional formatting rule
addJSON(/api/addjson/)	sets JSON data source
clear(/api/clear/)	clears the component’s data and view
clearFilter(/api/clearfilter/)	clears the filter applied previously to the specified hierarchy
clearXMLACache(/api/clearxmlacache/)	requests Microsoft Analysis Services to clear the cache
closeFieldsList(/api/closefieldslist/)	closes Fields List
collapseAllData(/api/collapsealldata/)	collapses all nodes and drills up all levels of all hierarchies
collapseData(/api/collapsedata/)	collapses all nodes of the specified hierarchy
connectTo(/api/connectto/)	connects to the data source without cleaning the report

customizeCell(/api/customizecell/)	allows customizing of separate cells
dispose(/api/dispose/)	prepares the pivot table instance to be deleted with the browser's garbage collection
expandAllData(/api/expandalldata/)	expands all nodes and drills down all levels of all hierarchies
expandData(/api/expanddata/)	expands all nodes of the specified hierarchy
exportTo(/api/exportto/)	exports grid or chart to CSV, HTML, PDF, Image or Excel format
fullScreen(/api/fullscreen/)	switches the component to full-screen mode
getAllConditions(/api/getallconditions/)	returns a list of conditional formatting rules of the report
getAllHierarchies(/api/getallhierarchies/)	returns a list of all available hierarchies
getAllMeasures(/api/getallmeasures/)	returns a list of all available measures
getCell(/api/getcell/)	returns information about cell by row and column indexes
getColumns(/api/getcolumns/)	returns a list of hierarchies selected in the report slice for columns
getCondition(/api/getcondition/)	returns a conditional formatting rule by id
getData(/api/getdata/)	retrieves the data that pivot instance is showing or from the selected slice
getFilter(/api/getfilter/)	returns the filtered members for the specified hierarchy
getFilterProperties(/api/getfilterproperties/)	returns the filter properties set for the specified hierarchy
getFormat(/api/getformat/)	returns Format Object(/api/format-object/) of a default number format or the number format for the specified measure
getMeasures(/api/getmeasures/)	returns a list of the selected measures in the report
getMembers(/api/getmembers/)	returns a list of members for the specified hierarchy
getOptions(/api/getoptions/)	returns Options Object(/api/options-object/) with component's options
getPages(/api/getpages/)	returns a list of hierarchies selected in the report slice for pages ("Report Filter")
getReport(/api/getreport/)	returns Report Object(/api/report-object/) which describes the current report
getRows(/api/getrows/)	returns a list of hierarchies selected in the report slice for rows
getSelectedCell(/api/getselectedcell/)	returns information about selected cell
getSort(/api/getsort/)	returns the sort type which is applied to the hierarchy
getXMLACatalogs(/api/getxmlacatalogs/)	obtains a list of all available catalogs on a given data source
getXMLACubes(/api/getxmlacubes/)	obtains a list of all available cubes on a given data source
getXMLADataSources(/api/getxmladatasources/)	obtains a list of all data sources by given URL for XMLA connect

getXMLAProviderName(/api/getxmlaprovidername/)	returns dataSourceType for given proxyUrl
load(/api/load/)	loads report JSON file from the specified URL
off(/api/off/)	removes JS handlers for specified event
on(/api/on/)	sets a JS function for the specified event
open(/api/open/)	opens local report file
openFieldsList(/api/openfieldslist/)	opens Fields List
print(/api/print/)	prints the content of the grid or chart via OS print manager
refresh(/api/refresh/)	redraws the component
removeAllCalculatedMeasures(/api/removeallcalculatedmeasures/)	removes all calculated measures
removeAllConditions(/api/removeallconditions/)	removes all conditional formatting rules
removeCalculatedMeasure(/api/removecalculatedmeasure/)	removes the calculated measure by measure unique name
removeCondition(/api/removecondition/)	removes the conditional formatting rule by id
removeSelection(/api/removeselection/)	removes a selection from cells on the grid
runQuery(/api/runquery/)	runs a query with specified Slice Object(/api/slice-object/) and displays the result data
save(/api/save/)	saves your current report to a specified location
setBottomX(/api/setbottomx/)	sets the Bottom X filter for the specified hierarchy and measure
setFilter(/api/setfilter/)	sets the filter for the specified hierarchy
setFormat(/api/setformat/)	sets a default number format or the number format for the specified measure
setOptions(/api/setoptions/)	sets the component’s options
setReport(/api/setreport/)	sets a report to be displayed in the component
setSort(/api/setsort/)	sets the sort type to the specified hierarchy
setTopX(/api/settopx/)	sets the Top X filter for the specified hierarchy and measure
showCharts(/api/showcharts/)	switches to the charts view and shows the chart of the specified type
showGrid(/api/showgrid/)	switches to the grid view
showGridAndCharts(/api/showgridandcharts/)	switches to the grid and charts view and shows the chart of the specified type
sortValues(/api/sortvalues/)	sorts values in a specific row or column in the pivot table

updateData(/api/updatedata/)	updates data for the report without cleaning the report

3.1. addCalculatedMeasure

addCalculatedMeasure(measure:Object)

[starting from version: 2.3]

This API call adds calculated measure. Calculated measure has formula property. If it is defined, the measure is calculated. You can create as many calculated measures for one report as you need, there is no limit towards the number of calculated measures. When you save the report all the calculated measures will be saved as well and loaded when report is retrieved. **Please note that you can add calculated measures only for reports based on JSON, CSV and OCSV data sources.**

Parameters

measure – the object that describes measure. This object has the following parameters:

- uniqueName – measure unique name, this property will be used as an identifier for the measure inside Pivot component and as the identifier to remove calculated measure via API.
- caption (optional) – measure caption.
- formula – string that represents the formula that can contain the following operations: +, -, *, /; the following operators:isNaN(), !isNaN(); other measures can be addressed using measure unique name and aggregation function, for example sum("Price") or max("Order"). Pivot supports the following aggregation functions for CSV, OCSV and JSON data sources: "sum", "count", "distinctcount", "average", "product", "min", "max", "percent", "percentofcolumn", "percentofrow", "index", "difference", "%difference".
- grandTotalCaption (optional) – measure grand total caption.
- active (optional) – boolean value that defines whether the calculated measure will be added to the list of available values but not selected (false) or will be selected for the report (true).
- individual (optional) – Boolean. Defines whether the formula is calculated using raw values (true) or using aggregated values (false). Default value is false.
- format (optional) – number formatting name.

Example

The following example shows the calculated measure Average Quantity which is calculated as sum("Quantity")/count("Quantity") and will be added to the list of available measures but not selected for the report (active: false):

```
var measure = {
  formula: 'sum("Quantity")/count("Quantity")',
  uniqueName: "Average Quantity",
  caption: "Average Quantity",
  grandTotalCaption: "Total Quantity",
  active: false
};
flexmonster.addCalculatedMeasure(measure);
```

Open the example on JSFiddle(https://jsfiddle.net/flexmonster/79keh0ee/)
.

See also

removeCalculatedMeasure(/api/removecalculatedmeasure/)

getAllMeasures(/api/getallmeasures/)

getMeasures(/api/getmeasures/)

3.2. addCondition

addCondition(condition:Conditional Format Object(/api/conditional-format-object/)
)

[starting from version: 1.5]

Adds a conditional formatting rule for cell values to format them with specific styles if the condition for the cell value is met.

Use refresh() API call after to redraw the component and see changes.

Parameters

Conditional Format Object(/api/conditional-format-object/)
– the object that describes the conditional formatting rule.

Examples

1) If cell value is more than 400000, then apply trueStyle to this cell:

```
var condition = {
  id: 1,
  formula: 'if(#value > 400000, "trueStyle")',
  trueStyle: {fontSize : 10, backgroundColor: "#33BB33"}
};
flexmonster.addCondition(condition);
flexmonster.refresh();
```

Open the example on JSFiddle(http://jsfiddle.net/flexmonster/18ppLy4z/)
.

2) This rule will be applied only to Sales measure totals and subtotals cells. If Sales is between 100000000 and 200000000, then apply trueStyle to the cell, else apply falseStyle.

```
var condition = {
  id: 2,
  measure: "Sales",
  isTotal: true,
  formula: 'if(AND(#value > 100000000, #value < 200000000), "trueStyle", "falseStyle
")',
```

```
  trueStyle: {fontSize : 11, backgroundColor: "#00FF00"},
  falseStyle: {fontSize : 11, color: "#000000"}
};
flexmonster.addCondition(condition);
flexmonster.refresh();
```

Check how it works on JSFiddle(http://jsfiddle.net/flexmonster/v9qxnt0o/)
.

3) If cell value is empty, then apply trueStyle to this cell:

```
var condition = {
  id: 1,
  formula: 'if(isNaN(#value), "trueStyle")',
  trueStyle: {backgroundColor: "#FFFF11"}
};
flexmonster.addCondition(condition);
flexmonster.refresh();
```

Open the example on JSFiddle(http://jsfiddle.net/flexmonster/310dvxtq/)
.

See also

getCondition(/api/getcondition/)

getAllConditions(/api/getallconditions/)

removeCondition(/api/removecondition/)

removeAllConditions(/api/removeallconditions/)

refresh(/api/refresh/)

3.3. addJSON

addJSON(json:Array)

[starting from version: 2.2]

Sets JSON data source.

Parameters

json – Array of objects. The component supports a certain format of JSON – array of objects, where each object is an unordered set of name/value pairs. In addition, the first object can be used to define data types, captions, etc.

Here is the list of supported properties that can be used in the first object of input array:

- type – data type. Can be:
  - "string" – field contains string data. You will be able to aggregate it only with Count and Distinct Count aggregations. It will be sorted as string data.
  - "number" – field contains numeric data. You will be able to aggregate it with all different aggregations. It will be sorted as numeric data.
  - "level" – field is a level of hierarchy. This type is used together with other properties such as: hierarchy, level and parent
  - "month" – field contains months.
  - "weekday" – field contains days of the week.
  - "date" – field is a date. Such field will be split into 3 different fields: Year, Month, Day.
  - "date string" – field is a date. Such field will be formatted using date pattern (default is dd/MM/yyyy).
  - "year/month/day" – field is a date. You will see such date as a hierarchy: Year > Month > Day.
  - "year/quarter/month/day" – field is a date. You will see such date as a hierarchy: Year > Quarter > Month > Day.
  - "time" – field is a time (numeric data). Such field will be formatted using HH:mm pattern. Min, Max, Count and Distinct Count aggregations can be applied to it.
  - "datetime" – field is a date (numeric data). Such field will be formatted using dd/MM/yyyy HH:mm:ss pattern. Min, Max, Count and Distinct Count aggregations can be applied to it.
  - "id" – field is an id of the fact. Such field is used for editing data. This field will not be shown in Fields List.
  - "hidden" – field is hidden. This field will not be shown in Fields List.
- caption – hierarchy caption.
- hierarchy – hierarchy name, if the field is a level of hierarchy ("type":"level").
- level – caption of the level, if the field is a level of hierarchy ("type":"level").
- parent – caption of the parent level, if the field is a level of hierarchy ("type":"level").
- dimensionUniqueName – dimension unique name. Can be used to group several fields under one dimension.
- dimensionCaption – dimension caption. Is used as a display name (folder name in Fields List) when several fields are grouped under one dimension.

Examples

1) To add JSON data using jQuery call:

```
/**
* The data can be set directly in jQuery call
*/
var pivot = $("#pivotContainer").flexmonster({
  toolbar: true,
  report: {
    dataSource: {
      data: [
        {"Category" : "Accessories", "Color" : "green", "Quantity" : 22}
      ]
    }
  }
});
```

Open the example on JSFiddle(https://jsfiddle.net/flexmonster/3wgbb191/)

.

2) To add JSON data using addJSON() API call. Please pay your attention that the first element of JSON in this example is used to define data types, captions, group fields under one dimension, define hierarchy with 3 levels.

```
var jsonData = [
  {
    "Color":{"type": "string"},
    "M":{"type": "month",
        "dimensionUniqueName": "Days",
        "dimensionCaption": "Days",
        "caption":"Month"},
    "W":{"type": "weekday",
        "dimensionUniqueName": "Days",
        "dimensionCaption": "Days",
        "caption":"Week Day"},
    "Country":{"type":"level",
        "hierarchy": "Geography",
        "level":"?ountry"},
    "State":{"type":"level",
        "hierarchy": "Geography",
        "level":"State",
        "parent": "?ountry"},
    "City":{"type":"level",
        "hierarchy": "Geography",
        "parent": "State"},
    "Price":0,
    "Quantity":{"type": "number"}
  },
{
    "Color":"green",
    "M":"September",
    "W":"Wed",
    "Country" : "Canada",
    "State" : "Ontario",
    "City" : "Toronto",
    "Price":174,
    "Quantity":22
  },
  {
    "Color":"red",
    "M":"March",
    "W":"Mon",
    "Time":"1000",
    "Country" : "USA",
    "State" : "California",
    "City" : "Los Angeles",
    "Price":1664,
    "Quantity":19
  },
  {
    "Color":"red",
    "M":"January",
    "W":"Mon",
    "Country" : "Canada",
```

```
  "State" : "Quebec",
  "City" : "Montreal",
  "Price":1190,
  "Quantity":292
},
{
  "Color":"green",
  "D":"04/08/2014",
  "M":"August",
  "W":"Fri",
  "Time":"1000",
  "Country" : "USA",
  "State" : "California",
  "City" : "Los Angeles",
  "Price":1222,
  "Quantity":730
},
{
  "Color":"white",
  "M":"March",
  "W":"Wed",
  "Country" : "USA",
  "State" : "California",
  "City" : "Los Angeles",
  "Price":7941,
  "Quantity":73
},
{
  "Color":"red",
  "M":"August",
  "W":"Wed",
  "Country" : "Canada",
  "State" : "Ontario",
  "City" : "Toronto",
  "Price":6829,
  "Quantity":19
},
{
  "Color":"green",
  "M":"January",
  "W":"Wed",
  "Country" : "Canada",
  "State" : "Quebec",
  "City" : "Montreal",
  "Price":2995,
  "Quantity":98
},
{
  "Color":"white",
  "M":"February",
  "W":"Mon",
  "Country" : "USA",
  "State" : "California",
  "City" : "Los Angeles",
  "Price":2471,
```

```
    "Quantity":93
  },
  {
    "Color":"yellow",
    "M":"March",
    "W":"Fri",
    "Country" : "Canada",
    "State" : "Ontario",
    "City" : "Toronto",
    "Price":6650,
    "Quantity":40
  },
  {
    "Color":"blue",
    "M":"February",
    "W":"Wed",
    "Country" : "Canada",
    "State" : "Ontario",
    "City" : "Toronto",
    "Price":865,
    "Quantity":45
  },
  {
    "Color":"purple",
    "M":"August",
    "W":"Wed",
    "Country" : "Canada",
    "State" : "Quebec",
    "City" : "Montreal",
    "Price":511,
    "Quantity":46
  },
  {
    "Color":"blue",
    "M":"September",
    "W":"Mon",
    "Country" : "Canada",
    "State" : "Quebec",
    "City" : "Montreal",
    "Price":981,
    "Quantity":18
  },
  {
    "Color":"blue",
    "M":"September",
    "W":"Fri",
    "Country" : "Canada",
    "State" : "Ontario",
    "City" : "Toronto",
    "Price":284,
    "Quantity":24
  }
];

/**
```

```
* The data can be set using addJSON() and setReport() API calls
* only after component is ready
*/
var pivot = $("#pivotContainer").flexmonster({
  toolbar: true,
  ready: function () {
    pivot.addJSON(jsonData);
  }
});

/**
* flexmonsterAddJSON() function illustrates how to set JSON data
* that is already on the page and define a slice based on this data.
*/
function flexmonsterAddJSON() {
  flexmonster.addJSON(jsonData);
  var slice = {
    rows: [{ uniqueName: "Color" }],
    columns: [{ uniqueName: "W" }, { uniqueName: "[Measures]" }],
    measures: [{ uniqueName: "Price"}]};
  flexmonster.runQuery(slice);
}
```

Try on JSFiddle(https://jsfiddle.net/flexmonster/qtspxcoh/)
.

3) To add JSON data using setReport() API call:

```
/**
* The data can be set using addJSON() and setReport() API calls
* only after component is ready
*/
var pivot = $("#pivotContainer").flexmonster({
  toolbar: true,
  ready: function () {
    flexmonsterSetReport();
  }
});

/**
* flexmonsterSetReport() function illustrates how to compose report
* based on JSON data that is already on the page.
*/
function flexmonsterSetReport() {
  var report = {
    dataSource:
    {
     data: jsonData
    },
    slice:
    {
     rows: [{ uniqueName: "M" }],
     columns: [{ uniqueName: "Geography" }, { uniqueName: "[Measures]" }],
```

```
     measures: [{ uniqueName: "Quantity"}]},
    options:
    {
     configuratorActive: true,
     viewType: "grid"
    }
  };
  flexmonster.setReport(report);
}
```

Check out on JSFiddle(https://jsfiddle.net/flexmonster/vspzdy59/)
.

4) Open local JSON file:

```
/**
* connectLocalJSONHandler() function
* illustrates how to connect to local JSON data in file.
*/
function connectLocalJSONHandler() {
  flexmonster.connectTo({ dataSourceType: "json", browseForFile: true });
}
```

5) Connect to remote JSON file:

```
/**
* connectRemoteJSONHandler() function
* illustrates how to connect to remote JSON file.
*/
function connectRemoteJSONHandler() {
  var url = "Url to remote JSON file";
  flexmonster.connectTo({ dataSourceType: "json", filename: url });
}
```

3.4. addMeasure

[removed]

addMeasure method was removed in version 2.3. To add calculated measure you can use addCalculatedMeasure()(/api/addcalculatedmeasure/)
.

3.5. addStyleToMember

[removed]

addStyleToMember method was removed in version 2.3.

3.6. addUrlToMember

[removed]

addUrlToMember method was removed in version 2.3.

3.7. clear

clear()

[starting from version: 1.6]

Clears the component's data and view.

Example

```
flexmonster.clear();
```

Open the example on JSFiddle(https://jsfiddle.net/flexmonster/y6d1re1r/)
.

3.8. clearFilter

clearFilter(hierarchyName:String)

[starting from version: 1.4]

Clears the filter which was applied previously to the specified hierarchy.

Parameters

- hierarchyName – the name of the hierarchy.

Example

```
var filter = [
    "category.[cars]",
    "category.[bikes]"
];
flexmonster.setFilter('Category', filter);
flexmonster.getFilter('Category');

/*
method getFilter() returns the following Array
[
  {'caption' : 'Cars', 'hierarchyName' : 'Category', 'uniqueName' : 'category.[cars]
'},
  {'caption' : 'Bikes', 'hierarchyName' : 'Category', 'uniqueName' : 'category.[bike
s]'}
```

```
]
*/

flexmonster.getFilterProperties('Category');
/*
method getFilterProperties() returns the following object, where type is 'members' a
nd members property has 2 elements
{
  type : 'members',
  members : [
   {'caption' : 'Cars', 'uniqueName' : 'category.[cars]'},
   {'caption' : 'Bikes', 'uniqueName' : 'category.[bikes]'}
  ]
}
*/

flexmonster.clearFilter('Category');

flexmonster.getFilter('Category');
/*
after clearFilter() call method getFilter() returns an empty Array
[]
*/

flexmonster.getFilterProperties('Category');
/*
after clearFilter() call method getFilterProperties() returns an Object with type 'n
one' and empty members array
{
  type : 'none',
  members : []
}
*/
```

Open the example on JSFiddle(http://jsfiddle.net/flexmonster/mq32shu2/)
.

See also

getFilter(/api/getfilter/)

getFilterProperties(/api/getfilterproperties/)

setFilter(/api/setfilter/)

setTopX(/api/settopx/)

setBottomX(/api/setbottomx/)

3.9. clearXMLACache

clearXMLACache(proxyURL:String, databaseId:String, callbackHandler:String, cubeId:String, measuresGroupId:String, username:String, password:String)

[starting from version: 2.2]

API call that requests Microsoft Analysis Services to clear the cache. Please visit official documentation from Microsoft for more details – https://msdn.microsoft.com/en-US/LIbrary/hh230974.aspx?f=255&MSPPError=-2147217396(https://msdn.microsoft.com/en-US/LIbrary/hh230974.aspx?f=255&MSPPError=-2147217396)

Parameters

- proxyUrl – the path to proxy URL to the Microsoft Analysis Services data source.
- databaseId – the ID of the database on the current connection.
- callbackHandler (optional) – JS function which will be called when the component completes the request. In case of success the following object will be returned: { complete: true, response: response from MSAS }. In case of failure the object will contain only one property: { complete: false }.
- cubeId (optional) – cube ID.
- measuresGroupId (optional) – alternatively, you can specify a path of a child object, such as a measure group, to clear the cache for just that object.
- username (optional) – the name of user account at server. This parameter is necessary to complete authentication process.
- password (optional) – the password to user account at server. This parameter is necessary to complete authentication process.

Example

```
function clearCache() {
    flexmonster.clearXMLACache(
     // replace with your proxyUrl
     "http://olap.flexmonster.com/olap/msmdpump.dll",
     // replace with DatabaseID
     "Adventure Works DW 2008",
     function (res) {
      // check response
      console.log(res);
     }
    );
  }
clearCache();
```

Open the example on JSFiddle(http://jsfiddle.net/flexmonster/m4m5xwrd/)
.

3.10. closeFieldsList

closeFieldsList()

[starting from version: 1.4]

Closes Fields List.

Example

```
flexmonster.closeFieldsList();
```

Open the example on JSFiddle(https://jsfiddle.net/flexmonster/g2exue63/)
.

See also

openFieldsList(/api/openfieldslist/)

3.11. collapseAllData

[starting from version: 1.4]

Collapses all nodes and drills up (starting from v2.1) all levels of all hierarchies in the slice on the grid and on charts. All expanded/drilled down nodes will be collapsed/drilled up on the grid and on charts.

Example

```
flexmonster.collapseAllData();
```

Try the example on JSFiddle(https://jsfiddle.net/flexmonster/eyz8evf6/)
.

See also

collapseData(/api/collapsedata/)

expandAllData(/api/expandalldata/)

expandData(/api/expanddata/)

3.12. collapseData

collapseData(hierarchyName:String)

[starting from version: 1.6]

Collapses all nodes of the specified hierarchy.

Example

```
flexmonster.collapseData('Country');
```

Try on JSFiddle(https://jsfiddle.net/flexmonster/r7t83L8L/)
.

See also

collapseAllData(/api/collapsealldata/)

expandAllData(/api/expandalldata/)

expandData(/api/expanddata/)

3.13. connectTo

connectTo(params:Object)

[starting from version: 1.4]

This method is used for connection to the new data source. It clears the current report and connects to the resource specified. Please note, starting from version 2.3 there is a new API call updateData(/api/updatedata/) to update data for the report without cleaning the report.

Possible data sources:

- OLAP cube via XMLA protocol (Microsoft Analysis Services, Mondrian, icCube)
- OLAP cube via our proxy (Microsoft Analysis Services, Mondrian)
- CSV (static file or data generated by server-side script)
- CSV file from the local file system
- JSON (starting from v2.2) – see addJSON()(/api/addjson/) for more details and samples

Parameters

params – the object which contains connection parameters. List of possible parameters:

- dataSourceType – type of data source. The component supports the following types: "microsoft analysis services", "mondrian", "iccube", "csv", "ocsv", "json".
- proxyUrl – the path to proxy URL to the OLAP data source, such as Microsoft Analysis Services, Mondrian, icCube (only for "microsoft analysis services", "mondrian", "iccube" data source types)
- dataSourceInfo – the service info of the OLAP data source (only for "microsoft analysis services", "mondrian", "iccube" data source types)
- catalog – the data source catalog name of the OLAP data source (only for "microsoft analysis services", "mondrian", "iccube" data source types)
- cube – given catalog's cube's name of the OLAP data source (only for "microsoft analysis services", "mondrian", "iccube" data source types)
- filename – the URL to CSV file or to server-side script which generates CSV data (only for "csv", "ocsv" and "json" data source type)
- browseForFile – this boolean parameter defines whether you want to load CSV file from the local file system (true) or not (false). It is *false* by default. (only for "csv", "ocsv" and "json" data source type)
- fieldSeparator – defines specific fields separator to split CSV row (only for "csv" data source type). There is no need to define it if CSV fields are separated by , or ;. This property is used if another char separates fields. For example, if you use TSV, where tab char is used to separate fields in row, fieldSeparator parameter should be defined explicitly.

Examples

1) This example(http://jsfiddle.net/flexmonster/ghvq9hhw/)
on JSFiddle demonstrates the connection to the following data sources: Microsoft Analysis Services, Mondrian, icCube, CSV, and JSON.

2) Connect to Microsoft Analysis Services:

```
flexmonster.connectTo({
  dataSourceType: 'microsoft analysis services',
  proxyUrl: 'http://olap.flexmonster.com/olap/msmdpump.dll',
  dataSourceInfo: 'Provider=MSOLAP; Data Source=extranet;',
  catalog: 'Adventure Works DW Standard Edition',
  cube: 'Adventure Works'
});
```

3) Connect to Mondrian:

```
flexmonster.connectTo({
  dataSourceType: 'mondrian',
  proxyUrl: 'http://olap.flexmonster.com:8080/mondrian/xmla',
  dataSourceInfo: 'MondrianFoodMart',
  catalog: 'FoodMart',
  cube: 'Sales'
});
```

4) Connect to icCube:

```
flexmonster.connectTo({
  dataSourceType: 'iccube',
  proxyUrl: 'http://olap.flexmonster.com:8282/icCube/xmla',
  dataSourceInfo: 'icCube-datasource',
  catalog: 'Sales',
  cube: 'Sales'
});
```

5) Connect to CSV data source:

```
flexmonster.connectTo({
  dataSourceType: 'csv',
  filename: 'data/csv/arabic.csv'
});
```

6) Connect to CSV file where colon char is used to separate fields in the row. You have to define fieldSeparator explicitly:

```
flexmonster.connectTo({
```

```
  dataSourceType: 'csv',
  filename: 'colon-data.csv',
  fieldSeparator: ':'
});
```

7) Open local CSV file:

```
flexmonster.connectTo({
  dataSourceType: 'csv',
  browseForFile: true
});
```

See also

updateData(/api/updatedata/)

open(/api/open/)

load(/api/load/)

save(/api/save/)

getReport(/api/getreport/)

setReport(/api/setreport/)

3.14. customizeCell

customizeCell(customizeCellFunction:Function)

[starting from version: 2.306]

This API call allows customizing of separate cells. For example, you can add links, custom styles or formatting.

Parameters

customizeCellFunction function or null in case you do not need to change anything. Data passed to the customizeCellFunction:

- html – String containing HTML.
- data – Cell Object(/api/cell-object/)
  which contains information about the cell.

customizeCellFunction should return the string with updated HTML.

Example

```
flexmonster.customizeCell(
 function (html, data)
```

```
 {
  // change html
  return html;
 }
);
```

This example(http://jsfiddle.net/flexmonster/q1gtwj48/)
illustrates how to add links to some cells. Click Clear Customizing button to remove customization and click Start Customizing to add it back.

3.15. dispose

dispose()

[starting from version: 2.3]

Prepares the pivot table instance to be deleted with the browser's garbage collection.

Example

```
flexmonster.dispose();
```

Open the example on JSFiddle(https://jsfiddle.net/flexmonster/xva6e0nk/)
.

3.16. embedPivotComponent

[removed]

embedPivotComponent method was deprecated in version 2.3. You should use
$("#pivotContainer").flexmonster()(/api/flexmonster/)
instead.

3.17. expandAllData

expandAllData(withAllChildren:Boolean)

[starting from version: 1.4]

Expands all nodes and drills down all levels of all hierarchies in the slice on the grid and on charts. All collapsed/drilled up nodes will be expanded/drilled down on the grid and on charts.

Parameters

- withAllChildren (starting from v2.1) (optional) – Boolean. It is used to expand nodes but not drill down the levels of hierarchies in the slice. Please set it to false if you want just expand nodes. Default value is true, which means that the drill down will be performed as well.

Examples

Expands all nodes and drills down all hierarchies in the slice.

```
flexmonster.expandAllData();
```

Try the example on JSFiddle(https://jsfiddle.net/flexmonster/eyz8evf6/)
.

Expands all nodes but does not drill down the hierarchies in the slice.

```
flexmonster.expandAllData(false);
```

See also

expandData(/api/expanddata/)

collapseAllData(/api/collapsealldata/)

collapseData(/api/collapsedata/)

3.18. expandData

expandData(hierarchyName:String)

[starting from version: 1.6]

Expands all nodes of the specified hierarchy.

Example

```
flexmonster.expandData('Country');
```

Try on JSFiddle(https://jsfiddle.net/flexmonster/r7t83L8L/)
.

See also

expandAllData(/api/expandalldata/)

collapseAllData(/api/collapsealldata/)

collapseData(/api/collapsedata/)

3.19. exportTo

exportTo(type:String, params:Object, callbackHandler:Function|String)

[starting from version: 1.4]

Exports grid or chart to CSV, HTML, PDF, Image or Excel format. The file can be saved to the local file system or to your server (you need to have a script on the server side).

Parameters

- type – type of export. There are such types available: "csv", "html", "pdf", "image" and "excel".
- params (optional) – export parameters. Object params can contain the following properties:
    - filename – default name of the resulting file.
    - destinationType – defines where the component's content will be exported. Destination type can be the following:
        - "file" – the component's content will be exported to the file to the local computer.
        - "server" – the component's content will be exported to the server (a server-side script is required).
    - excelSheetName (starting from v2.2) (optional) – String. To configure the sheet name when exporting to Excel file.
    - footer (starting from v2.211) (optional) – String. HTML and PDF only. Footer is set in HTML format (tags, inline styles, img with base64 src). For PDF it is rendered in the browser and added as an image to the exported file. The following tokens can be used for PDF export: ##CURRENT-DATE##, ##PAGE-NUMBER##. They will be replaced by appropriate data. ##CURRENT-DATE## is also available for HTML export.
    - header (starting from v2.211) (optional) – String. HTML and PDF only. Header is set in HTML format (tags, inline styles, img with base64 src). For PDF it is rendered in the browser and added as an image to the exported file. The following tokens can be used for PDF export: ##CURRENT-DATE##, ##PAGE-NUMBER##. They will be replaced by appropriate data. ##CURRENT-DATE## is also available for HTML export.
    - pageOrientation (optional) – defines the page orientation for a PDF file. Page orientation can be the following:
        - "portrait" (by default) – defines portrait page orientation for a PDF file.
        - "landscape" – defines landscape page orientation for a PDF file.
    - showFilters (starting from v2.1) (optional) – Boolean. Excel only. Indicates whether the filters info will be shown (true) in exported Excel file or not (false). Default value is false.
    - url – to save the file to the server you should provide the component with a path to the server-side script which can save this file.
    - useOlapFormattingInExcel (starting from v2.2) (optional) – Boolean. To configure how to export grid cells in Excel file if formatting is taken from OLAP cube – as a formatted string (true) or as numbers without formatting (false). Previously it was not configurable and the cells were exported as formatted strings.
- callbackHandler (optional) – Callback handler is called with the following object: {data: result}.

Examples

1) This example(https://jsfiddle.net/flexmonster/45e9k4c1/)
on JSFiddle demonstrates all types of export: CSV, HTML, PDF, Image and Excel.

2) Export to CSV, save as a local file and add a callback handler:

```
flexmonster.exportTo('csv', {filename : 'flexmonster.csv'},
    function(result) {console.log(result.data)}
);
```

3) Export to HTML and save as local file:

```
var params = {
  filename : 'flexmonster.html'
};
flexmonster.exportTo('html', params);
```

4) Export to PDF file, change page orientation to landscape and save file to the server:

```
var params = {
  filename : 'flexmonster.pdf',
  pageOrientation : 'landscape',
  destinationType : 'server',
  url : 'your server'
};
flexmonster.exportTo('pdf', params);
```

5) Export to Excel and save as local file:

```
flexmonster.exportTo('excel');
```

See also

print(/api/print/)

3.20. fullScreen

fullScreen()

[starting from version: 1.4]

Switches the component to full-screen mode.

Example

```
flexmonster.fullScreen();
```

Check out on JSFiddle(https://jsfiddle.net/flexmonster/hwx229ep/)
.

See also

showCharts(/api/showcharts/)

showGrid(/api/showgrid/)

3.21. getAllConditions

getAllConditions():Array

[starting from version: 1.6]

Returns a list of conditional formatting rules of the report. You may need this API call to edit existing conditional formatting rules.

Each element in array is Conditional Format Object(/api/conditional-format-object/)
.

Example

To get all the conditional formatting rules of the report use getAllConditions(), as follows:

```
var conditions = flexmonster.getAllConditions();
```

Try the example on JSFiddle(http://jsfiddle.net/flexmonster/pvu6m8fs/)
.

See also

addCondition(/api/addcondition/)

getCondition(/api/getcondition/)

removeCondition(/api/removecondition/)

removeAllConditions(/api/removeallconditions/)

3.22. getAllHierarchies

getAllHierarchies():Array

[starting from version: 1.4]

Returns a list of all available hierarchies.

Returns

Array of objects. Each object in array contains two parameters: caption and uniqueName.

If data load is in progress an empty array will be returned.

Example

```
flexmonster.getAllHierarchies();
```

```
/* method returns array of objects
[
{caption: "Business Type", uniqueName: "Business Type"},
{caption: "Category", uniqueName: "Category"},
{caption: "Country", uniqueName: "Country"}
]
*/
```

Check out on JSFiddle(https://jsfiddle.net/flexmonster/qqk9zcf1/)
.

See also

getAllMeasures(/api/getallmeasures/)

getColumns(/api/getcolumns/)

getRows(/api/getrows/)

getPages(/api/getpages/)

getMeasures(/api/getmeasures/)

3.23. getAllMeasures

getAllMeasures():Array

[starting from version: 1.4]

Returns a list of all available measures.

Returns

Array of objects. Each object in array contains the following parameters:

- name
- uniqueName
- aggregation
- availableAggregations – Array of available aggregations.
- availableAggregationsCaptions – Array of available aggregations captions.
- formula – For calculated measures.
- caption
- grandTotalCaption
- format

If data load is in progress an empty array will be returned.

Example

```
flexmonster.getAllMeasures();

/* method returns array of objects, where the 2nd measure is calculated
[
  {aggregation: "sum",
   availableAggregations: ["sum", "count", "distinctcount", "average", "product", "m
in", "max", "percent", "percentofcolumn"],
   availableAggregationsCaptions: ["Sum", "Count", "Distinct Count", "Average", "Pro
duct", "Min", "Max", "Percent", "Percent of Column"],
   caption: "Sum of Sales",
   format: "currency",
   grandTotalCaption: "Total Sum of Sales",
   name: "Sales",
   uniqueName: "Sales"},
  {aggregation: "none",
   availableAggregations: [ ],
   availableAggregationsCaptions: [ ],
   caption: "Test",
   format: "",
   formula: "(SUM("Price") / count("Price")) * 100",
   grandTotalCaption: "Total Test",
   name: "Test",
   uniqueName: "Test"}
]
*/
```

Try on JSFiddle(https://jsfiddle.net/flexmonster/79keh0ee/)
.

See also

getMeasures(/api/getmeasures/)

getAllHierarchies(/api/getallhierarchies/)

getColumns(/api/getcolumns/)

getRows(/api/getrows/)

getPages(/api/getpages/)

3.24. getCell

getCell(rowIdx:Number, colIdx:Number):Cell Object(/api/cell-object/)

[starting from version: 1.4]

Returns information about cell by row and column indexes.

Parameters

- rowIdx – index of the row.
- colIdx – index of the column.

Returns

Cell Object(/api/cell-object/)
which contains information about the requested cell.

Example

```
flexmonster.getCell(1,1);
```

Open the example on JSFiddle(https://jsfiddle.net/flexmonster/97fvkauL/)
.

See also

getSelectedCell(/api/getselectedcell/)

3.25. getColumns

getColumns():Array

[starting from version: 1.4]

Returns a list of hierarchies selected in the report slice for columns.

Returns

Array of objects. Each object in array contains the following parameters: caption, uniqueName, and sort.

If data load is in progress an empty array is returned.

Example

```
flexmonster.getColumns();

/* method returns array of objects
[
{caption: "Business Type", uniqueName: "Business Type", sort: "desc"},
{caption: "Category", uniqueName: "Category", sort: "asc"}
]
*/
```

Check out on JSFiddle(https://jsfiddle.net/flexmonster/6mn247eh/)
.

See also

getAllHierarchies(/api/getallhierarchies/)

getRows(/api/getrows/)

getPages(/api/getpages/)

getAllMeasures(/api/getallmeasures/)

getMeasures(/api/getmeasures/)

3.26. getColumnWidth

[removed]

getColumnWidth method was removed in version 2.3.

3.27. getCondition

getCondition(id:String):Conditional Format Object(/api/conditional-format-object/)

[starting from version: 1.6]

Returns a conditional formatting rule by id. You may need this API call to edit existing conditional formatting rule.

Parameters

- id – id of the condition.

Returns

Conditional Format Object(/api/conditional-format-object/)
that describes the conditional formatting rule.

Example

To get the conditional formatting rule by id use getCondition(), as follows:

```
var id = "9";
var condition = flexmonster.getCondition(id);
```

Open the example on JSFiddle(http://jsfiddle.net/flexmonster/pvu6m8fs/)
.

See also

addCondition(/api/addcondition/)

getAllConditions(/api/getallconditions/)

removeCondition(/api/removecondition/)

removeAllConditions(/api/removeallconditions/)

3.28. getData

getData(options:Object, callbackHandler:Function, updateHandler:Function)

[starting from version: 2.3]

This method is used for integration with 3rd party charting libraries. It asynchronously passes the data to callbackHandler and updateHandler. You can retrieve the data that pivot instance is showing or define the slice with the data you would like to get.

Parameters

- options – Object. This object has the following parameters:
  - slice (optional) – Object. Contains information about the slice. If not defined, the API call will return data displayed in the pivot table.
- callbackHandler – Function. Gets one input parameter – rawData. Tracks when the data is ready.
- updateHandler (optional) – Function. Gets one input parameter – rawData. Tracks if data in pivot was filtered/sorted/etc.

Response

rawData is the object asynchronously passed to callbackHandler and updateHandler. It has the following structure:

- data – Array. Array of objects that represents all the data from the dataset. Each object can have c0 - cN, r0 - rN and v0 - vN parameters, where c0 - cN are for column members, r0 - rN are for row members and v0 - vN are for values. Please note that if a cell has no value, NaN will be put in corresponding v0 - vN.
- meta – Object. Meta data about the returned data:
  - caption – String. Chart's title.
  - cAmount – Number. The number of fields in columns in the slice.
  - c0Name – String. The caption of the first field in columns in the slice. In meta object will be as many c0Name, c1Name, c2Name, etc as cAmount.
  - formats – Array. Formats of measures from the slice. It has as many format objects as vAmount.
  - rAmount – Number. The number of fields in rows in the slice.
  - r0Name – String. The caption of the first field in rows in the slice. In meta object will be as many r0Name, r1Name, r2Name, etc as rAmount.
  - vAmount – Number. The number of measures in the slice.
  - v0Name – String. The caption of the first measure in the slice. In meta object will be as many v0Name, v1Name, v2Name, etc as vAmount.

Example

Example of rawData response. We have the following pivot table:

Category

Color	Accessories	Components	Total Sum of Price
blue		553 584	553 584
green	28 008	207 128	235 136
Grand Total	28 008	760 712	788 720

The result of the API call

```
flexmonster.getData({}, function(data) {console.log(data)})
```

will be the following:

```
{
 data:[
  {
   v0:788720
  },
  {
   r0:"blue",
   v0:553584
  },
  {
   r0:"green",
   v0:235136
  },
  {
   c0:"Accessories",
   v0:28008
  },
  {
   c0:"Components",
   v0:760712
  },
  {
   c0:"Accessories",
   r0:"blue",
   v0:NaN
  },
  {
   c0:"Components",
   r0:"blue",
   v0:553584
  },
  {
   c0:"Accessories",
   r0:"green",
   r0:28008
  },
  {
   c0:"Components",
   r0:"green",
   v0:207128
  }
 ],
```

```
 meta: {
  caption:"",
  cAmount:1,
  c0Name:"Category",
  formats: [{
   name:"",
   currencySymbol:"",
   currencySymbolAlign:"left",
   decimalPlaces:-1,
   decimalSeparator:".",
   divideByZeroValue:"Infinity",
   infinityValue:"Infinity",
   isPercent:false,
   maxDecimalPlaces:-1,
   maxSymbols:20,
   nullValue:"",
   textAlign:"right",
   thousandsSeparator:" "
  }],
  rAmount:1,
  r0Name:"Color",
  vAmount:1,
  v0Name:"Sum of Price"
 }
}
```

data array in rawData object contains all numbers shown in the pivot table including grand totals, totals and subtotals.

Each object with grand totals contains values (v0 - vN) only. In our example this is:

```
  {
   v0:788720
  }
```

Each object with totals contains either values (v0 - vN) and columns (c0 - cN) or values and rows (r0 - rN). In our example this is:

```
  {
   r0:"blue",
   v0:553584
  },
  {
   r0:"green",
   v0:235136
  },
  {
   c0:"Accessories",
   v0:28008
```

```
  },
  {
   c0:"Components",
   v0:760712
  }
```

Depending on for which visualization tool you are requesting data from the pivot table, you may need data with grand totals, totals and subtotals or without them. For some chart types, only totals are necessary.

Check out the example on JSFiddle(http://jsfiddle.net/flexmonster/dhos9py3/)
.

3.29. getFilter

getFilter(hierarchyName:String):Array

[starting from version: 1.4]

Returns the filtered members for the specified hierarchy. If data load is in progress or filter is not applied an empty Array will be returned.

Parameters

- hierarchyName – the name of the hierarchy.

Returns

Array of filtered objects. Each object in array contains the following parameters: caption, uniqueName, and hierarchyName. It is empty if filter is not applied to the hierarchy.

Example

```
var filter = [
    "category.[cars]",
    "category.[bikes]"
];
flexmonster.setFilter('Category', filter);
flexmonster.getFilter('Category');

/*
method getFilter() returns the following Array
[
  {'caption' : 'Cars', 'hierarchyName' : 'Category', 'uniqueName' : 'category.[cars]
'},
  {'caption' : 'Bikes', 'hierarchyName' : 'Category', 'uniqueName' : 'category.[bike
s]'}
]
*/

flexmonster.clearFilter('Category');
flexmonster.getFilter('Category');
```

```
/*
after clearFilter() call method getFilter() returns an empty Array
[]
*/
```

Try the example on JSFiddle(http://jsfiddle.net/flexmonster/mq32shu2/)
.

See also

clearFilter(/api/clearfilter/)

getFilterProperties(/api/getfilterproperties/)

setFilter(/api/setfilter/)

setTopX(/api/settopx/)

setBottomX(/api/setbottomx/)

3.30. getFilterProperties

getFilterProperties(hierarchyName:String):Object

[starting from version: 1.6]

Returns the filter properties set for the specified hierarchy. If data load is in progress or filter is not applied an empty Object will be returned.

Parameters

- hierarchyName – the name of the hierarchy.

Returns

Object that describes the filter set for the hierarchy. The object always has type and members properties. It may have quantity and measure properties if filter on values is defined. It has all four properties if both filters, on hierarchy's members and on values, are defined:

- type – Represents the filter type applied to the hierarchy. It can be:
  - 'none' – filter is not applied to the hierarchy,
  - 'members' – the filter on hierarchy's members is applied,
  - 'top' – the filter Top X is applied on values,
  - 'bottom' – the filter Bottom X is applied on values.
- members – Array of objects represents the filter on hierarchy's members. Each object in array contains the following parameters: caption, uniqueName, and hierarchyName.
- quantity – Represents the filter on values. Number of elements to choose for the Top X filter if type is 'top' or for the Bottom X filter if type is 'bottom'.
- measure – Represents the filter on values. The name of the measure on which Top X or Bottom X filter will

be applied.

Example

```
var filter = [
    "category.[cars]",
    "category.[bikes]"
];
flexmonster.setFilter('Category', filter);

flexmonster.getFilterProperties('Category');
/*
method getFilterProperties() returns the following object, where type is 'members' a
nd members property has 2 elements
{
  type : 'members',
  members : [
   {'caption' : 'Cars', 'uniqueName' : 'category.[cars]'},
   {'caption' : 'Bikes', 'uniqueName' : 'category.[bikes]'}
  ]
}
*/

flexmonster.clearFilter('Category');

flexmonster.getFilterProperties('Category');
/*
after clearFilter() call method getFilterProperties() returns an Object with type 'n
one' and empty members array
{
  type : 'none',
  members : []
}
*/
```

Try the example on JSFiddle(http://jsfiddle.net/flexmonster/w3fq620e/)
.

See also

clearFilter(/api/clearfilter/)

getFilter(/api/getfilter/)

setFilter(/api/setfilter/)

setTopX(/api/settopx/)

setBottomX(/api/setbottomx/)

3.31. getFormat

getFormat(measureName:String):Format Object(/api/format-object/)

[starting from version: 1.4]

Returns Format Object(/api/format-object/)
of a default number format or the number format for the specified measure. The number format can be defined via report or via setFormat()(/api/setformat/)
API method. Each measure has only one format but a format can be applied to more than one measure.

Parameters

- measureName (optional) – the unique name of the measure. If measure's unique name is not specified or it is not found, the default number format will be returned.

Examples

1) How to get a precision:

```
var format = flexmonster.getFormat();
alert("Precision: " + format.decimalPlaces);
```

Try on JSFiddle(http://jsfiddle.net/flexmonster/8a8hcvaa/)
.

2) How to change a currency symbol:

```
var format = flexmonster.getFormat("Price");
format.currencySymbol = "$";
//format.currencySymbol = "&#163;" // pound sterling
//format.currencySymbol = "&#8364;" // euro
//format.currencySymbol = "&#165;" // yen
flexmonster.setFormat(format, "Price");
flexmonster.refresh();
```

Open the example on JSFiddle(http://jsfiddle.net/flexmonster/kc8fq69y/)
.

See also

setFormat(/api/setformat/)

Format Object(/api/format-object/)

3.32. getLabels

[removed]

getLabels method was removed in version 2.3. Instead you can use report.localization from getReport()(/api/getReport/)
.

3.33. getMeasures

getMeasures():Array

[starting from version: 1.4]

Returns a list of the selected measures in the report.

Returns

Array of objects. Each object in array contains the following parameters:

- name
- uniqueName
- aggregation
- availableAggregations – Array of available aggregations.
- availableAggregationsCaptions – Array of available aggregations captions.
- formula – For calculated measures.
- caption
- grandTotalCaption
- format

If data load is in progress an empty array will be returned.

Example

```
flexmonster.getMeasures();

/* method returns array of objects
[
  {aggregation: "sum",
   availableAggregations: ["sum", "average", "percent"],
   availableAggregationsCaptions: ["Sum", "Count", "Percent"],
   caption: "Sum of Sales",
   format: "currency",
   grandTotalCaption: "Total Sum of Sales",
   name: "Sales",
   uniqueName: "Sales"},
  {aggregation: "sum",
   availableAggregations: ["sum", "count", "distinctcount", "average", "product", "m
in", "max", "percent", "percentofcolumn", "percentofrow", "index"],
   availableAggregationsCaptions: ["Sum", "Count", "Distinct Count", "Average", "Pro
duct", "Min", "Max", "% of Grand Total", "% of Column", "% of Row", "Index"],
   caption: "Sum of Orders",
   format: "",
   grandTotalCaption: "Total Sum of Orders",
   name: "Orders",
   uniqueName: "Orders"}
```

```
]
*/
```

Try on JSFiddle(https://jsfiddle.net/flexmonster/79keh0ee/)
.

See also

getAllMeasures(/api/getallmeasures/)

getAllHierarchies(/api/getallhierarchies/)

getColumns(/api/getcolumns/)

getRows(/api/getrows/)

getPages(/api/getpages/)

3.34. getMembers

getMembers(hierarchyName:String, memberName:String, callbackHandler:String):Array

[starting from version: 1.6]

Returns a list of members for the specified hierarchy.

CSV data source. If the hierarchy has more than one level the method returns a tree where each member has a list of its children. It is enough to specify hierarchyName parameter only.

OLAP data source. If the hierarchy has more than one level the method returns only the members for the first level. To get children of the member, you should call getMembers() with hierarchyName, memberName and callbackHandler parameters.

Parameters

- hierarchyName – the name of the hierarchy.
- memberName (optional) – hierarchy's member name. This parameter can be an empty string, only the members for the first level of the hierarchy will be returned.
- callbackHandler (optional) – Callback handler becomes useful when you works with OLAP data sources and you are not really sure whether you have got all list of members taken in completely or not. OLAP data source is not loaded for the whole data from the moment of call but will be loaded on demand. Callback handler will be called by the component to pass the result asynchronously when the list of members is loaded. If you have already loaded the specified hierarchy members' list into the component, callbackHandler will return instantly the result of the object.

Returns

Array of objects with member's caption, uniqueName, hierarchyName, children, isLeaf and parentMember properties.

If data load is in progress and callbackHandler is not set an empty array will be returned.

Examples

1) CSV data source, where Category hierarchy has only one level:

```
flexmonster.getMembers("Category");

/* method returns array of objects
[
  {caption: "Accessories", hierarchyName: "Category", uniqueName: "category.[accesso
ries]", children: [], isLeaf:true, parentMember:"(All)"},
  {caption: "Cars", hierarchyName: "Category", uniqueName: "category.[cars]", childr
en: [], isLeaf:true, parentMember:"(All)"},
  {caption: "Clothing", hierarchyName: "Category", uniqueName: "category.[clothing]"
, children: [], isLeaf:true, parentMember:"(All)"}
]
*/
```

Check out on JSFiddle(http://jsfiddle.net/flexmonster/tsq6yL0h/)
.

2) CSV data source, where Date hierarchy has more than one level:

```
flexmonster.getMembers('Date');

/* method returns array of objects
[
  {
   caption: "2003",
   hierarchyName: "Date",
   uniqueName: "date.[2003]",
   children: [
    {
     caption: "Quarter 1",
     hierarchyName: "Date",
     uniqueName: "date.quarter.[2003/quarter 1]",
     children: [
     {
       caption: "January",
       hierarchyName: "Date",
       uniqueName: "date.quarter.[2003/quarter 1/january]",
       children: [ ],
       isLeaf:true,
       parentMember:"(All)
      }
    },
    {
     caption: "Quarter 2",
     hierarchyName: "Date",
     uniqueName: "date.quarter.[2003/quarter 2]",
     children: [
```

```
      {
       caption: "April",
       hierarchyName: "[Date].[Date]",
       uniqueName: "date.quarter.[2003/quarter 2/april]",
       children: [ ],
       isLeaf:true,
       parentMember:"(All)
      }
     }
    ],
    isLeaf:true,
    parentMember:"(All)
   },
   {
    caption: "2004",
    hierarchyName: "Date",
    uniqueName: "date.[2004]",
    children: [
     {
      caption: "Quarter 3",
      hierarchyName: "Date",
      uniqueName: "date.quarter.[2004/quarter 3]",
      children: [
       {
        caption: "July",
        hierarchyName: "Date",
        uniqueName: "date.quarter.[2004/quarter 3/july]",
        children: [ ],
        isLeaf:true,
        parentMember:"(All)
       },
       {
        caption: "August",
        hierarchyName: "Date",
        uniqueName: "date.quarter.[2004/quarter 3/august]",
        children: [ ],
        isLeaf:true,
        parentMember:"(All)
       }
     }
    ],
    isLeaf:true,
    parentMember:"(All)
   }
]
*/
```

3) OLAP data source, where [Product].[Product Categories] hierarchy has more than one level:

```
flexmonster.getMembers('[Product].[Product Categories]', '', 'onGetMembers');

function onGetMembers(members) {
```

```
  for (var i = 0; i < members.length; i++) {
   console.log(members[i]);
  }
}

/* the output will be the following:
(Object)#0
  caption = "Accessories"
  hierarchyName = "[Product].[Product Categories]"
  isLeaf = false
  parentMember = "[Product].[Product Categories].[All]"
  uniqueName = "[Product].[Product Categories].[Category].&[4]"
(Object)#0
  caption = "Bikes"
  hierarchyName = "[Product].[Product Categories]"
  isLeaf = false
  parentMember = "[Product].[Product Categories].[All]"
  uniqueName = "[Product].[Product Categories].[Category].&[1]"
(Object)#0
  caption = "Clothing"
  hierarchyName = "[Product].[Product Categories]"
  isLeaf = false
  parentMember = "[Product].[Product Categories].[All]"
  uniqueName = "[Product].[Product Categories].[Category].&[3]"
(Object)#0
  caption = "Components"
  hierarchyName = "[Product].[Product Categories]"
  isLeaf = false
  parentMember = "[Product].[Product Categories].[All]"
  uniqueName = "[Product].[Product Categories].[Category].&[2]"

//and now you can ask for '[Product].[Product Categories].[Category].&[4]' members:

flexmonster.getMembers('[Product].[Product Categories]', '[Product].[Product Categor
ies].[Category].&[4]', 'onGetMembers');
```

3.35. getOptions

getOptions():Options Object(/api/options-object/)

[starting from version: 1.6]

Returns Options Object(/api/options-object/)
with component's options.

Example

How to turn off totals:

```
var options = flexmonster.getOptions();
options.grid.showTotals = false;
flexmonster.setOptions(options);
```

```
flexmonster.refresh();
```

Try on JSFiddle(https://jsfiddle.net/flexmonster/afr1g5Lf/)
.

See also

Options Object(/api/options-object/)

setOptions(/api/setoptions/)

3.36. getPages

getPages():Array

[starting from version: 1.4]

Returns a list of hierarchies selected in the report slice for pages (“Report Filter”).

Returns

Array of objects. Each object in array contains the following parameters: caption, uniqueName, and sort.

If data load is in progress an empty array will be returned.

Example

```
flexmonster.getPages();

/* method returns array of objects
[
{caption: "Business Type", uniqueName: "Business Type", sort: "desc"},
{caption: "Category", uniqueName: "Category", sort: "asc"}
]
*/
```

Open the example on JSFiddle(https://jsfiddle.net/flexmonster/6mn247eh/)
.

See also

getAllHierarchies(/api/getallhierarchies/)

getColumns(/api/getcolumns/)

getRows(/api/getrows/)

getAllMeasures(/api/getallmeasures/)

getMeasures(/api/getmeasures/)

3.37. getReport

getReport(options:Object):ReportObject(/api/report-object/)

[starting from version: 1.4]

Returns Report Object(/api/report-object/)
which describes the current report. Use this object to save or edit report at runtime.

Parameters

options (optional) – Object. Can contain the following properties:

- withDefaults (optional) – Boolean. Indicates whether the default values for options will be included in the report (true) of not (false). The default value is false.
- withGlobals (optional) – Boolean. Indicates whether the options defined in global object will be included in the report (true) of not (false). The default value is false.

Example

1) Get report:

```
<div id="pivotContainer">The component will appear here</div>
<script src="flexmonster/lib/jquery.min.js"></script>
<script src="flexmonster/flexmonster.js"></script>
<script>
var pivot = $("#pivotContainer").flexmonster({
 licenseKey: "XXXX-XXXX-XXXX-XXXX-XXXX",
 report: {
  dataSource: {
   filename: "http://cdn.flexmonster.com/2.3/data/data.csv"
  }
 }
});
</script>

<button onclick="getReport()">Get Report</button>
<script>
 function getReport() {
  console.log(pivot.getReport());
 }
</script>
```

Open the example on JSFiddle(http://jsfiddle.net/flexmonster/eu3veppc/)
.

2) Swap two reports:

```
<div id="firstPivotContainer">The component will appear here</div>
<div id="secondPivotContainer">The component will appear here</div>
<script src="flexmonster/lib/jquery.min.js"></script>
<script src="flexmonster/flexmonster.js"></script>
<script>
var pivot1 = $("#firstPivotContainer").flexmonster({
 toolbar: true,
 report: {
  dataSource: {
   filename: "data1.csv"
  }
 },
 licenseKey: "XXXX-XXXX-XXXX-XXXX-XXXX"
});
var pivot2 = $("#secondPivotContainer").flexmonster({
 toolbar: true,
 report: {
  dataSource: {
   filename: "data2.csv"
  }
 },
 licenseKey: "XXXX-XXXX-XXXX-XXXX-XXXX"
});
</script>

<button onclick="javascript: swapReports()">Swap Reports</button>
<script>
  function swapReports() {
    var report1 = pivot1.getReport();
    var report2 = pivot2.getReport();

    pivot1.setReport(report2);
    pivot2.setReport(report1);
  }
</script>
```

Try on JSFiddle(http://jsfiddle.net/flexmonster/1dLjhu0s/)
.

3) Get report with defaults or globals: JSFiddle(http://jsfiddle.net/flexmonster/ft6revLs/)
.

See also

setReport(/api/setreport/)

Report Object(/api/report-object/)

open(/api/open/)

load(/api/load/)

save(/api/save/)

3.38. getRowHeight

[removed]

getRowHeight method was removed in 2.3 version.

3.39. getRows

getRows():Array

[starting from version: 1.4]

Returns a list of hierarchies selected in the report slice for rows.

Returns

Array of objects. Each object in array contains the following parameters: caption, uniqueName, and sort.

If data load is in progress an empty array will be returned.

Example

```
flexmonster.getRows();

/* method returns array of objects
[
{caption: "Business Type", uniqueName: "Business Type", sort: "desc"},
{caption: "Category", uniqueName: "Category", sort: "asc"}
]
*/
```

Check out on JSFiddle(https://jsfiddle.net/flexmonster/6mn247eh/)
.

See also

getAllHierarchies(/api/getallhierarchies/)

getColumns(/api/getcolumns/)

getPages(/api/getpages/)

getAllMeasures(/api/getallmeasures/)

getMeasures(/api/getmeasures/)

3.40. getSelectedCell

getSelectedCell():Cell Object(/api/cell-object/)

[starting from version: 1.4]

Returns information about selected cell.

Returns

Cell Object(/api/cell-object/)
which contains information about selected cell.

Example

```
flexmonster.getSelectedCell();
```

Open the example on JSFiddle(https://jsfiddle.net/flexmonster/97fvkauL/)
.

See also

getCell(/api/getcell/)

removeSelection(/api/removeselection/)

3.41. getSort

getSort(hierarchyName:String):String

[starting from version: 1.4]

Returns the sort type which is applied to the hierarchy.

Parameters

- hierarchyName – The name of the hierarchy.

Returns

One of the following sort types: 'asc', 'desc', or 'unsorted'.

Example

```
flexmonster.getSort("Category");
```

Open the example on JSFiddle(http://jsfiddle.net/flexmonster/9h15aeye/)
.

See also

setSort(/api/setsort/)

sortValues(/api/sortvalues/)

3.42. getValue

[removed]

getValue method was removed in version 2.3. Instead you can use getCell()(/api/getcell/)

3.43. getXMLACatalogs

getXMLACatalogs(proxyURL:String, dataSourceInfo:String, callbackHandler:String, username:String, password:String)

[starting from version: 1.5]

One of four API calls of OLAP/XMLA connectivity diagnostics.

Obtains a list of all available catalogs on a given data source by proxyURL and dataSourceInfo. Returns an Array of catalog names.

Parameters

- proxyUrl – the path to proxy URL to the OLAP data source, such as Microsoft Analysis Services, Mondrian, icCube.
- dataSourceInfo – the service info of the OLAP data source.
- callbackHandler – JS function which will be called when the component obtains a list of all available catalogs.
- username (optional) – the name of user account at server. This parameter is necessary to complete authentication process.
- password (optional) – the password to user account at server. This parameter is necessary to complete authentication process.

Example

OLAP/XMLA connectivity step by step. First, getXMLADataSources() obtains a list of all data sources by given URL (for example: 'http://olap.flexmonster.com/olap/msmdpump.dll') and getXMLAProviderName() returns dataSourceType. Second, getXMLACatalogs() gets all available catalogs for the first data source from the list of available data sources. Then getXMLACubes() obtains a list of all cubes for the first catalog of the list of available catalogs. Finally, connectTo()uses all the information about data source to connect to it:

```
var proxyUrl = 'http://olap.flexmonster.com/olap/msmdpump.dll';
var dataSourceInfo = '';
var dataSourceType = '';
var catalog = '';
var cube = '';

function diagnostic() {
  getDataSources();
}

function getDataSources() {
  flexmonster.getXMLADataSources(proxyUrl, 'onDataSourceLoaded');
}

function onDataSourceLoaded(result) {
  dataSourceInfo = result[0];
  dataSourceType = flexmonster.getXMLAProviderName(proxyUrl);
  flexmonster.getXMLACatalogs(proxyUrl, dataSourceInfo, 'onCatalogsLoaded');
}

function onCatalogsLoaded(result) {
  catalog = result[0];
  flexmonster.getXMLACubes(proxyUrl, dataSourceInfo, catalog, 'onCubesLoaded');
}

function onCubesLoaded(result) {
  cube = result[0];
  flexmonster.connectTo({
   dataSourceType: dataSourceType,
   proxyUrl: proxyUrl,
   dataSourceInfo: dataSourceInfo,
   catalog: catalog,
   cube: cube
  });
}

diagnostic();
```

Open the example on JSFiddle(http://jsfiddle.net/flexmonster/hp6t9w8y/)
.

See also

getXMLADataSources(/api/getxmladatasources/)

getXMLAProviderName(/api/getxmlaprovidername/)

getXMLACubes(/api/getxmlacubes/)

3.44. getXMLACubes

getXMLACubes(proxyURL:String, dataSourceInfo:String, catalog:String, callbackHandler:String, username:String, password:String)

[starting from version: 1.5]

One of four API calls of OLAP/XMLA connectivity diagnostics.

Obtains a list of all available cubes on a given data source by proxyURL, dataSourceInfo and catalog. Returns an Array of cube names.

Parameters

- proxyUrl – the path to proxy URL to the OLAP data source, such as Microsoft Analysis Services, Mondrian, icCube.
- dataSourceInfo – the service info of the OLAP data source.
- catalog – the data source catalog name of the OLAP data source.
- callbackHandler – JS function which will be called when the component obtains a list of all available cubes.
- username (optional) – the name of user account at server. This parameter is necessary to complete authentication process.
- password (optional) – the password to user account at server. This parameter is necessary to complete authentication process.

Example

OLAP/XMLA connectivity step by step. First, getXMLADataSources() obtains a list of all data sources by given URL (for example: 'http://olap.flexmonster.com/olap/msmdpump.dll') and getXMLAProviderName() returns dataSourceType. Second, getXMLACatalogs() gets all available catalogs for the first data source from the list of available data sources. Then getXMLACubes() obtains a list of all cubes for the first catalog of the list of available catalogs. Finally, connectTo()uses all the information about data source to connect to it:

```
var proxyUrl = 'http://olap.flexmonster.com/olap/msmdpump.dll';
var dataSourceInfo = '';
var dataSourceType = '';
var catalog = '';
var cube = '';

function diagnostic() {
  getDataSources();
}

function getDataSources() {
  flexmonster.getXMLADataSources(proxyUrl, 'onDataSourceLoaded');
}

function onDataSourceLoaded(result) {
  dataSourceInfo = result[0];
  dataSourceType = flexmonster.getXMLAProviderName(proxyUrl);
  flexmonster.getXMLACatalogs(proxyUrl, dataSourceInfo, 'onCatalogsLoaded');
}

function onCatalogsLoaded(result) {
  catalog = result[0];
  flexmonster.getXMLACubes(proxyUrl, dataSourceInfo, catalog, 'onCubesLoaded');
}
```

```
function onCubesLoaded(result) {
  cube = result[0];
  flexmonster.connectTo({
   dataSourceType: dataSourceType,
   proxyUrl: proxyUrl,
   dataSourceInfo: dataSourceInfo,
   catalog: catalog,
   cube: cube
  });
}

diagnostic();
```

Check out on JSFiddle(http://jsfiddle.net/flexmonster/hp6t9w8y/)
.

See also

getXMLADataSources(/api/getxmladatasources/)

getXMLAProviderName(/api/getxmlaprovidername/)

getXMLACatalogs(/api/getxmlacatalogs/)

3.45. getXMLADataSources

getXMLADataSources(proxyURL:String, callbackHandler:String, username:String, password:String)

[starting from version: 1.5]

One of four API calls of OLAP/XMLA connectivity diagnostics.

Obtains a list of all data sources by given URL for XMLA connect (proxyURL). Returns an Array of data sources.

Parameters

- proxyUrl – the path to proxy URL to the OLAP data source, such as Microsoft Analysis Services, Mondrian, icCube.
- callbackHandler – JS function which will be called when the component obtains a list of all data sources.
- username (optional) – the name of the user account at the server. This parameter is necessary to complete the authentication process.
- password (optional) – the password to the user account at the server. This parameter is necessary to complete the authentication process.

Example

OLAP/XMLA connectivity step by step. First, getXMLADataSources() obtains a list of all data sources by given URL (for example: 'http://olap.flexmonster.com/olap/msmdpump.dll') and getXMLAProviderName() returns dataSourceType. Second, getXMLACatalogs() gets all available catalogs for the first data source from the list of

available data sources. Then getXMLACubes() obtains a list of all cubes for the first catalog of the list of available catalogs. Finally, connectTo()uses all the information about data source to connect to it:

```
var proxyUrl = 'http://olap.flexmonster.com/olap/msmdpump.dll';
var dataSourceInfo = '';
var dataSourceType = '';
var catalog = '';
var cube = '';

function diagnostic() {
  getDataSources();
}

function getDataSources() {
  flexmonster.getXMLADataSources(proxyUrl, 'onDataSourceLoaded');
}

function onDataSourceLoaded(result) {
  dataSourceInfo = result[0];
  dataSourceType = flexmonster.getXMLAProviderName(proxyUrl);
  flexmonster.getXMLACatalogs(proxyUrl, dataSourceInfo, 'onCatalogsLoaded');
}

function onCatalogsLoaded(result) {
  catalog = result[0];
  flexmonster.getXMLACubes(proxyUrl, dataSourceInfo, catalog, 'onCubesLoaded');
}

function onCubesLoaded(result) {
  cube = result[0];
  flexmonster.connectTo({
   dataSourceType: dataSourceType,
   proxyUrl: proxyUrl,
   dataSourceInfo: dataSourceInfo,
   catalog: catalog,
   cube: cube
  });
}

diagnostic();
```

Try on JSFiddle(http://jsfiddle.net/flexmonster/hp6t9w8y/)
.

See also

getXMLAProviderName(/api/getxmlaprovidername/)

getXMLACatalogs(/api/getxmlacatalogs/)

getXMLACubes(/api/getxmlacubes/)

3.46. getXMLAProviderName

getXMLAProviderName(proxyURL:String, callbackHandler:String, username:String, password:String):String

[starting from version: 1.5]

One of four API calls of OLAP/XMLA connectivity diagnostics.

Returns dataSourceType for given proxyUrl. dataSourceType is a type of data source. For OLAP/XMLA connectivity it can be one of the following types: "microsoft analysis services", "mondrian", "iccube".

If getXMLADataSources() is called for proxyUrl first and then getXMLAProviderName() is called for the same proxyURL, getXMLAProviderName() will be ready to return dataSourceType immediately. Otherwise callbackHandler is needed to get dataSourceType.

Parameters

- proxyUrl – the path to proxy URL to the OLAP data source, such as Microsoft Analysis Services, Mondrian, icCube.
- callbackHandler (optional) – JS function which will be called when the component is ready to give dataSourceType.
- username (optional) – the name of user account at server. This parameter is necessary to complete authentication process.
- password (optional) – the password to user account at server. This parameter is necessary to complete authentication process.

Example

OLAP/XMLA connectivity step by step. First, getXMLADataSources() obtains a list of all data sources by given URL (for example: 'http://olap.flexmonster.com/olap/msmdpump.dll') and getXMLAProviderName() returns dataSourceType. Second, getXMLACatalogs() gets all available catalogs for the first data source from the list of available data sources. Then getXMLACubes() obtains a list of all cubes for the first catalog of the list of available catalogs. Finally, connectTo()uses all the information about data source to connect to it:

```
var proxyUrl = 'http://olap.flexmonster.com/olap/msmdpump.dll';
var dataSourceInfo = '';
var dataSourceType = '';
var catalog = '';
var cube = '';

function diagnostic() {
  getDataSources();
}

function getDataSources() {
  flexmonster.getXMLADataSources(proxyUrl, 'onDataSourceLoaded');
}

function onDataSourceLoaded(result) {
  dataSourceInfo = result[0];
  dataSourceType = flexmonster.getXMLAProviderName(proxyUrl);
  flexmonster.getXMLACatalogs(proxyUrl, dataSourceInfo, 'onCatalogsLoaded');
}
```

```
function onCatalogsLoaded(result) {
  catalog = result[0];
  flexmonster.getXMLACubes(proxyUrl, dataSourceInfo, catalog, 'onCubesLoaded');
}

function onCubesLoaded(result) {
  cube = result[0];
  flexmonster.connectTo({
   dataSourceType: dataSourceType,
   proxyUrl: proxyUrl,
   dataSourceInfo: dataSourceInfo,
   catalog: catalog,
   cube: cube
  });
}

diagnostic();
```

Open the example on JSFiddle(http://jsfiddle.net/flexmonster/hp6t9w8y/)
.

See also

getXMLADataSources(/api/getxmladatasources/)

getXMLACatalogs(/api/getxmlacatalogs/)

getXMLACubes(/api/getxmlacubes/)

3.47. gridColumnCount

[removed]

gridColumnCount method was removed in version 2.3.

3.48. gridRowCount

[removed]

gridRowCount method was removed in version 2.3.

3.49. load

load(url:String)

[starting from version: 1.4]

Loads report JSON file from the specified URL. XML reports are also supported in terms of backward compatibility.

Parameters

- url – the URL to report JSON file.

Examples

1) Load local report:

```
flexmonster.load("data/reports/report2.json");
```

2) Load remote report:

```
flexmonster.load("http://yourserver.com/script_which_returns_report_json.php");
```

Open the example on JSFiddle(http://jsfiddle.net/flexmonster/4h7LcL0m/)
.

See also

open(/api/open/)

save(/api/save/)

getReport(/api/getreport/)

setReport(/api/setreport/)

3.50. off

off(eventName:String, functionName:String)

[starting from version: 2.3]

Removes JS handlers for specified event.

Parameters

- eventName – the name of the component's event.
- functionName(optional) – the name of JS handler to remove. If not specified, all handlers will be removed for event.

Examples

```
//remove all handlers for 'cellclick' event
```

```
flexmonster.off('cellclick');

//add handler for 'cellclick' event
flexmonster.on('cellclick', 'onCellClick');

function onCellClick(result) {
  alert('The cell is clicked');
}

//remove specified handler for event
flexmonster.off('cellclick', 'onCellClick');

//please note, that if you added the listener the following way:
flexmonster.on('celldoubleclick', function () {
  alert('The cell is double clicked');
});
//it is impossible to remove such listener by name,
//so the only option here is to remove all the handlers by calling:
flexmonster.off('celldoubleclick');
```

Check the example on JSFiddle(https://jsfiddle.net/flexmonster/tygxsg5d/)
.

See also

on(/api/on/)

list of events(/api/events/)

3.51. on

on(eventName:String, function:Function|String)

[starting from version: 2.3]

Sets a JS function for the specified event. Check the list of events here(/api/events/)
.

Parameters

- eventName – the name of the component's event.
- function – the name of the JS function which will be a handler for the specified component's event or function itself.

Examples

```
flexmonster.on('celldoubleclick', function () {
  alert('The cell is double clicked');
});
```

```
//If you plan on removing some certain event handler later,
//add it the following way
flexmonster.on('cellclick', 'onCellClick');
function onCellClick(result) {
  alert('The cell is clicked');
}
```

Open the example on JSFiddle(https://jsfiddle.net/flexmonster/3hv3cho6/)
.

See also

off(/api/off/)

list of events(/api/events/)

3.52. open

open()

[starting from version: 1.6]

Opens local report file. Use this API call to open report JSON file from local file system. XML reports are also supported in terms of backward compatibility.

Example

```
flexmonster.open();
/*
* The component will provide the dialog box to choose the file
* which supposed to be loaded from the local file system.
* Select report JSON file to be loaded.
*/
```

Open the example on JSFiddle(http://jsfiddle.net/flexmonster/kt1bph9y/)
.

See also

load(/api/load/)

save(/api/save/)

getReport(/api/getreport/)

setReport(/api/setreport/)

3.53. openFieldsList

openFieldsList()

[starting from version: 1.4]

Opens Fields List.

Example

```
flexmonster.openFieldsList();
```

Open the example on JSFiddle(https://jsfiddle.net/flexmonster/g2exue63/)
.

See also

closeFieldsList(/api/closefieldslist/)

3.54. percentZoom

[removed]

percentZoom method was removed in version 2.3.

3.55. print

print(options:Object)

[starting from version: 1.4]

Prints the content of the grid or chart via OS print manager.

Parameters

options (optional) – Object. Can contain the following print properties:

- header (starting from v2.211) (optional) – String. Header is set in HTML format (tags, inline styles, img with base64 src), rendered in the browser and added as an image to the printed PDF file. The following tokens can be used: ##CURRENT-DATE##, ##PAGE-NUMBER##. They will be replaced by appropriate data.
- footer (starting from v2.211) (optional) – String. Footer is set in HTML format (tags, inline styles, img with base64 src), rendered in the browser and added as an image to the printed PDF file. The following tokens can be used: ##CURRENT-DATE##, ##PAGE-NUMBER##. They will be replaced by appropriate data.

Examples

1) Print the content:

```
flexmonster.print();
```

Try the example on JSFiddle(http://jsfiddle.net/flexmonster/8ns8yho1/)
.

2) Add header and footer:

```
var options = {
 header:"<div>##CURRENT-DATE##</div>",
 footer:"<div>##PAGE-NUMBER##</div>"
}
flexmonster.print(options);
```

Open the example on JSFiddle(http://jsfiddle.net/flexmonster/pq18xucm/)
.

See also

exportTo(/api/exportto/)

3.56. refresh

refresh()

[starting from version: 1.6]

Redraws the component.

This API call allows you to control redrawing of the component when you use the following API calls:

- addCondition()
- removeAllConditions()
- removeCondition()
- setFormat()
- setOptions()

Example

To add grid title and redraw the component to see changes, use the following API calls:

```
flexmonster.setOptions({
 grid: {
  title: "Table One"
 }
});
flexmonster.refresh();
```

Try on JSFiddle(https://jsfiddle.net/flexmonster/afr1g5Lf/)
.

See also

addCondition(/api/addcondition/)

removeAllConditions(/api/removeallconditions/)

removeCondition(/api/removecondition/)

setFormat(/api/setformat/)

setOptions(/api/setoptions/)

3.57. removeAllCalculatedMeasures

removeAllCalculatedMeasures()

[starting from version: 2.3]

Please note that this feature is available only for reports based on CSV or JSON data source.

Removes all calculated measures.
Example

To remove all calculated measures:

```
flexmonster.removeAllCalculatedMeasures();
```

Check out on JSFiddle(https://jsfiddle.net/flexmonster/79keh0ee/)
.

See also

addCalculatedMeasure(/api/addcalculatedmeasure/)

getAllMeasures(/api/getallmeasures/)

getMeasures(/api/getmeasures/)

removeCalculatedMeasure(/api/removecalculatedmeasure/)

3.58. removeAllConditions

removeAllConditions()

[starting from version: 1.5]

Removes all conditional formatting rules.

Use refresh() API call after to redraw the component and see changes.

Example

To remove all conditional formatting rules for the report use removeAllConditions(), as follows:

```
flexmonster.removeAllConditions();
flexmonster.refresh();
```

Open on JSFiddle(http://jsfiddle.net/flexmonster/v9qxnt0o/)
.

See also

addCondition(/api/addcondition/)

getCondition(/api/getcondition/)

getAllConditions(/api/getallconditions/)

removeCondition(/api/removecondition/)

refresh(/api/refresh/)

3.59. removeAllMeasures

[removed]

removeAllMeasures method was removed in version 2.3. To remove all calculated measures you can use removeAllCalculatedMeasures()(/api/removeallcalculatedmeasures/)

3.60. removeCalculatedMeasure

removeCalculatedMeasure(measureName:String)

[starting from version: 2.3]

Please note that this feature is available only for reports based on CSV or JSON data source.

Removes the calculated measure by measure unique name.

Parameters

- measureName – the measure unique name.

Example

To remove the calculated measure use removeCalculatedMeasure(), as follows:

```
var measureName = "Average Price";
flexmonster.removeCalculatedMeasure(measureName);
```

Open the example on JSFiddle(https://jsfiddle.net/flexmonster/79keh0ee/)
.

See also

addCalculatedMeasure(/api/addcalculatedmeasure/)

getAllMeasures(/api/getallmeasures/)

getMeasures(/api/getmeasures/)

removeAllCalculatedMeasures(/api/removeallcalculatedmeasures/)

3.61. removeCondition

removeCondition(id:String)

[starting from version: 1.5]

Removes the conditional formatting rule by id.

Use refresh() API call after to redraw the component and see changes.

Parameters

- id – the id of the conditional formatting rule.

Example

To remove the conditional formatting rule by id use removeCondition(), as follows:

```
var id = 1;
flexmonster.removeCondition(id);
flexmonster.refresh();
```

Check out on JSFiddle(http://jsfiddle.net/flexmonster/18ppLy4z/)
.

See also

addCondition(/api/addcondition/)

getCondition(/api/getcondition/)

getAllConditions(/api/getallconditions/)

removeAllConditions(/api/removeallconditions/)

refresh(/api/refresh/)

3.62. removeMeasure

[removed]

removeMeasure method was removed in version 2.3. To remove calculated measure you can use removeCalculatedMeasure()(/api/removecalculatedmeasure/)

3.63. removeSelection

removeSelection()

[starting from version: 1.4]

Removes a selection from cells on the grid.

Example

```
flexmonster.removeSelection();
```

Open the example on JSFiddle(https://jsfiddle.net/flexmonster/97fvkauL/)
.

See also

getSelectedCell(/api/getselectedcell/)

getCell(/api/getcell/)

3.64. runQuery

runQuery(slice:Slice Object(/api/slice-object/)
)

[starting from version: 1.6]

Runs a query with specified Slice Object(/api/slice-object/)
and displays the result data. Use this method to rearrange hierarchies on the axes or to compose a new report based on the current data source.

Parameters

- slice – Slice Object(/api/slice-object/)
  that defines rows, columns, pages, measures, filters, sorting etc.

Example

```
var slice =
{
 rows:
 [
  {uniqueName: "Country"}
 ],
 columns:
 [
  {uniqueName: "Color"}
 ],
 measures:
 [
  {uniqueName: "Price"}
 ]
};
flexmonster.runQuery(slice);
```

Open the example on JSFiddle(http://jsfiddle.net/flexmonster/jtqkf0yn/)
.

See also

getReport(/api/getreport/)

setReport(/api/setreport/)

3.65. save

save(params:Object)

[starting from version: 2.302]

You can use this method to save your current report to a specified location. Thus you can open it later with all fields, applied filters, and sortings same like you placed it. This method saves report in JSON format. XML reports are also supported in terms of backward compatibility.

Parameters

params object can have the following properties:

- filename – a default name of the file.
- destination (optional) – parameter defines how to save a generated file. File can be saved to "server" or "file". The default value is "file".
- callbackHandler (optional) – JS function which will be called when the report is saved.

- url (optional) – an URL to the server-side script which saves the generated file. The file is sent as a POST parameter. Use this parameter only if destination parameter is "server".
- embedData (optional) – specifies whether to save CSV/OCSV data within the report or not. Default value is false.
- reportType (optional) – String. Report can be saved in "json" or "xml" format. Default value is "json".
- withDefaults (optional) – Boolean. Indicates whether the default values for options will be included in the report (true) of not (false). The default value is false.
- withGlobals (optional) – Boolean. Indicates whether the options defined in global object will be included in the report (true) of not (false). The default value is false.

Examples

1) How to save a report to the local file system:

```
flexmonster.save({
    filename: 'myreport.json',
    destination: 'file'
});
```

Check out on JSFiddle(https://jsfiddle.net/flexmonster/b1a7gydf/)
.

2) How to save a report to the server:

```
flexmonster.save({
    filename: 'myreport.json',
    destination: 'server',
    url: 'http://yourserver.com/yourscript.php'
});
```

Please note that the server-side script should be created on your back-end to be able save reports to the server. And an *url* parameter is the path to this server-side script.

3) How to save a report and perform some JS code right after the report was already saved:

```
<button onclick="javascript:
    flexmonster.save({
    filename: 'myreport.json',
    destination: 'file',
    callbackHandler: 'reportSaved'
})">
 Save Report
</button>
<script type="text/javascript">
 function reportSaved() {
 // some JS code
 }
</script>
```

See also

load(/api/load/)

getReport(/api/getreport/)

setReport(/api/setreport/)

3.66. setBottomX

setBottomX(hierarchyName:String, num:Number, measureName:String)

[starting from version: 1.4]

Sets the Bottom X filter for the specified hierarchy and measure.

Parameters

- hierarchyName – the name of the hierarchy.
- num – number of elements to choose.
- measureName – the name of the measure on which Bottom X filter will be applied.

Example

```
flexmonster.setBottomX("Category", 3, "Price");

flexmonster.getFilterProperties("Category");

/*
method getFilterProperties() returns the following object:
{
  type: "bottom",
  members: [],
  measure: "Price",
  quantity: 3
}
*/

flexmonster.getFilter("Category");

/*
method getFilter() returns the following Array:
[
  {caption: "Clothing", hierarchyName: "Category", uniqueName: "category.[clothing]"
},
  {caption: "Accessories", hierarchyName: "Category", uniqueName: "category.[accesso
ries]"},
  {caption: "Components", hierarchyName: "Category", uniqueName: "category.[componen
ts]"}
]
*/
```

Check out on JSFiddle(http://jsfiddle.net/flexmonster/w3fq620e/)
.

See also

setTopX(/api/settopx/)

clearFilter(/api/clearfilter/)

getFilter(/api/getfilter/)

getFilterProperties(/api/getfilterproperties/)

setFilter(/api/setfilter/)

3.67. setChartTitle

[removed]

setChartTitle method was removed in version 2.3. Instead you can use setOptions()(/api/setoptions/)
.

3.68. setColumnWidth

[removed]

setColumnWidth method was removed in version 2.3.

3.69. setFilter

setFilter(hierarchyName:String, items:Array, negation:Boolean)

[starting from version: 1.4]

Sets the filter for the specified hierarchy.

Version: 1.7

One more property has been added – negation. If negation is false, setFilter() works the same as it works in previous versions, setFilter() tells the component to show the members of hierarchy specified in items. If negation is true, setFilter() tells the component to show all the members of hierarchy except the items. This allows for providing the shorter list of items when applying filter.

Parameters

- hierarchyName – the name of the hierarchy
- items – Array of hierarchy's members to be reflected/shown according to the applied filter.
- negation (optional) – It is false by default. It tells the component to show the members of hierarchy specified in items (true) or to show all the members of hierarchy except the items (false).

Examples

1) If you want to see data on ‘Cars’ and ‘Bikes’:

```
flexmonster.setFilter("Category",
  [
    "category.[cars]",
    "category.[bikes]"
  ]);
```

Try the example on JSFiddle(http://jsfiddle.net/flexmonster/mq32shu2/)
.

2) If you want to see all the categories except ‘Accessories’ now you can do this using the following code, where negation property is true:

```
flexmonster.setFilter("Category",
  [
    "category.[accessories]"
  ],
  true);
```

instead of

```
flexmonster.setFilter("Category",
  [
    "category.[bikes]",
    "category.[cars]",
    "category.[clothing]",
    "category.[components]"
  ]);
```

Check out on JSFiddle(http://jsfiddle.net/flexmonster/mq32shu2/)
.

See also

clearFilter(/api/clearfilter/)

getFilter(/api/getfilter/)

setBottomX(/api/setbottomx/)

setTopX(/api/settopx/)

3.70. setFormat

setFormat(format:Format Object(/api/format-object/)

, measureName:String)

[starting from version: 1.4]

Sets a default number format or the number format for the specified measure.

You can apply a format to all measures if you leave measureName parameter undefined. Or you can apply a format only to a specific measure if you specify measureName parameter.

Use refresh() API call after setting a format to redraw the component and see changes.

Parameters

- Format Object(/api/format-object/)
  – the object that contains the number format parameters.
- measureName (optional) – the unique name of the measure. Specify measure's unique name to apply a format to it or leave it undefined if you want to override a default format. Please note that if you want to override a default format, name property in format object should be an empty string – "".

Examples

1) How to override a default number format in run time:

```
var format = {
  name: "",
  decimalPlaces: 0,
  thousandsSeparator: ","
};
flexmonster.setFormat(format);
flexmonster.refresh();
```

Try how the sample works on JSFiddle(http://jsfiddle.net/flexmonster/qprzjx2z/)
.

2) How to change a currency symbol:

```
var format = flexmonster.getFormat("Price");
format.currencySymbol = "$";
//format.currencySymbol = "&#163;" // pound sterling
//format.currencySymbol = "&#8364;" // euro
//format.currencySymbol = "&#165;" // yen
flexmonster.setFormat(format, "Price");
flexmonster.refresh();
```

Open the example on JSFiddle(http://jsfiddle.net/flexmonster/kc8fq69y/)
.

See also

getFormat(/api/getformat/)

Format Object(/api/format-object/)

refresh(/api/refresh/)

3.71. setGridTitle

[removed]

setGridTitle method was removed in version 2.3. Instead you can use setOptions()(/api/setoptions/)
.

3.72. setHandler

[removed]

setHandler method was removed in version 2.3. Instead you can use on()(/api/on/)
and off()(/api/off/)
.

3.73. setLabels

[removed]

setLabels method was removed in version 2.3. Instead you can use report.localization from
setReport()(/api/setReport/)
.

3.74. setOptions

setOptions(options:Options Object(/api/options-object/)
)

[starting from version: 1.6]

Sets the component’s options.

Use refresh() API call after to redraw the component and see changes.

Parameters

Options Object(/api/options-object/)
– the object which contains the list of component’s options.

Examples

1) How to set grid title:

```
flexmonster.setOptions({
 grid: {
```

```
  title: "Table One"
 }
});
flexmonster.refresh();
```

Check the example on JSFiddle(https://jsfiddle.net/flexmonster/afr1g5Lf/)
.

2) How to turn off totals:

```
var options = flexmonster.getOptions();
options.grid.showTotals = false;
flexmonster.setOptions(options);
flexmonster.refresh();
```

Try on JSFiddle(https://jsfiddle.net/flexmonster/afr1g5Lf/)
.

3) How to set chart title:

```
flexmonster.setOptions({
 chart: {
  title: "Chart One"
 }
});
flexmonster.refresh();
```

See also

Options Object(/api/options-object/)

getOptions(/api/getoptions/)

refresh(/api/refresh/)

3.75. setReport

setReport(report:ReportObject(/api/report-object/)
| String)

[starting from version: 1.4]

Sets a report to be displayed in the component. Use this method to load and show previously saved reports.

Parameters

Report Object(/api/report-object/)
which describes the report and contains all its properties. XML reports are also supported in terms of backward compatibility.

Example

1) Set report:

```
<div id="pivotContainer">The component will appear here</div>
<script src="flexmonster/lib/jquery.min.js"></script>
<script src="flexmonster/flexmonster.js"></script>
<script>
var pivot = $("#pivotContainer").flexmonster({
 licenseKey: "XXXX-XXXX-XXXX-XXXX-XXXX"
});

</script>

<button onclick="setReport()">Set Report</button>
<script>
 function setReport() {
  var report = {
     dataSource: {
       filename: "http://cdn.flexmonster.com/2.3/data/data.csv"
      },
      options: {
       configuratorActive: false
      }
  }
  pivot.setReport(report);
 }
</script>
```

Open the example on JSFiddle(http://jsfiddle.net/flexmonster/L0n1r933/)
.

2) Swap two reports:

```
<div id="firstPivotContainer">The component will appear here</div>
<div id="secondPivotContainer">The component will appear here</div>
<script src="flexmonster/lib/jquery.min.js"></script>
<script src="flexmonster/flexmonster.js"></script>
<script">
var pivot1 = $("#firstPivotContainer").flexmonster({
 toolbar: true,
 report: {
  dataSource: {
   filename: "data1.csv"
  }
 },
 licenseKey: "XXXX-XXXX-XXXX-XXXX-XXXX"
});
```

```
var pivot2 = $("#secondPivotContainer").flexmonster({
 toolbar: true,
 report: {
  dataSource: {
   filename: "data2.csv"
  }
 },
 licenseKey: "XXXX-XXXX-XXXX-XXXX-XXXX"
});
</script>

<button onclick="javascript: swapReports()">Swap Reports</button>
<script>
  function swapReports() {
    var report1 = pivot1.getReport();
    var report2 = pivot2.getReport();

    pivot1.setReport(report2);
    pivot2.setReport(report1);
  }
</script>
```

Open on JSFiddle(http://jsfiddle.net/flexmonster/1dLjhu0s/)
.

See also

getReport(/api/getreport/)

Report Object(/api/report-object/)

open(/api/open/)

load(/api/load/)

save(/api/save/)

3.76. setRowHeight

[removed]

setRowHeight method was removed in version 2.3.

3.77. setSelectedCell

[removed]

setSelectedCell method was removed in version 2.3.

3.78. setSort

setSort(hierarchyName:String, sortType:String)

[starting from version: 1.4]

Sets the sort type to the specified hierarchy.

Parameters

- hierarchyName – String. The name of the hierarchy.
- sortType – String. The following sorting types can be applied: "asc", "desc", or "unsorted".

Example

```
flexmonster.setSort("Category", "desc");
```

Try the example on JSFiddle(http://jsfiddle.net/flexmonster/9h15aeye/)
.

See also

getSort(/api/getsort/)

sortValues(/api/sortvalues/)

3.79. setStyle

[removed]

setStyle method was removed in version 2.3.

3.80. setTopX

setTopX(hiearchyName:String, num:Number, measureName:String)

[starting from version: 1.4]

Sets the Top X filter for the specified hierarchy and measure.

Parameters

- hierarchyName – the name of the hierarchy.
- num – number of elements to choose.
- measureName – the name of the measure on which Top X filter will be applied.

Example

```
flexmonster.setTopX("Category", 2, "Price");
```

```
flexmonster.getFilterProperties("Category");

/*
method getFilterProperties() returns the following object:
{
  type: "top",
  members: [],
  measure: "Price",
  quantity: 2
}
*/

flexmonster.getFilter("Category");

/*
method getFilter() returns the following Array:
[
  {caption: "Cars", hierarchyName: "Category", uniqueName: "category.[cars]"},
  {caption: "Bikes", hierarchyName: "Category", uniqueName: "category.[bikes]"}
]
*/
```

Open on JSFiddle(http://jsfiddle.net/flexmonster/w3fq620e/)
.

See also

setBottomX(/api/setbottomx/)

clearFilter(/api/clearfilter/)

getFilter(/api/getfilter/)

getFilterProperties(/api/getfilterproperties/)

setFilter(/api/setfilter/)

3.81. showCharts

showCharts(type:String, multiple:Boolean)

[starting from version: 1.4]

Switches to the charts view and shows the chart of the specified type. The following chart types are supported: "bar", "bar_h" (Horizontal Bar), "line", "scatter", "pie", "bar_stack" and "bar_line" (starting from v1.9).
After showCharts() API call options.viewType property in report will be "charts".

Parameters

- type (optional) – the type of charts to show. Default value is "bar".
- multiple (optional) – to show one measure on the chart (false) or multiple (true) – as many measures as selected in the slice. Default value is false. (Available since version 1.9)

Examples

1) Show default chart type:

```
flexmonster.showCharts();
```

Try on JSFiddle(http://jsfiddle.net/flexmonster/zadn4cqn/)
.

2) Show pie chart:

```
flexmonster.showCharts("pie");
```

Check out the example on JSFiddle(http://jsfiddle.net/flexmonster/zadn4cqn/)
.

See also

showGrid(/api/showgrid/)

showGridAndCharts(/api/showgridandcharts/)

3.82. showGrid

showGrid()

[starting from version: 1.4]

Switches to the grid view.
After showGrid() API call options.viewType property in report will be "grid".

Example

```
flexmonster.showGrid();
```

Open on JSFiddle(http://jsfiddle.net/flexmonster/zadn4cqn/)
.

See also

showCharts(/api/showcharts/)

showGridAndCharts(/api/showgridandcharts/)

3.83. showGridAndCharts

showGridAndCharts(type:String, position:String, multiple:Boolean)

[starting from version: 1.9]

Switches to the grid and charts view and shows the chart of the specified type. The following chart types are supported: "bar", "bar_h" (Horizontal Bar), "line", "scatter", "pie", "bar_stack" and "bar_line".
After showGridAndCharts() API call options.viewType property in report will be "grid_charts".

Parameters

- type (optional) – the type of charts to show. Default value is "bar".
- position (optional) – Position of charts related to the grid. It can be "bottom", "top", "left" or "right". Default value is "bottom". (Available since version 2.2)
- multiple (optional) – to show one measure on the chart (false) or multiple (true) – as many measures as selected in the slice. Default value is false.

Examples

1) Show grid and charts:

```
flexmonster.showGridAndCharts();
```

Open on JSFiddle(http://jsfiddle.net/flexmonster/0b5c6ent/)
.

2) Show line chart at the left:

```
flexmonster.showGridAndCharts("line","left");
```

Check out on JSFiddle(http://jsfiddle.net/flexmonster/0b5c6ent/6/)
.

See also

showCharts(/api/showcharts/)

showGrid(/api/showgrid/)

3.84. sortValues

sortValues(axisName:String, type:String, tuple:Array, measureName:String)

[starting from version: 1.4]

Sorts values in a specific row or column in the pivot table.

Parameters

- axisName – String. The name of the axis to be sorted. It can be 'rows' or 'columns'. In order to define the sorting for numbers in a specific row, put 'rows' as the first parameter. If the API call will define the sorting for a column, put 'columns'.
- type – String. The type of sorting: 'asc' or 'desc'.
- tuple – Array. The tuple identifies the column or the row in the table and consists of member’s unique names.
- measureName – String. Identifies the measure on which sorting will be applied.

Example

```
flexmonster.sortValues(
 "columns",
 "asc",
   ["category.[bikes]", "color.[red]"],
   "Price"
);
```

Check out on JSFiddle(http://jsfiddle.net/flexmonster/dcs4tmoc/)
.

See also

getSort(/api/getsort/)

setSort(/api/setsort/)

3.85. updateData

updateData(params:Object)

[starting from version: 2.3]

Helps to update data for the report without cleaning the report. Only the dataSource is updated, whereas the slice, all defined options, number and conditional formatting, the scroll position stay the same.

Parameters

params is the object which contains connection parameters. All of them are optional.

Structure of params object:

- data – Property to set JSON data if it is already on the page. Contains the array of objects, where each object is an unordered set of name/value pairs. The first object can be used to define data types, captions, etc. Here is the list of supported properties that can be used in the first object of input array:

    - type – data type. Can be:
        - "string" – field contains string data. You will be able to aggregate it only with Count and Distinct Count aggregations. It will be sorted as string data.
        - "number" – field contains numeric data. You will be able to aggregate it with all different aggregations. It will be sorted as numeric data.
        - "level" – field is a level of hierarchy. This type is used together with other properties such as: hierarchy, level and parent
        - "month" – field contains months.
        - "weekday" – field contains days of the week.
        - "date" – field is a date. Such field will be split into 3 different fields: Year, Month, Day.
        - "date string" – field is a date. Such field will be formatted using date pattern (default is dd/MM/yyyy).
        - "year/month/day" – field is a date. You will see such date as a hierarchy: Year > Month > Day.
        - "year/quarter/month/day" – field is a date. You will see such date as a hierarchy: Year > Quarter > Month > Day.
        - "time" – field is a time (numeric data). Such field will be formatted using HH:mm pattern. Min, Max, Count and Distinct Count aggregations can be applied to it.
        - "datetime" – field is a date (numeric data). Such field will be formatted using dd/MM/yyyy HH:mm:ss pattern. Min, Max, Count and Distinct Count aggregations can be applied to it.
        - "id" – field is an id of the fact. Such field is used for editing data. This field will not be shown in Fields List.
        - "hidden" – field is hidden. This field will not be shown in Fields List.
    - caption – hierarchy caption.
    - hierarchy – hierarchy name, if the field is a level of hierarchy ("type":"level").
    - level – caption of the level, if the field is a level of hierarchy ("type":"level").
    - parent – caption of the parent level, if the field is a level of hierarchy ("type":"level").
    - dimensionUniqueName – dimension unique name. Can be used to group several fields under one dimension.
    - dimensionCaption – dimension caption. Is used as a display name (folder name in Fields List) when several fields are grouped under one dimension.
- dataSourceType – type of data source. The component supports the following types: "microsoft analysis services", "mondrian", "iccube", "csv", "ocsv", "json".
- proxyUrl – the path to proxy URL to the OLAP data source, such as Microsoft Analysis Services, Mondrian, icCube (only for "microsoft analysis services", "mondrian", "iccube" data source types)
- dataSourceInfo – the service info of the OLAP data source (only for "microsoft analysis services", "mondrian", "iccube" data source types)
- catalog – the data source catalog name of the OLAP data source (only for "microsoft analysis services", "mondrian", "iccube" data source types)
- cube – given catalog's cube's name of the OLAP data source (only for "microsoft analysis services", "mondrian", "iccube" data source types)
- filename – the URL to CSV file or to server side script which generates CSV data (only for "csv", "ocsv" and "json" data source type)
- browseForFile – this boolean parameter defines whether you want to load CSV file from the local file system (true) or not (false). It is *false* by default. (only for "csv", "ocsv" and "json" data source type)
- fieldSeparator – defines specific fields separator to split CSV row (only for "csv" data source type). There is no need to define it if CSV fields are separated by , or ;. This property is used if another char separates fields. For example, if you use TSV, where tab char is used to separate fields in row, fieldSeparator parameter should be defined explicitly.

Examples

1) Update data from Microsoft Analysis Services:

```
flexmonster.updateData({
```

```
  dataSourceType: 'microsoft analysis services',
  proxyUrl: 'http://olap.flexmonster.com/olap/msmdpump.dll',
  dataSourceInfo: 'Provider=MSOLAP; Data Source=extranet;',
  catalog: 'Adventure Works DW Standard Edition',
  cube: 'Adventure Works'
});
```

Open the example on JSFiddle(http://jsfiddle.net/flexmonster/xf5rn80j/)
.

2) Update data from CSV data source:

```
flexmonster.updateData({
  dataSourceType: 'csv',
  filename: 'data.csv'
});
```

Try on JSFiddle(http://jsfiddle.net/flexmonster/xf5rn80j/)
.

3) Update data from JSON inline data:

```
var jsonData = [{
  "Color": { "type": "string" },
  "M": {
    "type": "month",
    "dimensionUniqueName": "Days",
    "dimensionCaption": "Days",
    "caption": "Month"
  },
  "W": {
    "type": "weekday",
    "dimensionUniqueName": "Days",
    "dimensionCaption": "Days",
    "caption": "Week Day"
  },
  "Country": {
    "type": "level",
    "hierarchy": "Geography",
    "level": "?ountry"
  },
  "State": {
    "type": "level",
    "hierarchy": "Geography",
    "level": "State",
    "parent": "?ountry"
  },
  "City": {
    "type": "level",
```

```
    "hierarchy": "Geography",
    "parent": "State"
  },
  "Price": { "type": "number" },
  "Quantity": { "type": "number" }
}, {
  "Color": "green",
  "M": "September",
  "W": "Wed",
  "Country": "Canada",
  "State": "Ontario",
  "City": "Toronto",
  "Price": 174,
  "Quantity": 22
}];
flexmonster.updateData({ data: jsonData });
```

Check out on JSFiddle(http://jsfiddle.net/flexmonster/xf5rn80j/)
.

See also

connectTo(/api/connectto/)

addJSON(/api/addjson/)

3.86. zoomTo

[removed]

zoomTo method was removed in version 2.3.

3.87. jsCellClickHandler

[removed]

jsCellClickHandler event was removed in version 2.3. Instead you can use cellclick(/api/cellclick/)
.

3.88. jsFilterOpenHandler

[removed]

jsFilterOpenHandler event was removed in version 2.3. Instead you can use filteropen(/api/filteropen/)
.

3.89. jsFieldsListCloseHandler

[removed]

jsFieldsListCloseHandler event was removed in version 2.3. Instead you can use fieldslistclose(/api/fieldslistclose/)
.

3.90. jsFieldsListOpenHandler

[removed]

jsFieldsListOpenHandler event was removed in version 2.3. Instead you can use fieldslistopen(/api/fieldslistopen/)
.

3.91. jsFullScreenHandler

[removed]

jsFullScreenHandler event was removed in version 2.3.

3.92. jsPivotCreationCompleteHandler

[removed]

jsPivotCreationCompleteHandler event was removed in version 2.3. Instead you can use reportcomplete(/api/reportcomplete/)
.

3.93. jsPivotUpdateHandler

[removed]

jsPivotUpdateHandler event was removed in version 2.3. Instead you can use update(/api/update/)
.

3.94. jsReportChangeHandler

[removed]

jsReportChangeHandler event was removed in version 2.3. Instead you can use reportchange(/api/reportchange/)
.

3.95. jsReportLoadedHandler

[removed]

jsReportLoadedHandler event was removed in version 2.3. Instead you can use reportcomplete(/api/reportcomplete/)
.

4. Events

Flexmonster Pivot Table & Charts Component offers plenty of events. Try on()(/api/on/)
and off()(/api/off/)
functions to handle any event.

List of events:

reportcomplete(/api/reportcomplete/)	triggered when the operations can be performed with the component (data source file or OLAP structure was loaded successfully and the grid/chart was rendered)
beforetoolbarcreated(/api/beforetoolbarcreated/)	triggered before the creation of Toolbar
cellclick(/api/cellclick/)	triggered when a cell is clicked on the grid
celldoubleclick(/api/celldoubleclick/)	triggered when a cell is double clicked on the grid
datachanged(/api/datachanged/)	triggered after the user edits data
dataerror(/api/dataerror/)	triggered when some error occurred during the loading of data
datafilecancelled(/api/datafilecancelled/)	triggered when Open file dialog was opened and customer clicks Cancel button
dataloaded(/api/dataloaded/)	triggered when the component loaded data
exportcomplete(/api/exportcomplete/)	triggered after the export file was generated successfully
exportstart(/api/exportstart/)	triggered when the export of grid or chart starts
fieldslistclose(/api/fieldslistclose/)	triggered when the built-in Fields List is closed
fieldslistopen(/api/fieldslistopen/)	triggered when the built-in Fields List is opened
filteropen(/api/filteropen/)	triggered when a filter icon is pressed
loadingdata(/api/loadingdata/)	triggered when data starts loading from from local or remote CSV, JSON or after report was loaded
loadinglocalization(/api/loadinglocalization/)	triggered when localization file starts loading
loadingolapstructure(/api/loadingolapstructure/)	triggered when structure of OLAP cube starts loading
loadingreportfile(/api/loadingreportfile/)	triggered when a report file started loading
localizationerror(/api/localizationerror/)	triggered when some error appeared while loading localization file
localizationloaded(/api/localizationloaded/)	triggered when localization file was loaded
olapstructureerror(/api/olapstructureerror/)	triggered when some error appeared while loading OLAP structure
olapstructureloaded(/api/olapstructureloaded/)	triggered when OLAP structure was loaded
openingreportfile(/api/openingreportfile/)	triggered when customer uses menu Open -> Local report or API method open()(/api/open/) is called
printcomplete(/api/printcomplete/)	triggered after OS print manager was closed

printstart(/api/printstart/)	triggered when print(options:Object)(/api/print/) method was called or Export > Print was selected in the Toolbar
querycomplete(/api/querycomplete/)	triggered after the data query was complete
queryerror(/api/queryerror/)	triggered if some error occurred while running the query
ready(/api/ready/)	triggered when the component's initial configuration completed and the component is ready to receive API calls
reportchange(/api/reportchange/)	triggered when a report is changed in the component
reportfilecancelled(/api/reportfilecancelled/)	triggered when customer uses menu Open -> Local report and clicks Cancel button
reportfileerror(/api/reportfileerror/)	triggered when some error occurred during the loading of the report file
runningquery(/api/runningquery/)	triggered before data query is started
update(/api/update/)	triggered when the change occurred in the component

4.1. beforetoolbarcreated

beforetoolbarcreated:String
[starting from version: 2.3]

It is triggered before the creation of Toolbar. Use this event to override default Toolbar and customize it without changing flexmonster.toolbar.js. As an input beforetoolbarcreated gets toolbar object. Override existing tabs and set custom ones using toolbar.getTabs() function. Each tab object should have the following structure:

- title – String. Label of the tab.
- id – String. Id used in CSS styles.
- handler – String. Name of the function that handles click on this tab.
- args (optional) – Any. Argument to pass to handler.
- menu (optional) – Array. Dropdown menu items.
- mobile (optional) – Boolean. If false – doesn't show on mobile devices.
- ios (optional) – Boolean. If false – doesn't show on iOS devices.
- android (optional) – Boolean. If false – doesn't show on Android devices.

Example

Hide all default tabs and add a new one to load a CSV file:

```
$("#pivot").flexmonster({
    toolbar: true,
    beforetoolbarcreated: customizeToolbar
});

function customizeToolbar(toolbar) {
    // override tabs
    toolbar.getTabs = function() {
        var tabs = [];
        tabs.push({
        id: "fm-tab-connect",
```

```
        title: this.Labels.connect,
        handler: function() {
            this.pivot.connectTo({
                dataSourceType: "csv",
                filename: "http://cdn.flexmonster.com/2.3/data/data.csv"
            });
        }
        });
        return tabs;
    }
}
```

Try the example on JSFiddle(http://jsfiddle.net/flexmonster/m0gwv0at/)
.

4.2. cellclick

cellclick:String
[starting from version: 2.3]

It is triggered when a cell is clicked on the grid.

Data passed to the handler

cell – Cell Object(/api/cell-object/)
which contains information about the clicked cell.

Example

```
flexmonster.on('cellclick', function (cell) {
          alert(
            "Click on cell - row: "
            + cell.rowIndex + ", column: "
            + cell.columnIndex
            + ", label: "
            + cell.label);
});
```

Open the example on JSFiddle(https://jsfiddle.net/flexmonster/h8otpuLh/)
.

See also

on(/api/on/)

off(/api/off/)

4.3. celldoubleclick

celldoubleclick:String

[starting from version: 2.3]

It is triggered when a cell is double clicked on the grid.

Data passed to the handler

cell – Cell Object(/api/cell-object/)
which contains information about the clicked cell.

Example

```
flexmonster.on('celldoubleclick', function (cell) {
          alert(
            "Double click on cell - row: "
            + cell.rowIndex + ", column: "
            + cell.columnIndex
            + ", label: "
            + cell.label);
});
```

Open the example on JSFiddle(http://jsfiddle.net/flexmonster/409xv0hp/)
.

4.4. datachanged

datachanged:String

[starting from version: 2.306]

It is triggered after the user edits data. Editing is available through the Flat view or Drill Through popup. This event should be used when editing is enabled (editing: true). When datachanged is triggered it gets as an input the object with data property. This property is the array of objects. Each of them has the following structure:

- id – String | Number. Row number (by default) or ID column.
- field – String. The unique name of the hierarchy that was updated.
- value – String. New value.

If you want to get the value of the ID column with an event, you should set column type id for this column (read more about data types in JSON(/doc/data-types-in-json/)
and CSV(/doc/data-types-in-csv/)
). Such field will not be shown in Fields List but its value will be sent within the event. This helps to identify which row was changed.

Example

```
flexmonster.on('datachanged', function (e) {
```

```
    alert("Data changed - id: "
            + e.data[0].id + ", field: "
            + e.data[0].field
            + ", value: "
            + e.data[0].value);
});
```

Open the example on JSFiddle(http://jsfiddle.net/flexmonster/vy78qb4o/)
.

See also

Report Object(/api/report-object/)

4.5. dataerror

dataerror:String

[starting from version: 2.3]

It is triggered when some error occurred during the loading of data. Please use olapstructureerror for OLAP data sources.

Example

```
flexmonster.on('dataerror', function () {
  alert('Error with data!');
});
```

Open the example on JSFiddle(http://jsfiddle.net/flexmonster/quzf2gko/)
.

See also

loadingdata(/api/loadingdata/)

dataloaded(/api/dataloaded/)

olapstructureerror (/api/olapstructureerror/)

4.6. datafilecancelled

datafilecancelled:String

[starting from version: 2.3]

It is triggered when Open file dialog was opened and customer clicks Cancel button. It happens when:

1. customer uses menu Connect -> to local CSV/JSON
2. API method updateData()(/api/updateData/)
   is called with parameter browseForFile = true

Example

```
flexmonster.on('datafilecancelled', function () {
  alert('Data file cancelled!');
});
```

Try on JSFiddle(http://jsfiddle.net/flexmonster/6xz51tws/)
.

See also

dataloaded(/api/dataloaded/)

dataerror(/api/dataerror/)

4.7. dataloaded

dataloaded:String

[starting from version: 2.3]

It is triggered when the component loaded data. To track possible errors use dataerror. Please use olapstructureloaded for OLAP data sources.

Example

```
flexmonster.on('dataloaded', function () {
  alert('Data loaded!');
});
```

Check out the example on JSFiddle(http://jsfiddle.net/flexmonster/quzf2gko/)
.

See also

dataerror(/api/dataerror/)

olapstructureloaded(/api/olapstructureloaded/)

4.8. exportcomplete

exportcomplete:String

[starting from version: 2.3]

It is triggered after the export file was generated successfully. That means the file is ready to be saved. To track when the export process starts use exportstart.

Example

```
flexmonster.on('exportcomplete', function () {
    alert('Exporting completed!');
});
```

Check out the example on JSFiddle(http://jsfiddle.net/flexmonster/40df9vz4/)
.

See also

exportstart(/api/exportstart/)

4.9. exportstart

exportstart:String

[starting from version: 2.3]

It is triggered when the export of grid or chart starts. Export is possible to CSV, HTML, PDF, Image or Excel format. Use exportTo(type:String, params:Object, callbackHandler:String) for more export options. To make sure that export process was successful and the file is ready to be saved use exportcomplete.

Example

```
flexmonster.on('exportstart', function () {
    alert('Exporting started!');
});
```

Try the example on JSFiddle(http://jsfiddle.net/flexmonster/40df9vz4/)
.

See also

exportcomplete(/api/exportcomplete/)

exportto(/api/exportto/)

4.10. fieldslistclose

fieldslistclose:String

[starting from version: 2.3]

It is triggered when the built-in Fields List is closed.

Example

```
flexmonster.on('fieldslistclose', function () {
  alert('Field list is closed!');
});
```

Check the example on JSFiddle(https://jsfiddle.net/flexmonster/2pgzjyse/)
.

See also

fieldslistopen(/api/fieldslistopen/)

4.11. fieldslistopen

fieldslistopen:String

[starting from version: 2.3]

It is triggered when the built-in Fields List is opened.

Example

```
flexmonster.on('fieldslistopen', function () {
  alert('Field list is opened!');
});
```

Open the example on JSFiddle(https://jsfiddle.net/flexmonster/2pgzjyse/)
.

See also

fieldslistclose(/api/fieldslistclose/)

4.12. filteropen

filteropen:String

[starting from version: 2.3]

It is triggered when a filter icon is pressed. If this handler is defined the native Filter Window will not appear. In other words, the component's handler will be replaced by the JS handler. Use this handler to create your own customized Filter Window.

Data passed to the handler

params – the object which contains the following Filter Window parameters:

- hierarchy – object that describes hierarchy
- rect – object that describes filter cell's x, y, width and height

Example

```
flexmonster.on('filteropen', function (params) {
  alert('The filter is opened!');
});
```

Open the example on JSFiddle(http://jsfiddle.net/flexmonster/drwyzj8o/)
.

See also

ready(/api/ready/)

4.13. loadingdata

loadingdata:String

[starting from version: 2.3]

It is triggered when data starts loading from from local or remote CSV, JSON or after report was loaded. To track possible errors use dataerror. To make sure that loading of the data was successful use dataloaded. Please use loadingolapstructure for OLAP data sources.

Example

```
flexmonster.on('loadingdata', function () {
  alert('Loading data!');
});
```

Open the example on JSFiddle(http://jsfiddle.net/flexmonster/quzf2gko/)
.

See also

loadingolapstructure(/api/loadingolapstructure/)

dataloaded(/api/dataloaded/)

dataerror(/api/dataerror/)

datafilecancelled(/api/datafilecancelled/)

4.14. loadinglocalization

loadinglocalization:String

[starting from version: 2.3]

It is triggered when localization file starts loading. To track possible errors use localizationerror. To make sure that loading of the localization file was successful use localizationloaded.

Example

```
flexmonster.on('loadinglocalization', function () {
  alert('Loading localization file!');
});
```

Try on JSFiddle(http://jsfiddle.net/flexmonster/5Lcu5k6o/)
.

See also

localizationloaded(/api/localizationloaded/)

localizationerror(/api/localizationerror/)

4.15. loadingolapstructure

loadingolapstructure:String

[starting from version: 2.3]

It is triggered when structure of OLAP cube starts loading. Once the connection was established the component begins loading OLAP structure. To make sure that structure was loaded, use olapstructureloaded. To track any errors use olapstructureerror.

Example

```
flexmonster.on('loadingolapstructure', function () {
  alert('Loading olap structure!');
});
```

Open the example on JSFiddle(http://jsfiddle.net/flexmonster/b7qz3m9n/)
.

See also

olapstructureloaded(/api/olapstructureloaded/)

olapstructureerror(/api/olapstructureerror/)

4.16. loadingreportfile

loadingreportfile:String

[starting from version: 2.3]

It is triggered when a report file started loading. To make sure that loading of the report was successful use reportcomplete. To catch any errors use reportfileerror.

Example

```
flexmonster.on('loadingreportfile', function () {
  alert('Loading report file!');
});
```

Open the example on JSFiddle(http://jsfiddle.net/flexmonster/sd4mfet9/)
.

See also

reportcomplete(/api/reportcomplete/)

reportfileerror(/api/reportfileerror/)

4.17. localizationerror

localizationerror:String

[starting from version: 2.3]

It is triggered when some error appeared while loading localization file.

Example

```
flexmonster.on('localizationerror', function () {
  alert('Error with localization file!');
});
```

Check out on JSFiddle(http://jsfiddle.net/flexmonster/5Lcu5k6o/)
.

See also

loadinglocalization(/api/loadinglocalization/)

localizationloaded(/api/localizationloaded/)

4.18. localizationloaded

localizationloaded:String

[starting from version: 2.3]

It is triggered when localization file was loaded. To track possible errors use localizationerror.

Example

```
flexmonster.on('localizationloaded', function () {
  alert('Localization file was loaded!');
});
```

Open the example on JSFiddle(http://jsfiddle.net/flexmonster/5Lcu5k6o/)
.

See also

loadinglocalization(/api/loadinglocalization/)

localizationerror(/api/localizationerror/)

4.19. olapstructureerror

olapstructureerror:String

[starting from version: 2.3]

It is triggered when some error appeared while loading OLAP structure.

Example

```
flexmonster.on('olapstructureerror', function () {
  alert('Error with olap structure!');
});
```

Check out on JSFiddle(http://jsfiddle.net/flexmonster/b7qz3m9n/)
.

See also

olapstructureloaded(/api/olapstructureloaded/)

loadingolapstructure(/api/loadingolapstructure/)

4.20. olapstructureloaded

olapstructureloaded:String

[starting from version: 2.3]

It is triggered when OLAP structure was loaded. To track any errors use olapstructureerror.

Example

```
flexmonster.on('olapstructureloaded', function () {
  alert('Olap structure loaded!');
});
```

Try on JSFiddle(http://jsfiddle.net/flexmonster/b7qz3m9n/)
.

See also

loadingolapstructure(/api/loadingolapstructure/)

olapstructureerror(/api/olapstructureerror/)

4.21. openingreportfile

openingeportfile:String

[starting from version: 2.3]

It is triggered when:

1. customer uses menu Open -> Local report
2. API method open()(/api/open/)
   is called

To track when this report will be loading, use loadingreportfile. To make sure that loading of the report was successful use reportcomplete.

Example

```
flexmonster.on('openingreportfile', function () {
  alert('Opening report file!');
});
```

Try on JSFiddle(http://jsfiddle.net/flexmonster/venyhc5a/)
.

See also

loadingreportfile(/api/loadingreportfile/)

reportcomplete(/api/reportcomplete/)

reportfileerror(/api/reportfileerror/)

reportfilecancelled(/api/reportfilecancelled/)

4.22. printcomplete

printcomplete:String

[starting from version: 2.3]

It is triggered after OS print manager was closed. It can be closed when the user clicks ‘Print’ or ‘Cancel’. printcomplete will be triggered in both cases. To track when OS print manager was opened use printstart.

Example

```
flexmonster.on('printcomplete', function () {
    alert('Printing completed!');
});
```

Check the example on JSFiddle(http://jsfiddle.net/flexmonster/40df9vz4/)
.

See also

printstart(/api/printstart/)

4.23. printstart

printstart:String

[starting from version: 2.3]

It is triggered when print(options:Object) method was called or Export > Print was selected in the Toolbar. To track when OS print manager was closed use printcomplete.

Example

```
flexmonster.on('printstart', function () {
    alert('Printing started!');
});
```

Open the example on JSFiddle(http://jsfiddle.net/flexmonster/40df9vz4/)
.

See also

printcomplete(/api/printcomplete/)

print(/api/print/)

4.24. querycomplete

runningquery:String

[starting from version: 2.3]

It is triggered after the data query was complete. To track any errors use queryerror.

Example

```
flexmonster.on('querycomplete', function () {
  alert('Query is complete!');
});
```

Try on JSFiddle(http://jsfiddle.net/flexmonster/rnhzjxh3/)
.

See also

runningquery(/api/runningquery/)

queryerror(/api/queryerror/)

4.25. queryerror

runningquery:String

[starting from version: 2.3]

It is triggered if some error occurred while running the query.

Example

```
flexmonster.on('queryerror', function () {
  alert('Query error!');
});
```

Check the example on JSFiddle(http://jsfiddle.net/flexmonster/rnhzjxh3/)
.

See also

querycomplete(/api/querycomplete/)

runningquery(/api/runningquery/)

4.26. ready

ready:String

[starting from version: 2.3]

It is triggered when the component's initial configuration completed and the component is ready to receive API calls. Please note that all JS API calls are blocked until the component's ready event is dispatched. In other words, when ready handler is called you know that the component's creation completed and now you can use API calls.

Also, when ready is called data structure is still loading if report parameter was specified in createPivotComponent(). To retrieve information about data source structure use reportcomplete. If you want to track changes in a report object, we recommend using reportchange.

Examples

1) How to set JSON data using API call:

addJSON() API call can be used to set JSON data only after ready is called by Flexmonster Pivot Table component. Please see the sample code below:

```
var jsonData = [
  {
    "Color":"green",
    "M":"September",
    "W":"Wed",
    "?ountry" : "Canada",
    "State" : "Ontario",
    "City" : "Toronto",
    "Price":174,
    "Quantity":22
  },
  {
    "Color":"red",
    "M":"March",
    "W":"Mon",
    "Time":"1000",
    "?ountry" : "USA",
    "State" : "California",
    "City" : "Los Angeles",
    "Price":1664,
    "Quantity":19
  }
];

/**
* The data can be set using addJSON() and setReport() API calls
* only after ready event is dispatched by Flexmonster Component
*/
var pivot = $("#pivotContainer").flexmonster({
 toolbar: true,
 ready: function () {
  pivot.addJSON(jsonData);
 }
});

/**
* flexmonster addJSON() function illustrates how to set JSON data
* that is already on the page and define a slice based on this data.
*/
function flexmonsterAddJSON() {
 flexmonster.addJSON(jsonData);
 var slice = {
  rows: [{ uniqueName: "Color" }],
  columns: [{ uniqueName: "W" }, { uniqueName: "[Measures]" }],
  measures: [{ uniqueName: "Price"}]};
 flexmonster.runQuery(slice);
}
```

Check out the example on JSFiddle(http://jsfiddle.net/flexmonster/ea1jd593/)
.

2) How to define report from JS:

Defining a report from JS after the component is created:

```
var pivot = $("#pivotContainer").flexmonster({
 toolbar: true,
 ready: function () {
  var report = {
   dataSource: {
    filename: "data.csv"
   },

   options: {
    grid: {
     title: "Report set via JS API"
    }
   },

   slice: {
    rows: [
     {uniqueName: "Country"},
     {uniqueName: "[Measures]"}
    ],
    columns: [
     {uniqueName: "Color"}
    ],
    measures: [
     {uniqueName: "Price"}
    ]
   }
  };
  pivot.setReport(report);
 }
});
```

Try on JSFiddle(http://jsfiddle.net/flexmonster/dqh3b80e/)
.

See also

flexmonster(/api/flexmonster/)

reportcomplete(/api/reportcomplete/)

reportchange(/api/reportchange/)

4.27. reportchange

reportchange:String

[starting from version: 2.3]

It is triggered when a report is changed in the component. If you want to track changes in a report object or perform some action on a report change, please use it.

Example

```
flexmonster.on('reportchange', function () {
  alert('Report changed!');
});
```

Open the example on JSFiddle(http://jsfiddle.net/flexmonster/w3a7hxs9/)
.

See also

ready(/api/ready/)

update(/api/update/)

reportcomplete(/api/reportcomplete/)

4.28. reportcomplete

reportcomplete:String

[starting from version: 2.3]

It is triggered when OLAP structure or data from report, localization file and all data source files were loaded successfully and the grid/chart was rendered. This event means the operations can be performed with the component.

Example

```
flexmonster.on('reportcomplete', function () {
  alert('Report completed!');
});
```

Try on JSFiddle(http://jsfiddle.net/flexmonster/pfa5z129/)
.

See also

reportchange(/api/reportchange/)

4.29. reportfilecancelled

reportfilecancelled:String

[starting from version: 2.3]

It is triggered when customer uses menu Open -> Local report and clicks Cancel button.

Example

```
flexmonster.on('reportfilecancelled', function () {
  alert('Opening of report file was cancelled!');
});
```

Open the example on JSFiddle(http://jsfiddle.net/flexmonster/sd4mfet9/)
.

See also

openingreportfile(/api/openingreportfile/)

loadingreportfile(/api/loadingreportfile/)

reportcomplete(/api/reportcomplete/)

reportfileerror(/api/reportfileerror/)

4.30. reportfileerror

reportfileerror:String

[starting from version: 2.3]

It is triggered when some error occurred during the loading of the report file.

Example

```
flexmonster.on('reportfileerror', function () {
  alert('Report file error!');
});
```

Open the example on JSFiddle(http://jsfiddle.net/flexmonster/sd4mfet9/)
.

See also

loadingreportfile(/api/loadingreportfile/)

reportcomplete(/api/reportcomplete/)

4.31. reportfileloaded

reportfileloaded:String

[starting from version: 2.3]

It is triggered when the component loaded a report file. Next, the component starts the process of loading the data. For tracking that process use loadingdata.

Example

```
flexmonster.on('reportfileloaded', function () {
  alert('Report file loaded!');
});
```

See also

ready(/api/ready/)

update(/api/update/)

reportchange(/api/reportchange/)

loadingdata(/api/loadingdata/)

4.32. runningquery

runningquery:String

[starting from version: 2.3]

It is triggered before data query is started. It is used for both cases when data is already loaded and stored inside the component's local storage or when it is necessary to load data from the external data storage in case of OLAP data source. Data query is started when:

- Any filter was used;
- Slice was changed;
- Drill up/drill down was used;
- Columns or rows were expanded;
- Drill through was used.

To make sure that running of the query was successful use querycomplete. To track any errors use queryerror.

Example

```
flexmonster.on('runningquery', function () {
  alert('Query is running!');
```

```
});
```

Open the example on JSFiddle(http://jsfiddle.net/flexmonster/rnhzjxh3/)
.

See also

querycomplete(/api/querycomplete/)

queryerror(/api/queryerror/)

4.33. update

update:String

[starting from version: 2.3]

It is triggered when the component loaded data, updated data slice, filter or sort. In other words, when the change occurred in the component. It is good to use it to retrieve information about the data source structure.
If you want to track changes in a report object, we recommend using reportchange instead.

Example

```
flexmonster.on('update', function () {
  alert('Updated!');
});
```

Check the example on JSFiddle(http://jsfiddle.net/flexmonster/jr890f2m/)
.

See also

ready(/api/ready/)

reportchange(/api/reportchange/)