API Reference

by Flexmonster

1. API reference

1.1. Introduction

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

This API documentation describes objects (#objects), calls (#calls) and events (#events).

The following example illustrates how to embed the component via initial new Flexmonster() call:

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

<script>
    var pivot = new Flexmonster({
        container: "pivotContainer",
        toolbar: true
    });
</script>
```

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

List of objects

Report Object (/api/report-object/)	contains all the possible aspects of pivot tables and pivot charts configuration
Data Source Object (/api/data-source-object/)	contains information about the data source
Mapping Object (/api/mapping-object/)	contains information about field data types, captions, multi-level hierarchies, and other view configurations of the data source
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
Filter Object (/api/filter-object/)	contains filtering information
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
Table Sizes Object (/api/table-sizes-object/)	contains information about table sizes
Cell Data Object (/api/cell-object/)	contains information about the cell
Chart Data Object (/api/chart-data-object/)	contains information about the chart segment
Toolbar Object (/api/toolbar-object/)	contains information about the Toolbar
Chart Legend Data Object (/api/chart-legend-data-object/)	contains information about the chart legend element

List of API calls

addCalculatedMeasure (/api/addcalculatedmeasure/)	adds calculated measure
addCondition (/api/addcondition/)	adds a conditional formatting rule
alert (/api/alert/)	shows an alert pop-up window with a custom message
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 the Field 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
customizeAPIRequest (/api/customizeapirequest/)	allows customizing the request before it is sent to a server
customizeCell (/api/customizecell/)	allows customizing of separate cells
customizeChartElement (/api/customizechartelement/)	allows customizing separate chart elements in Flexmonster Charts
customizeContextMenu (/api/customizecontextmenu/)	allows customizing context menu
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
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
getFilter (/api/getfilter/)	returns the filtered members for the specified hierarchy
getFlatSort (/api/getflatsort/)	returns an array of objects defining the sorting on the flat grid
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
getReportFilters (/api/getreportfilters/)	returns a list of hierarchies selected in the report slice for Report Filters
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

getTableSizes (/api/gettablesizes/)	returns table sizes that are set for the component
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
openCalculatedValueEditor (/api/opencalculatedvalueeditor/)	opens the calculated value pop-up window editor
openFieldsList (/api/openfieldslist/)	opens the Field List
openFilter (/api/openfilter/)	opens the filter pop-up window for the specified hierarchy
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 rows, columns, measures and reportFilters from Slice Object (/api/slice-object/) and displays the result data
save (/api/save/)	saves your current report to a specified location
scrollToColumn (/api/scrolltocolumn/)	scrolls the grid to the specified column
scrollToRow (/api/scrolltorow/)	scrolls the grid to the specified row
setFilter (/api/setfilter/)	sets the filter for the specified hierarchy
setFlatSort (/api/setflatsort/)	sets the flat table multi-column sorting
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
setTableSizes (/api/settablesizes/)	sets table sizes for the component
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
sortingMethod (/api/sortingmethod/)	sets custom sorting for hierarchy members
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)
afterchartdraw (/api/afterchartdraw/)	triggered after chart rendering
aftergriddraw (/api/aftergriddraw/)	triggered after grid rendering
beforegriddraw (/api/beforegriddraw/)	triggered before grid rendering
beforetoolbarcreated (/api/beforetoolbarcreated/)	triggered before the creation of the Toolbar
cellclick (/api/cellclick/)	triggered when a cell is clicked on the grid
chartclick (/api/chartclick/)	triggered when a chart element is clicked
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
drillthroughclose (/api/drillthroughclose/)	triggered when the drill-through view is closed
drillthroughopen (/api/drillthroughopen/)	triggered when the drill-through view is opened
exportcomplete (/api/exportcomplete/)	triggered after the export process has finished
exportstart (/api/exportstart/)	triggered when the export of grid or chart starts
fieldslistclose (/api/fieldslistclose/)	triggered when the built-in Field List is closed
fieldslistopen (/api/fieldslistopen/)	triggered when the built-in Field List is opened
filterclose (/api/filterclose/)	triggered when the filter pop-up window is closed
filteropen (/api/filteropen/)	triggered when the filter pop-up window is opened
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
unauthorizederror (/api/unauthorizederror/)	triggered when the Accelerator sends the 401 Unauthorized error in response to the Flexmonster request
update (/api/update/)	triggered when the change occurred in the component

1.2. Flexmonster()

new Flexmonster({
container: String,
componentFolder: String,
global: Report Object (/api/report-object/),
width: Number,
height: Number,
report: Report Object (/api/report-object/) | String,
toolbar: Boolean,
customizeCell: Function,
customizeChartElement: Function,
customizeContextMenu: Function,
customizeAPIRequest: Function,
licenseKey: String
})

[starting from version: 2.4]

Embeds the component into the HTML page. This method allows you to insert the component into your HTML page and provide it with all the 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 the global object.

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

Parameters

- container – String. The selector of the HTML element you would like to have as a container for the component.
- componentFolder – String. 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. *Default value: flexmonster/*.
- global – Object. It allows you to preset options for all reports. These options can be overwritten for specific reports. The object structure is the same as for the Report Object (/api/report-object/).
- width – Number. Width of the component on the page (pixels or percent). *Default value: 100%*.
- height – Number. Weight of the component on the page (pixels or percent). *Default value: 500*.
- report – Report Object | String. A property to set a report. It can be inline Report Object (/api/report-object/) or URL to report JSON.
- toolbar – Boolean. It indicates whether the toolbar is embedded or not. *Default value: false (without the Toolbar)*.
- customizeCell – Function. Allows customizing of separate cells. Have a look at customizeCell definition and examples (/api/customizecell/).
- customizeChartElement – Function. Allows customizing separate chart elements in Flexmonster Charts. Have a look at the customizeChartElement definition and examples (/api/customizeChartElement/).
- customizeContextMenu – Function. Allows customizing the context menu. Have a look at customizeContextMenu definition and examples (/api/customizecontextmenu/).
- customizeAPIRequest – Function. Allows adding custom properties to the request body before the request is sent to a server. Have a look at customizeAPIRequest definition and examples (/api/customizeapirequest).
- licenseKey – String. The license key.

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

The only required property is container. If you run new Flexmonster({container: "pivotContainer"}), where "pivotContainer" is the selector of container HTML element, – an empty component without the toolbar will be added with the default width and height.

Returns

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

To get a reference to the component after initialization, use the uielement property of the component's container:

```
const flexmonsterAPI = document.getElementById("flexmonsterContainer")
                       .uielement.flexmonster;
```

Flexmonster's methods are available through this reference as well:

```
flexmonsterAPI.getReport();
```

Examples

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

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

<script>
 var pivot = new Flexmonster({
  container: "pivotContainer",
  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 the Toolbar:

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

<script>
 var pivot = new Flexmonster({
  container: "pivotContainer",
  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) Add and operate multiple instances:

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

<script>
 var pivot1 = new Flexmonster({
  container: "pivotContainer",
  toolbar: true,
  report: {
   dataSource: {
    filename: "data.csv"
   }
  },
  licenseKey: "XXXX-XXXX-XXXX-XXXX-XXXX"
 });

 var pivot2 = new Flexmonster({
  container: "pivotContainer",
  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/).

4) Set event handler via Flexmonster():

```
var pivot = new Flexmonster ({
 container: "pivotContainer",
 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/).

5) How to use customizeCell:

```
new Flexmonster({
   customizeCell: function(cell, data) {
  // change cell
 },
 ...
});
```

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

See also

list of events (/api/events/)

1.3. $.flexmonster

$.flexmonster() method was deprecated in version 2.4. You should use the new Flexmonster() (/api/new-flexmonster/) instead.

$("#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. *Default value: flexmonster/*.
- global – an object that allows you to preset options for all reports. These options can be overwritten for concrete reports. The object structure is the same as for Report Object (/api/report-object/).
- width – width of the component on the page (pixels or percent). *Default value: 100%*.
- height – height of the component on the page (pixels or percent). *Default value: 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 – the parameter to embed the Toolbar or not. *Default: false (without the Toolbar)*.
- customizeCell – the function that allows the 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>
```

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>
```

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>
```

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");
 }
});
```

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/q1gtwj48/5/).

See also

list of events (/api/events/)

2. Objects

2.1. All 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
Data Source Object (/api/data-source-object/)	contains information about the data source
Mapping Object (/api/mapping-object/)	contains information about field data types, captions, multi-level hierarchies, and other view configurations of the data source
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
Filter Object (/api/filter-object/)	contains filtering information
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
Table Sizes Object (/api/table-sizes-object/)	contains information about table sizes
Cell Data Object (/api/cell-object/)	contains information about the cell
Chart Data Object (/api/chart-data-object/)	contains information about the chart segment

Toolbar Object (/api/toolbar-object/)	contains information about the Toolbar
Chart Legend Data Object (/api/chart-legend-data-object/)	contains information about the chart legend element

2.2. 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 – Data Source Object (/api/data-source-object/). Contains information about the data source.
- 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 the 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 – Table Sizes Object (/api/table-sizes-object/). Contains information about table sizes.
- customFields – Array of objects. Allows setting custom fields for Excel export or storing some additional information. They are not shown on the grid but they will be displayed in the exported Excel table. Each object has the following properties:
    - name – String. The custom field's name.
    - value – String. The custom field's value. This live sample on JSFiddle (https://jsfiddle.net/flexmonster/petcnwsx/) illustrates how to set the customFields property.
- localization – String | Object. A property to set a localization. It can be inline JSON or URL to a localization JSON file.
- version – String. It contains the current version of Flexmonster.
- creationDate – String. Represents the date (in ISO format) of the report creation.

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

To learn more about the usage of Report Object, please refer to What is a report (/doc/configuring-report/) tutorial.

Example

Example of report object:

```
{
    "dataSource": {
        "type": "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,
        "showTotals": "on",
        "showGrandTotals": "on",
        "showExtraTotalLabels": false,
        "showHierarchies": true,
        "showHierarchyCaptions": true,
        "showReportFiltersArea": true
    },
    "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": false,
    "configuratorButton": true,
    "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",
    "showOutdatedDataAlert": false
},
"conditions": [
    {
        "formula": "#value < 40",
        "format": {
            "backgroundColor": "#FFCCFF",
            "color": "#000033",
```

```
                "fontFamily": "Arial",
                "fontSize": "12px"
            }        }
    ],
    "formats": [
        {
            "name": "29mvnel3",
            "thousandsSeparator": " ",
            "decimalSeparator": ".",
            "decimalPlaces": -1,
            "maxDecimalPlaces": -1,
            "maxSymbols": 20,
            "currencySymbol": "$",
            "negativeCurrencyFormat": "-$1",
            "positiveCurrencyFormat": "$1",
            "nullValue": "",
            "infinityValue": "Infinity",
            "divideByZeroValue": "Infinity",
            "textAlign": "right",
            "isPercent": false
        }
    ],
    "tableSizes": {
        "columns": [
            {
                "tuple": [],
                "measure": "[Measures].[Reseller Order Count]",
                "width": 182
            }
        ]
    },
    "localization": "loc-ua.json",
    "version": "2.7.21",
    "creationDate": "2019-12-17T17:17:02.258Z"
}
```


2.3. Data Source Object

The data source is a required part of the report object. Flexmonster supports data from OLAP data sources, Elasticsearch, SQL databases, MongoDB databases, the custom data source API, CSV and JSON static files, and inline JSON data. Each data source requires specific properties to be set inside the dataSource section of the report object. Here is a list of all available properties for a dataSource:

- mapping (optional) – Mapping Object (/api/mapping-object/) | String. Allows defining field data types, captions, multilevel hierarchies, grouping fields under separate dimensions, and setting other view configurations of hierarchies. It can be either the inline Mapping Object (https://www.flexmonster.com/api/mapping-object/) or the URL to a JSON file with mapping. See an example on JSFiddle (https://jsfiddle.net/flexmonster/5oxfubmc/).
- binary (optional) – Boolean. Flag to use Data Speed Accelerator instead of XMLA protocol. Only for

"microsoft analysis services" and "mondrian" data source types.

- browseForFile (optional) – Boolean. Only for "csv" and "json" data source types. Defines whether you want to load a file from the local file system (true) or not (false). *Default value: false*.
- catalog (optional) – String. The data source catalog name of the OLAP data source. Only for "microsoft analysis services" and "mondrian" data source types.
- cube (optional) – String. The given catalog's cube's name of the OLAP data source. Only for "microsoft analysis services" and "mondrian" data source types.
- data (optional) – JSON. A property to set JSON data if it is already on the page.
- dataSourceInfo (optional) – String. The service info of the OLAP data source. Only for "microsoft analysis services" and "mondrian" data source types.
- type (optional) – String. Type of data source. The component supports the following types: "json", "csv", "api", "microsoft analysis services", "elasticsearch", and "mondrian".
- effectiveUserName (optional) – String. Use when an end-user identity must be impersonated on the server. Specify the account in a domain\user format. Only for "microsoft analysis services" data source type.
- fieldSeparator (optional) – String. Defines the specific fields separator to split each CSV row. There is no need to define it if the CSV fields are separated by , or ;. This property is required only if another character separates fields.
  For example, if you use TSV, where a tab character is used to separate fields, then the fieldSeparator parameter should be set to "\t".
  Alternatively, you can specify the field separator in the CSV file's first row using the sep prefix. Supported prefix formats are the following: sep=fieldSep and "sep=fieldSep". For example:

  ```
  sep=|
  Country|Color|Price
  Canada|blue|1000
  ```

  Only for the "csv" data source type.
- thousandSeparator (optional) – String. Allows importing CSV data with commas used to separate groups of digits in numbers (e.g., 1,000 for one thousand). To load such data, set thousandSeparator to ",".
- filename (optional) – String. The URL to CSV or JSON file or a server-side script that generates CSV data or JSON data. Only for "csv" and "json" data source type.
- ignoreQuotedLineBreaks (optional) – Boolean. Indicates whether the line breaks in quotes will be ignored (true) in CSV files or not (false). When set to true, CSV parsing is 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. *Default value: true*.
- localeIdentifier (optional) – Number. Microsoft Locale ID Value for your language. Only for "microsoft analysis services" data source type.
- proxyUrl (optional) – String. The path to proxy URL to the OLAP data source, such as Microsoft Analysis Services, Mondrian. Only for "microsoft analysis services" and "mondrian" data source types. In the case of Microsoft Analysis Services, both tabular and multidimensional model types are supported.
- url (optional) – String. The path to the API endpoints. Only for "api" data source type.
- roles (optional) – String. Comma-delimited list of predefined roles to connect to a server or database using permissions conveyed by that role. If this property is omitted, all roles are used, and the effective permissions are the combination of all roles. Supported only for "microsoft analysis services" and "mondrian" data source types.
- subquery (optional) – String | Object. The parameter to set the server-side filter, which helps decrease the size of the server's response. Only for "microsoft analysis services" and "elasticsearch" data source types. For "microsoft analysis services" data source type should be set as a string. 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]". For "elasticsearch" data source type should be set as a Bool Query Object (https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-bool-query.html).
- requestHeaders (optional) – Object. For all data sources. This object allows you to add custom request headers. This object consists of "key": "value" pairs, where "key" is a header name and "value" is its value.

Check out a live sample on JSFiddle (https://jsfiddle.net/flexmonster/7z40n1r8/). Important note: requestHeaders is not saved when obtaining the report via save() and getReport() API calls.
- singleEndpoint – Boolean. When set to true, all custom data source API requests are sent to a single endpoint specified in the url property. Only for the "api" data source type. *Default value: false*.
- node (optional) – String. The URL string for the connection. Only for "elasticsearch" data source type.
- index (optional) – String. The name of the Elasticsearch index to connect or the dataset identifier for the custom data source API. Only for "elasticsearch" and "api" data source types.
- useStreamLoader (optional) – Boolean. Optimizes the large file processing using the stream loader. When set to true, the stream loader is enabled. Available only when loading JSON files via URL. See an example on JSFiddle (https://jsfiddle.net/flexmonster/qd6h4tsv/). Default value: false.
- withCredentials (optional) – Boolean. It indicates whether cross-site Access-Control requests should be made using credentials such as authorization headers (true) or not (false). For more details, refer to MDN web docs (https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/withCredentials). Setting the withCredentials flag to true is recommended when using Windows authentication and other types of server authentications. When set to false, the browser does not request credentials and does not include them into outgoing requests. Not supported for the "mondrian" data source type. *Default value: false.*

The API calls connectTo() (/api/connectto/), updateData() (/api/updatedata/), load() (/api/load/), and open() (/api/open/) are used to change the data source at runtime. The API call save() (/api/save/) is used to save the report.

Find more details about Data Source Object in this tutorial with examples (/doc/data-source/).

2.4. Mapping Object

The Mapping Object allows defining field data types, captions, and multilevel hierarchies; grouping fields under separate dimensions, and setting other view configurations of hierarchies.

The Mapping Object is available for all data sources. It presents a powerful way to neatly separate a data source from its presentation. Find more details about the Mapping Object in this tutorial with examples (https://www.flexmonster.com/doc/mapping/).

For each field in the data source, you can set the following properties:

- caption (optional) – String. The hierarchy's caption.
- type (optional) – String. The field's data type. Only for "json", "csv", and "api" data source types. type can be:
    - "string" – The field stores string data.
    - "number" – The field stores numerical data. It can be aggregated with all available aggregations.
    - "month" – The field stores months. Note that if the field stores month names only (in either short or full form), the field will be recognized by Flexmonster as a field of the "month" type automatically. If the field contains custom month names, specify its type as "month" explicitly.
    - "weekday" – The field stores days of the week.
    - "date" – The field stores a date. Fields of this type are split into 3 different fields: Year, Month, Day. Only for "json" and "csv" data source types.
    - "year/month/day" – The field stores a date. It's displayed as a multilevel hierarchy with the following levels: Year > Month > Day. Only for "json" and "csv" data source types.
    - "year/quarter/month/day" – The field is a date. It's displayed as a multilevel hierarchy with the following levels: Year > Quarter > Month > Day. Only for "json" and "csv" data source types.
    - "date string" – The field stores a date. Fields of this type are represented as strings and can be used in rows, columns, or report filters. The component sorts members of such a field as dates. Fields of the "date string" type can be formatted using the datePattern (https://www.flexmonster.com/api/options-object/#datePattern) option (the default pattern is

"dd/MM/yyyy").
    - "datetime" – The field stores a date. You can select fields of this type for values and apply min, max, count, and distinctcount aggregations to them.
      Fields of the "datetime" type can be formatted using the dateTimePattern (https://www.flexmonster.com/api/options-object/#dateTimePattern) option (the default pattern is "dd/MM/yyyy HH:mm:ss").
    - "time" – The field stores time. Fields of this type can be formatted using the timePattern (https://www.flexmonster.com/api/options-object/#timePattern) option (the default pattern is "HH:mm:ss").
    - "id" – The field is an id. This field is not shown in the Field List. Fields of this type can be used for editing data.
    - "property" – The field for setting member properties. This field is not shown in the Field List. For example, it can be used to associate a productID with a product. See an example here (https://jsfiddle.net/flexmonster/nm09d7zh/).
- hierarchy (optional) – String. The hierarchy's name. When configuring hierarchies, specify this property to mark the field as a level of a hierarchy or as a member property of a hierarchy (in this case, the type parameter should be set to "property"). Only for "json", "csv", and "api" data source types.
- parent (optional) – String. The unique name of the parent level. This property is necessary if the field is a level of a hierarchy and has a parent level. Only for "json", "csv", and "api" data source types.
- folder (optional) – String. The field's folder. Folders are used to group several fields in the Field List. folder supports nesting via / (e.g., "Folder/Subfolder/"). Only for "json", "csv", and "api" data source types.
- aggregations (optional) — Array of strings. This property represents the list of aggregation functions that can be applied to the current measure.
- filters (optional) – Boolean. This property allows enabling and disabling the UI filters for a specific hierarchy. When set to false, the UI filters are disabled. *Default value: true*.
- visible (optional) – Boolean. When set to false, hides the field from the Field List. Only for "elasticsearch", "csv", and "api" data source types.
- interval (optional) – String. Allows aggregating dates by the given interval. The interval property can be used in the following ways:
    - For the date histogram (https://www.elastic.co/guide/en/elasticsearch/reference/current/search-aggregations-bucket-datehistogram-aggregation.html) in Elasticsearch. Check out supported time units (https://www.elastic.co/guide/en/elasticsearch/reference/current/common-options.html#date-math). Only for the "elasticsearch" data source type.
    - For the "date string" and "datetime" field types. Supported date intervals are the following: "d" for days, "h" for hours, "m" for minutes, and "s" for seconds (e.g., "1d", "7h", "20m", "30s"). Note that grouping by days starts from 1 January 1970. Only for "csv" and "json" data source types.
- isMeasure (optional) – Boolean. When set to true, the field can be selected only as a measure. The isMeasure property should be used only with the strictDataTypes option (https://www.flexmonster.com/api/options-object/#strictDataTypes) (see an example on JSFiddle (https://jsfiddle.net/flexmonster/b9xy0j5u/)). Only for the "json" data source type. *Default value: false*.
- time_zone (optional) – String. Used for the date histogram (https://www.elastic.co/guide/en/elasticsearch/reference/current/search-aggregations-bucket-datehistogram-aggregation.html). You can specify time zones as either an ISO 8601 UTC offset (e.g., +01:00 or -08:00) or as a time zone ID as specified in the IANA time zone database, such as America/Los_Angeles. Only for the "elasticsearch" data source type. Check out an example here (https://www.elastic.co/guide/en/elasticsearch/reference/current/search-aggregations-bucket-datehistogram-aggregation.html#_time_zone_2).
- format (optional) – String. Used to format different types of date fields. format can be used in the following ways:
    - For the date histogram (https://www.elastic.co/guide/en/elasticsearch/reference/current/search-aggregations-bucket-datehistogram-aggregation.html) in Elasticsearch. The date format/pattern is described in the Elasticsearch documentation (https://www.elastic.co/guide/en/elasticsearch/reference/current/search-aggregations-bucket-daterange-aggregation.html#date-format-pattern). Try a live sample on JSFiddle (https://jsfiddle.net/flexmonster/uvwnrzL0/).
      If the datePattern (/api/options-object/#datePattern) option is defined, format will override its value for the field.

Only for the "elasticsearch" data source type.
    - For a field of the "date string" type, it allows overriding the datePattern (https://www.flexmonster.com/api/options-object/#datePattern) set in the Options Object. The pattern string for the format is the same as in the datePattern option (i.e., "dd/MM/yyyy"). Only for "json", "csv", and "api" data source types. Learn more about date and time formatting (https://www.flexmonster.com/doc/date-and-time-formatting/).
    - For a field of the "datetime" type, it allows overriding the dateTimePattern (https://www.flexmonster.com/api/options-object/#dateTimePattern) set in the Options Object. The pattern string for the format is the same as in the dateTimePattern option (i.e., "dd/MM/yyyy HH:mm:ss"). Only for "json", "csv", and "api" data source types. Learn more about date and time formatting (https://www.flexmonster.com/doc/date-and-time-formatting/).
    - For a field of the "time" type, it allows overriding the timePattern (https://www.flexmonster.com/api/options-object/#timePattern) set in the Options Object. The pattern string for the format is the same as in the timePattern option (i.e., "HH:mm:ss"). Only for "json", "csv", and "api" data source types. Learn more about date and time formatting (https://www.flexmonster.com/doc/date-and-time-formatting/).
- min_doc_count (optional) – Number. Only for the "elasticsearch" data source type. Used for the date histogram (https://www.elastic.co/guide/en/elasticsearch/reference/current/search-aggregations-bucket-datehistogram-aggregation.html). Can be used to show intervals with empty values (min_doc_count: 0). *Default value: 1 (empty intervals are hidden)*.

2.5. 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. A hierarchy unique name.
    - caption (optional) – String. A hierarchy caption.
    - dimensionName (optional) – String. A dimension name.
    - filter (optional) – Filter Object (/api/filter-object/). It contains filtering information.
    - levelName (optional) – String. It used to specify the level of the hierarchy that is shown on the grid.
    - showTotals (optional) — Boolean. Defines whether totals are shown or hidden for the hierarchy. When set to true, totals are shown. Only for the classic view.
      If showTotals is not specified, totals' visibility is defined by the options.showTotals property. Learn more about options.showTotals (https://www.flexmonster.com/api/options-object/#showTotals).
      To show and hide totals via UI, use a context menu. Open it by right-clicking the hierarchy header.
    - sort (optional) – String. A 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.]. Only for "csv" and "json" data source types.
- drills (optional) – Object. Stores the information about drill-downs in multilevel hierarchies:
    - drillAll (optional) – Boolean. It 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.
- drillThrough (optional) – Array. It allows pre-defining slice for the drill-through view. Only for "csv", "json", and "api" data source types. Can be specified the following way: ["Hierarchy name 1", "Hierarchy name 2", etc.] (see live demo (https://jsfiddle.net/flexmonster/ra6cbgoq/)).
- 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.
- flatSort – Array of objects. Only for "json", "csv", and "api" data source types. It contains the list of objects defining the multicolumn sorting on the flat grid. Each object has the following properties:
    - uniqueName – String. The unique name of the hierarchy being sorted.
    - sort – String. The sorting type ("asc", "desc", or "undefined").

  Note: the hierarchies are sorted in the order they were specified (i.e., the first hierarchy is sorted first, and so on). Therefore, take the element's order into account when defining the flat table multicolumn sorting.
  See theexample on JSFiddle (https://jsfiddle.net/flexmonster/3u1o2mry/).
  To perform multicolumn sorting via UI, use Ctrl+click.
- 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. A measure unique name.
    - active (optional) – Boolean. A 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. A unique name of aggregation that will be applied to the measure. To see the list of supported aggregation functions for each data source type, refer to Flexmonster's technical specifications (https://www.flexmonster.com/technical-specifications/#aggregations). If it is a calculated measure, it will be "none".
    - availableAggregations (optional) — Array of strings. Note that starting from version 2.8, the availableAggregations property is considered deprecated. Use the Mapping object's (/doc/mapping/#aggregations) aggregations property instead.
      availableAggregations represents the list of aggregation functions which can be applied to the current measure. If it is a calculated measure, it will be [].
    - caption (optional) – String. A measure caption.
    - formula (optional) – String. Represents the formula. Refers to the calculated measure (https://www.flexmonster.com/doc/calculated-values/). It can contain the following operators: +, -, *, / (check a full list (/doc/calculated-values/#!formula-operators)). Other measures can be referenced using the measure's unique name and the associated aggregation function, for example sum("Price") or max("Order"). To see the list of supported aggregation functions for each data source type, refer to Flexmonster's technical specifications (/technical-specifications/#aggregations).
    - individual (optional) – Boolean. It refers to the calculated measure. Defines whether the formula is calculated using raw values (true) or using aggregated values (false). Only for "json" and "csv" data source types. *Default value: false*.
    - calculateNaN (optional) – Boolean. Defines whether the formula is calculated using the NaN values (true) or using the null values (false). *Default value: true*.
    - format (optional) – String. A name of number formatting.
    - grandTotalCaption (optional) – String. A measure grand total caption.
- flatOrder – Array of strings. Defines the order of the hierarchies for the "flat" grid type. flatOrder can be specified like this: ["Hierarchy name 1", "Hierarchy name 2", etc.] (see live demo (https://jsfiddle.net/flexmonster/xj4py32w/)). Only for "json", "csv", and "api" data source types.
- memberProperties – Array of objects. Only for "microsoft analysis services" and "mondrian" data source types. Each object in the array has the following properties:
    - levelName – String. A hierarchy unique name.
    - properties – Array. It contains member properties, which will be shown on the grid. Other available member properties can be accessed through the context menu.
- reportFilters – Array of objects. A list of hierarchies selected in the report slice for Report Filters. Each object has the following properties:
    - uniqueName – String. A hierarchy unique name.
    - caption (optional) – String. A hierarchy caption.
    - dimensionName (optional) – String. A dimension name.
    - filter (optional) – Filter Object (/api/filter-object/). It contains filtering information.
    - levelName (optional) – String. Used to specify the level of the hierarchy that is shown on the grid.

    - sort (optional) – String. A sorting type for the members ("asc", "desc" or "unsorted").
    - sortOrder (optional) – Array. Using this property you can set custom sorting for hierarchy members. Only for "csv" and "json" data source types. 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. A hierarchy unique name.
    - caption (optional) – String. A hierarchy caption.
    - dimensionName (optional) – String. A dimension name.
    - filter (optional) – Filter Object (/api/filter-object/). It contains filtering information.
    - levelName (optional) – String. Used to specify the level of the hierarchy that is shown on the grid.
    - showTotals (optional) — Boolean. Defines whether totals are shown or hidden for the hierarchy. When set to true, totals are shown. Only for the classic view.
      If showTotals is not specified, totals' visibility is defined by the options.showTotals property. Learn more about options.showTotals (https://www.flexmonster.com/api/options-object/#showTotals).
      To show and hide totals via UI, use a context menu. Open it by right-clicking the hierarchy header.
    - sort (optional) – String. A sorting type for the members ("asc", "desc" or "unsorted").
    - sortOrder (optional) – Array. Using this property you can set custom sorting for hierarchy members. Only for "csv" and "json" data source types. 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 – Object. Identifies the measure on which sorting will be applied. Has the following properties:
            - uniqueName – String. A measure's unique name.
            - aggregation (optional) – String. A measure aggregation type.
        - type – String. A 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 identify the row in the table based on data in it.
        - measure – Object. Identifies the measure on which sorting will be applied. Has the following properties:
            - uniqueName – String. A measure unique name.
            - aggregation (optional) – String. A measure aggregation type.
        - type – String. A sorting type ("asc" or "desc").

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

Find more details about Slice Object in this tutorial with examples (/doc/slice/).

2.6. Options Object

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

- viewType – String. The type of view to show: "grid" or "charts" or "grid_charts" (starting from v1.9). *Default value: "grid"* .
- grid – Object. Contains information about grid:
    - type – String. The selected grid's type. The following grid types are supported: "compact", "classic", and "flat". *Default value: "compact"*.

    - title – String. The title of the grid. *Default value: ""*.
    - showFilter – Boolean. Indicates whether column/row filter controls and report filter controls are visible on the grid (true) or not (false). *Default value: true*.
    - showHeaders – Boolean. Indicates whether the spreadsheet headers are visible (true) or not (false). *Default value: true*.
    - showTotals – String. Specifies where to show totals: in rows ("rows"), in columns ("columns"), in rows and columns ("on") or not at all ("off"). *Default value: "on"*.
    - showGrandTotals – String. Specifies where to show grand totals: in rows ("rows"), in columns ("columns"), in rows and columns ("on") or not at all ("off"). *Default value: "on"*.
    - grandTotalsPosition – String. Indicates whether the grand totals are displayed at the top of the table ("top") or at the bottom ("bottom"). Only available for the flat view. *Default value: "top"*.
    - showHierarchies – Boolean. Specifies how to show drillable hierarchy cells on the grid: with a link on the right (true) or with an icon on the left (false). *Default value: true*.
    - showHierarchyCaptions – Boolean. Indicates whether the hierarchy captions are visible (true) on the grid or not (false). *Default value: true*.
    - drillthroughMaxRows – Number. Sets the maximum number of rows for the SSAS drill-through pop-up window. *Default value: 1000*.
    - showReportFiltersArea (starting from v2.2) – Boolean. Indicates whether the reports filtering cells on the grid should be visible (true) or not (false). *Default value: true*.
    - dragging – Boolean. Indicates whether the hierarchies on the grid can be dragged (true) or not (false). *Default value: true*.
    - showAutoCalculationBar – Boolean. Indicates whether the autoCalculationBar feature (/user-interface/#autoCalculationBar) is turned on (true) or not (false). *Default value: true*.
- chart – Object. Contains information about charts:
    - type – String. The selected chart type. The following chart types are supported: "column", "bar_h", "line", "scatter", "pie", "stacked_column", and "column_line". *Default value: "column"*.
    - title – String. The title of the chart. *Default value: ""*.
    - showFilter – Boolean. Indicates whether column/row filter controls and report filter controls are visible on the charts (true) or not (false). *Default value: true*.
    - multipleMeasures – Boolean. Available from v1.9. Indicates whether to show multiple measures on charts. *Default value: false*.
    - oneLevel – Boolean. In 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 the legend (true). *Default value: false*.
    - autoRange – Boolean. Indicates whether the range of values in the charts is selected automatically or not. *Default value: false*.
    - reversedAxes – Boolean. Reverses the columns and rows when drawing charts. *Default value: false*.
    - showLegend – Boolean. Indicates whether the legend for the charts is visible (true) or not (false). *Default value: true*.
    - showLegendButton – Boolean. Indicates whether the button to show/hide the legend for the charts is visible. When set to false, the legend is visible, without a button to hide it. *Default value: false*.
    - showDataLabels – Boolean. Setting this value to true allows showing labels in charts. If the value is false, the labels will be hidden. Use showAllLabels to configure the labels in a pie chart. *Default value: false*.
    - showAllLabels – Boolean. Setting this value to true allows showing all the labels in a pie chart. If the value is false then only the labels that can be shown without overlapping will be shown. *Default value: false*.
    - showMeasures – Boolean. Hides all dropdowns on the tops of the charts. This is useful if you want to show a simple chart without any controls or if you want to save space. When set to true, the dropdowns are visible. *Default value: true*.
    - showOneMeasureSelection – Boolean. When set to true, the measures dropdown is always visible – regardless of the number 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. *Default value: true*.

    - showWarning – Boolean. Indicates whether warnings are shown if the data is too big for charts. *Default value: true*.
    - position – String. The positions of the charts in relation to the grid. It can be "bottom", "top", "left", or "right". *Default value: "bottom"*.
    - activeMeasure – Object. Identifies the selected measure in the charts view. Has the following properties:
        - uniqueName – String. The unique measure name. This measure must be present in the slice.measures (/api/slice-object/#measures) property.
        - aggregation – String. The measure aggregation type. This aggregation must be defined for the measure in the slice.measures.aggregation (/api/slice-object/#aggregation) property.
    - pieDataIndex – String. The selected column member index. Learn more (/doc/flexmonster-pivot-charts/#pie-chart) about the pie chart structure.
      Indexes start from "0" (the first member of a column field). If pieDataIndex is set to a non-existing index, the default index will be used. *Default value: "0"*.
    - axisShortNumberFormat – Boolean. Indicates whether axes labels on charts are displayed in short number format like 10K, 10M, 10B, 10T (true) or not (false). *Default value: undefined (show short format if max value > 10M)*.
- filter – Object. Filtering options:
    - timezoneOffset – Number. The difference (in minutes) between UTC and user's local time zone. Used to specify time zone for the dates in the date query filter (https://www.flexmonster.com/api/date-query-object/). *Default value: user's local time*.
    - weekOffset – Number. Sets amount of days to be added to the start of the week (Sunday). Used to adjust the first day of the week in the filter's calendar. *Default value: 1 (Monday is the first day of the week)*.
    - dateFormat – String. Date pattern to format dates in filter's date inputs. Has two possible values: "dd/MM/yyyy" and "MM/dd/yyyy". *Default value: "dd/MM/yyyy"*.
    - liveSearch – Boolean. Indicates whether the search in the filter pop-up window is performed while the user types (true) or requires the Enter button to start searching (false). *Default value: true*.
- configuratorActive – Boolean. Indicates whether the Field List is opened (true) or closed (false) after the component is initialized. *Default value: false*.
- configuratorButton – Boolean. Indicates whether the Field List toggle button is visible (true) or not (false). *Default value: true*.
- showAggregations – Boolean. Indicates whether the aggregation selection control is visible (true) or not (false) for measures on the Field List. *Default value: true*.
- showCalculatedValuesButton – Boolean. Controls the visibility of "Add calculated value" in the Field List. *Default value: true*.
- showEmptyValues – Boolean | String. Specifies where to show members with empty values on the grid and charts: in rows ("rows"), in columns ("columns"), in rows and columns (true), or not at all (false). Only for "csv" and "json" data source types. *Default value: false*.
- grouping – Boolean. Indicates whether grouping is enabled. This feature allows customers to group chosen elements using a filter window. For example, if the customer has shops in different cities and wants to analyze sales information, it would be possible to combine several cities in one group by geography, by sales numbers, etc. Only available for "csv"and "json" data source types. *Default value: false*.
- editing – Boolean. Indicates whether the editing feature is enabled (true) or disabled (false) on the drill-through pop-up window for CSV and JSON data sources. The user will be able to double-click the cell and enter a new value into it if the editing feature is enabled. *Default value: false*.
- drillThrough – Boolean. Indicates whether the drill-through feature is enabled (true) or disabled (false). The user can drill through by double-clicking the cell with a value. *Default value: true*.
- showDrillThroughConfigurator – Boolean. Indicates whether the Field List toggle button is visible in the drill-through view. Only for "csv", "json", "api","microsoft analysis services" (for Flexmonster Accelerator), and "elasticsearch" data source types. *Default value: true*.
- sorting – 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 at all ("off" or false). *Default value: "on"*.
- readOnly – Boolean. When set to true, enables the read-only mode for the component.

In the read-only mode, you cannot interact with the report via UI (e.g., the context menu, expands, and drills are disabled). In addition, configurations of the following options are ignored: showFilter (for both grid and charts), dragging, showMeasures, configuratorButton, sorting, and drillThrough.
We also suggest hiding the Toolbar when using the read-only mode.
Default value: false.

- strictDataTypes – Boolean. When set to true, the component treats fields marked as a measure only as measures and does not allow using them as hierarchies. This allows increasing data analyzing speed. To mark a field as a measure, set its mapping.isMeasure (https://www.flexmonster.com/api/mapping-object/#isMeasure) property to true. Check out an example (https://jsfiddle.net/flexmonster/b9xy0j5u/). Only for the "json" data source type. *Default value: false*.
- distinguishNullUndefinedEmpty – Boolean. Allows distinguishing null, undefined, and empty values. When set to false, the component treats null, undefined, and empty values as the same value. When set to true, there are three different values. *Default value: false*.
- defaultDateType – String. Used to specify which data types should be applied to date fields by default ("date", "date string", "year/month/day", "year/quarter/month/day" or "datetime"). Only for "json" and "csv" data source types. *Default value: "date"*.
- datePattern – String. Allows specifying how the component displays fields of the "date string" type. Only for "json", "csv", "api", and "elasticsearch" data source types. *Default pattern string: "dd/MM/yyyy"*. Learn more about date and time formatting (https://www.flexmonster.com/doc/date-and-time-formatting/).
- dateTimePattern – String. Allows specifying how the component displays fields of the "datetime" type. Only for "json", "csv", and "api" data source types. *Default pattern string: "dd/MM/yyyy HH:mm:ss"*. Learn more about date and time formatting (https://www.flexmonster.com/doc/date-and-time-formatting/).
- timePattern – String. Allows specifying how the component displays fields of the "time" type. Only for "json", "csv", and "api" data source types. *Default pattern string: "HH:mm:ss"*. Learn more about date and time formatting (https://www.flexmonster.com/doc/date-and-time-formatting/).
- dateTimezoneOffset – Number. It allows setting the time zone for hierarchical dates (for JSON: "date", "year/month/day" and "year/quarter/month/day"; for CSV: "d+", "D+" and "D4+"). See an example (https://jsfiddle.net/flexmonster/r8dk7v63/).
- saveAllFormats – Boolean. If there are more than 20 formats defined, only the formats that are used for "active: true" measures will be saved in the report. In order to save all the formats, set the saveAllFormats property to true. *Default value: false*.
- showDefaultSlice – Boolean. Defines whether the component selects a default slice for a report with an empty slice (when nothing is set in rows, columns, report filters, 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, set this property to false. Only available for "csv" and "json" data source types. *Default value: true*.
- useOlapFormatting – 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: false*.
- showMemberProperties – Boolean. Indicates whether the member properties for an OLAP data source are available in the component (true) or not (false). This feature is only for "microsoft analysis services" and "mondrian" data source types. *Default value: false*.
- showEmptyData – Boolean. It defines the component's behavior in case of an empty data source. Flexomontser handles the empty data in two ways:
    - when showEmptyData is set to true, the component will show an empty grid for an empty CSV/JSON file, a JSON file with an empty array, or an empty inline JSON array. For an empty CSV data source with the defined header, the component will show the slice with empty data cells.
    - when showEmptyData is set to false, the component will show a message "Data source is empty" and trigger a dataerror event for all mentioned types of the empty data source. See an example on JSFiddle (https://jsfiddle.net/flexmonster/4ku7Lscz/).

  Default value: true.
- defaultHierarchySortName – String. The sorting type for hierarchies' members ("asc", "desc", or "unsorted"). *Default value: "asc"*.
- showOutdatedDataAlert – Boolean. Setting this value to true will show a warning to the user before automatic reloading of data from the cube. When set to false, there is no warnings. Only for Flexmonster Accelerator. *Default value: false*.

- expandExecutionTimeout – Number. Allows specifying an execution timeout for the expandAllData() (https://www.flexmonster.com/api/expandalldata/) function.
  The timeout is set in milliseconds (i.e., 9000 equals 9 seconds). The minimum timeout value is 9000. Note that a large execution timeout can cause an unresponsive script alert.
  Default value: 9000.
- showAggregationLabels – Boolean. Indicates whether aggregation labels like "Total Sum of", "Sum of", etc. are shown in the column/row titles. *Default value: true*.
- showAllFieldsDrillThrough – Boolean. Indicates whether prefiltering the drill-through view columns is enabled (false) or the drill-through view displays all the available columns (true) when using SSAS with Flexmonster Accelerator. *Default value: false*.
- liveFiltering – Boolean. Indicates whether the live filtering for hierarchies' members is enabled (true) or disabled (false). *Default value: false*.
- showFieldListSearch – Boolean. Indicates whether the search bar in the Field List is shown (true) or hidden (false). When the search bar is hidden, it will be shown only in case the number of the hierarchies exceeds 50 (40 for the flat form). *Default value: false*.
- useCaptionsInCalculatedValueEditor – Boolean. By default, Flexmonster uses field unique names in the calculated value editor. If useCaptionsInCalculatedValueEditor is set to true, the component will use field captions instead of unique names. *Default value: false*.
- validateFormulas – Boolean. Indicates whether the validation is performed for calculated values (true) or not (false). In case the validation is turned on and the report contains invalid formula, the "Wrong formula format" alert message is shown. To turn off the "Wrong formula format" alert message, set the validateFormulas property to false. *Default value: true*.
- caseSensitiveMembers – Boolean. Indicates whether the hierarchies' members are case-sensitive (true) or not (false). *Default value: false*.
- simplifyFieldListFolders – Boolean. Indicates whether the folders containing one field should show this field in the root (true) or not (false). Only for Elasticsearch and SSAS data sources. *Default value: false*.
- validateReportFiles – Boolean. Indicates whether validation of report files is turned on (true) or turned off (false). Setting this value to false allows loading report files in the old format without an error message. Should be used in global options. *Default value: true*.
- fieldListPosition – String. Indicates whether the Field List is always shown on the right ("right") or in the pop-up window (undefined). *Default value: undefined*.
- allowBrowsersCache – Boolean. Indicates whether browsers are allowed to cache the data (true) or not (false). When allowBrowsersCache is set to false, caching is not allowed, and Flexmonster appends a unique id to the data source's URL to ensure that the URL stays unique. The unique URL prevents the browser from loading the data from its cache, so the data is updated every time it is requested. Setting allowBrowsersCache to true allows caching, and the URL remains unmodified (check out an example (https://jsfiddle.net/flexmonster/1c73vtup/)). *Default value: false.*

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

More information about options is available in article about Options Object (/doc/options/).

2.7. Filtering

2.7.1. Filter Object

A filter is an object that contains filtering information. Here is a list of all available properties for a filter:

- query (optional) – Query Object. Conditional filter to apply to the hierarchy. Available properties for Query Object depend on the hierarchy type:
  - Number Query Object (/api/number-query-object/)

    - String Query Object (/api/string-query-object/)
    - Date Query Object (/api/date-query-object/)
    - Time Query Object (/api/time-query-object/)

  Value Query Object (/api/value-query-object/) describes the filter on values.
- members (optional) – Array of strings. The hierarchy's members to be reflected/shown. Must not be used together with query. Example: { "members": ["country.[united states]"] }.
- exclude (optional) – Array of strings. The hierarchy's members to be excluded/hidden. Can be used together with query. Example: { "exclude": ["country.[united states]"] }.
- include (optional) – Array of strings. The hierarchy's members to be reflected/shown in addition to the results returned by query. Can be used for Number/String/Date/Time Query Objects. Example: { "include": ["country.[united states]"] }.
- measure (optional) – Object. Refers to the filter on values. Identifies the measure on which the Value Query Object (/api/value-query-object/) will be applied. measure has two properties:
    - uniqueName – String. The measure's unique name.
    - aggregation (optional) – String. The measure's aggregation type.

2.7.2. Number Query Object

Conditional filter to apply to the number hierarchies. Here is a list of all available conditions:

- equal – Number. Equal to some value. Example: { "equal": 5 }.
- not_equal – Number. Not equal to some value. Example: { "not_equal": 5 }.
- greater – Number. Greater than some value. Example: { "greater": 5 }.
- greater_equal – Number. Greater than or equal to some value. Example: { "greater_equal": 5 }.
- less – Number. Less than some value. Example: { "less": 5 }.
- less_equal – Number. Less than or equal to some value. Example: { "less_equal": 5 }.
- between – Array of numbers. Between two values (including them). Example: { "between": [0, 5] }.
- not_between – Array of numbers. Not between two values (excluding them). Example: { "not_between": [0, 5] }.

2.7.3. String Query Object

Conditional filter to apply to the string hierarchies. Here is a list of all available conditions:

- equal – String. Equal to some value. Example: { "equal": "aaa" }.
- not_equal – String. Not equal to some value. Example: { "not_equal": "aaa" }.
- begin – String. Begins with some value. Example: { "begin": "aaa" }.
- not_begin – String. Does not begin with some value. Example: { "not_begin": "aaa" }.
- end – String. Ends with some value. Example: { "end": "aaa" }.
- not_end – String. Does not end with some value. Example: { "not_end": "aaa" }.
- contain – String. Contains some value. Example: { "contain": "aaa" }.
- not_contain – String. Does not contain some value. Example: { "not_contain": "aaa" }.
- greater – String. Greater than some value. Example: { "greater": "aaa" }.
- greater_equal – String. Greater than or equal to some value. Example: { "greater_equal": "aaa" }.
- less – String. Less than some value. Example: { "less": "aaa" }.
- less_equal – String. Less than or equal to some value. Example: { "less_equal": "aaa" }.
- between – Array of strings. Between two values (including them). Example: { "between": ["aaa", "bbb"] }.
- not_between – Array of strings. Not between two values (excluding them). Example: { "not_between": ["aaa", "bbb"] }.

2.7.4. Date Query Object

Conditional filter to apply to the date hierarchies (except "date" type for CSV/JSON that creates separate hierarchies for years, months and days). Accepts dates in "YYYY-MM-DD" format. Here is a list of all available conditions:

- equal – String. Equal to some date. Example: { "equal": "2018-12-31" }.
- not_equal – String. Not equal to some date. Example: { "not_equal": "2018-12-31" }.
- before – String. Before some date. Example: { "before": "2018-12-31" }.
- before_equal – String. Before or equal to some date. Example: { "before_equal": "2018-12-31" }.
- after – String. After some date. Example: { "after": "2018-12-31" }.
- after_equal – String. After or equal to some date. Example: { "after_equal": "2018-12-31" }.
- between – Array of strings. Between two dates (including them). Example: { "between": ["2018-12-31", "2018-12-31"] }.
- not_between – Array of strings. Not between two dates (excluding them). Example: { "not_between": ["2018-12-31", "2018-12-31"] }.
- last – String. Filter dates of some previous period. Possible values: "day", "week", "month", "quarter", "year". Example: { "last": "week" }.
- current – String. Filter dates of some current period. Possible values: "day", "week", "month", "quarter", "year". Example: { "current": "week" }.
- next – String. Filter dates of some next period. Possible values: "day", "week", "month", "quarter", "year". Example: { "next": "week" }.

2.7.5. Time Query Object

Conditional filter to apply to the time hierarchies. Accepts time period in the following formats:

- "Xs" – X seconds. Example: "5s".
- "Xm" – X minutes. Example: "5m".
- "Xh" – X hours. Example: "5h".
- "Xd" – X days. Example: "5d".

Here is a list of all available conditions:

- equal – String. Equal to some value. Example: { "equal": "30m" }.
- not_equal – String. Not equal to some value. Example: { "not_equal": "30m" }.
- greater – String. Greater than some value. Example: { "greater": "30m" }.
- greater_equal – String. Greater than or equal to some value. Example: { "greater_equal": "30m" }.
- less – String. Less than some value. Example: { "less": "30m" }.
- less_equal – String. Less than or equal to some value. Example: { "less_equal": "30m" }.
- between – Array of strings. Between two values (including them). Example: { "between": ["30m", "1h"] }.
- not_between – Array of strings. Not between two values (excluding them). Example: { "not_between": ["30m", "1h"] }.

2.7.6. Value Query Object

Refers to the filter on values. Here is a list of all available conditions:

- top – Number. Top X values. Example: { "top": 5 }.
- bottom – Number. Bottom X values. Example: { "bottom": 5 }.
- equal – Number. Equal to some value. Example: { "equal": 5 }.
- not_equal – Number. Not equal to some value. Example: { "not_equal": 5 }.
- greater – Number. Greater than some value. Example: { "greater": 5 }.
- greater_equal – Number. Greater than or equal to some value. Example: { "greater_equal": 5 }.
- less – Number. Less than some value. Example: { "less": 5 }.

- less_equal – Number. Less than or equal to some value. Example: { "less_equal": 5 }.
- between – Array of numbers. Between two values (including them). Example: { "between": [0, 5] }.
- not_between – Array of numbers. Not between two values (excluding them). Example: { "not_between": [0, 5] }.

2.8. Format Object

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

- name – String. Should be unique as it identifies the format in the report. Note: the format with the name property set to "" defines a default number format and it is applied to all the measures without a specific number format. *Default value: ""*.
- thousandsSeparator – String. *Default value: " " (space)*.
- decimalSeparator – String. *Default value: "."*.
- decimalPlaces – Number. The exact number of decimals to show after the decimal separator. When set to -1, the entire number is shown. Note that for e-notation numbers (e.g., 5.8e+23), at least one decimal is always shown after the decimal separator, even if decimalPlaces is set to 0. *Default value: -1*.
- maxDecimalPlaces – Number. The maximum number of decimals to show after the decimal separator. When set to -1, the entire number is shown. Note that for e-notation numbers (e.g., 5.8e+23), at least one decimal is always shown after the decimal separator, even if maxDecimalPlaces is set to 0. *Default value: -1*.
- maxSymbols – Number. The maximum number of symbols in a cell. *Default value: 20*.
- negativeNumberFormat – String. The format of the negative numbers. It can be one of the following values: "-1", "- 1", "1-", "1 -", and "(1)". *Default value: "-1"*.
- currencySymbol – String. The symbol which is shown to the left or the right of the value (e.g. currency symbol, hours, percent, etc.). *Default value: ""*.
- positiveCurrencyFormat – String. The format of the currency symbol. It can be either "$1" or "1$". *Default value: "$1"*.
- negativeCurrencyFormat – String. The format of the currency symbol to display negative amounts. It can have the following values: "-$1", "-1$", "$-1", "$1-", "1-$", "1$-", "($1)", and "(1$)". *Default value: "-$1"*.
- isPercent – Boolean. When set to true, data is formatted as percentage. The behavior is the same as in Excel. Setting isPercent to true will result in numbers being multiplied by 100 and shown with a % symbol. For example, 0.56 gets changed to 56%. Note: if % is set as currencySymbol, setting isPercent to true will not multiply numbers by 100. *Default value: false*.
- nullValue – String. Defines how to show null values in the grid. *Default value: ""*.
- infinityValue – String. Defines how to show infinity values in the grid. *Default value: "Infinity"*.
- divideByZeroValue – String. Defines how to show divided by zero values in the grid. *Default value: "Infinity"*.
- textAlign – String. The alignment of formatted values in cells on the grid. It can have the following values: "right", "left", and "center". *Default value: "right"*.
- beautifyFloatingPoint – Boolean. When set to true, formats numbers like 1.560000000000001 as 1.56. Setting beautifyFloatingPoint to false means that the entire number will be shown. *Default value: true*.

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

Find more details in Number formatting tutorial (/doc/number-formatting/).

2.9. Conditional Format Object

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

- formula – String. A condition that can contain the following logical operators: AND, OR, ==, !=, >, <, >=,

<=, +, -, *, /, isNaN(), !isNaN().
#value is used as a reference to the cell value in the condition. Example: "#value > 2".
To refer to another field's value in the condition, use the field's name. Example: "'Price' > 2".

- format – Object. The style object that will be applied to a cell if the condition for the cell value is met. Note: when exporting to Excel and PDF, colors should be set to hex color codes.
- formatCSS (optional, read-only) – String. Represents a ready to use CSS string of the format style object. The format style object has properties with names that differ from CSS. The component transforms format to formatCSS.
- id (optional) – String. The id of the conditional formatting rule. If the id property is not set, the id for the rule will be set inside the pivot component.
- row (optional) – Number. The row index to which the condition should be applied.
- column (optional) – Number. The column index to which the condition should be applied.
- measure (optional) – String. The unique measure name to which the condition should be applied.
- hierarchy (optional) – String. The unique hierarchy name to which the condition should be applied. Must be used with the member property.
- member (optional) – String. The unique member name to which the condition should be applied. Must be used with the hierarchy property.
- isTotal (optional) – Boolean. If it is not defined, the condition will be applied to all cells. If it is set to true, the condition will be applied to total and subtotal cells only. If it is set to false, the condition will be applied to regular cells only.

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

To learn more about the usage of Conditional Format Object check our tutorial (/doc/conditional-formatting/).

2.10. Table Sizes Object

Table Sizes Object contains information about table sizes. It has the following properties:

- columns – Array of objects. Each object is used to save and restore the width of some column in the report:
  - width – Number. Column width in pixels.
  - idx – Number. Column's index, starts from 0. It is necessary to use either idx or tuple, not both.
  - tuple – Array. Consists of unique names that identify the column in the table based on data in it. It is necessary to use either idx or tuple, not both.
  - measure (optional) – Object. Identifies the measure. This property is defined only if "[Measures]" is selected for columns in the slice. The measure property is used with tuple and is not set when idx is used. Has the following properties:
    - uniqueName – String. The measure's unique name.
    - aggregation (optional) – String. The measure's aggregation type.
- rows – Array of objects. Each object is used to save and restore the height of some row in the report:
  - height – Number. Row height in pixels.
  - idx – Number. Row's index, starts from 0. It is necessary to use either idx or tuple, not both.
  - tuple – Array. Consists of unique names that identify the row in the table based on data in it. It is necessary to use either idx or tuple, not both.
  - measure (optional) – Object. Identifies the measure. This property is defined only if "[Measures]" is selected for rows in the slice. The measure property is used with tuple and is not set when idx is used. Has the following properties:
    - uniqueName – String. The measure's unique name.
    - aggregation (optional) – String. The measure's aggregation type.

To set table sizes for the component, use the setTableSizes() (/api/settablesizes/) method.

To get table sizes, use the getTableSizes() (/api/gettablesizes/) method.

2.11. Cell Data Object

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

- collapsed – Boolean. Indicates whether the cell is collapsed (true) or not (false). Only for cells with the "header" type.
- columnIndex – Number. Index of the cell column.
- columns – Array of objects. Cell column tuple. Each object contains information about column hierarchies. For JSON, SSAS and Mondrian can also contain member properties.
- escapedLabel – String. Cell label where certain characters, such as <, >, ", ' or /, are replaced by a hexadecimal escape sequence.
- expanded – Boolean. Indicates whether the cell is expanded (true) or not (false). Only for cells with the "header" type.
- drilledUp – Boolean. Indicates whether the cell is drilled up (true) or not (false). Only for cells with the "header" type.
- drilledDown – Boolean. Indicates the cell is drilled down (true) or not (false). Only for cells with the "header" type.
- height – Number. Cell height in px.
- hierarchy – Object. Hierarchy connected with the cell.
- isClassicTotalRow – Boolean. Indicates whether the cell contains the total value from the classic view.
- isDrillThrough – Boolean. Indicates whether the cell is from the grid (false) or from the drill through pop-up (true).
- isGrandTotal – Boolean. Indicates whether the cell contains the grand total value.
- isGrandTotalColumn – Boolean. Indicates whether the cell contains the column grand total value.
- isGrandTotalRow – Boolean. Indicates whether the cell contains the row grand total value.
- isTotal – Boolean. Identifies whether the cell contains the total value.
- isTotalColumn – Boolean. Indicates whether the cell contains the column total value.
- isTotalRow – Boolean. Indicates whether the cell contains the row total value.
- label – String. Cell label.
- level – Number. The level of the hierarchy.
- measure – Object. Measure connected with the cell.
- member – Object. Member connected with the cell.
- recordId – String | Array of strings. The property contains the values of the id field of records that were used to compose a cell. For the flat form, only one id is returned as a string. For the compact and classic forms, an array of such ids is returned. Only for CSV and JSON data sources.
- rowData – Array of Cell Data Objects (/api/cell-object/). The property is defined when the cell is double-clicked in the drill-through view. Contains the array of cells from the underlying data row. Only for CSV and JSON data sources.
- rowIndex – Number. Index of the cell row.
- rows – Array of objects. Cell row tuple. Each object contains information about row hierarchies. For JSON, SSAS and Mondrian can also contain member properties.
- type – String. Type of the cell. Can be "header" or "value".
- value – Number. Cell value.
- 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.

2.12. Chart Data Object

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

- chartType – String. The selected chart type. This property is returned only with the customizeChartElement (/api/customizechartelement/) API call.
- type – String. The element type. Can be either "series" or "series_label". This property is returned only with the customizeChartElement (/api/customizechartelement/) API call.
- columns – Array of objects. Column tuple. Each object contains information about the hierarchy members from columns associated with this chart segment.
- element – Object. The HTML DOM element of the segment.
- id – String. The id of the segment.
- label – String. The label of the segment.
- measure – Object. Measure associated with the segment.
- rows – Array of objects. Row tuple. Each object contains information about the hierarchy members from rows associated with this chart segment.
- value – Number. The value of the segment.

2.13. Toolbar Object

An object which contains information about the Toolbar. Use flexmonster.toolbar to get a reference to the Toolbar instance. It allows calling its functions on the page from outside of Flexmonster Pivot. To customize the Toolbar, refer to our customizing tutorial (/doc/customizing-toolbar/).

Example

Open conditional formatting dialog:

```
function openConditionalFormattingDialog() {
 flexmonster.toolbar.conditionalFormattingHandler();
}
```

Open on JSFiddle (https://jsfiddle.net/flexmonster/mpLh7sgw/).

See also

Customizing the Toolbar (/doc/customizing-toolbar/)

2.14. Chart Legend Data Object

Object which contains information about the chart legend element. It has the following properties:

- chartType – String. The selected chart type.
- type – String. The element type. For the Chart Legend Data Object it is "legend".
- label – String. The legend element’s label.
- color – String. The color of the series connected with the legend element.
- tuple – Array of objects. Each object contains information about the hierarchy members associated with this chart segment.
- member – Object. A member connected with the legend element.
- measure – Object. A measure connected with the legend element.
- level – Number. The nesting level of the legend element.
- isExpanded – Boolean. Indicates whether the legend element is expanded (true) or not (false).
- isCollapsed – Boolean. Indicates whether the legend element is collapsed (true) or not (false).
- isDrilledUp – Boolean. Indicates whether the legend element is drilled up (true) or not (false).

- isDrilledDown – Boolean. Indicates whether the legend element is drilled down (true) or not (false).

3. Methods

3.1. All methods

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

List of API calls:

addCalculatedMeasure (/api/addcalculatedmeasure/)	adds calculated measure
addCondition (/api/addcondition/)	adds a conditional formatting rule
alert (/api/alert/)	shows an alert pop-up window with a custom message
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 the Field 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
customizeAPIRequest (/api/customizeapirequest/)	allows customizing the request before it is sent to a server
customizeCell (/api/customizecell/)	allows customizing of separate cells
customizeChartElement (/api/customizechartelement/)	allows customizing separate chart elements in Flexmonster Charts
customizeContextMenu (/api/customizecontextmenu/)	allows customizing context menu
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
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
getFilter (/api/getfilter/)	returns the filtered members for the specified hierarchy
getFlatSort (/api/getflatsort/)	returns an array of objects defining the sorting on the flat grid
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
getReportFilters (/api/getreportfilters/)	returns a list of hierarchies selected in the report slice for Report Filters
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
getTableSizes (/api/gettablesizes/)	returns table sizes that are set for the component
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
openCalculatedValueEditor (/api/opencalculatedvalueeditor/)	opens the calculated value pop-up window editor
openFieldsList (/api/openfieldslist/)	opens the Field List
openFilter (/api/openfilter/)	opens the filter pop-up window for the specified hierarchy
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 rows, columns, measures and reportFilters from Slice Object (/api/slice-object/) and displays the result data
save (/api/save/)	saves your current report to a specified location
scrollToColumn (/api/scrolltocolumn/)	scrolls the grid to the specified column
scrollToRow (/api/scrolltorow/)	scrolls the grid to the specified row
setFilter (/api/setfilter/)	sets the filter for the specified hierarchy
setFlatSort (/api/setflatsort/)	sets the flat table multi-column sorting
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
setTableSizes (/api/settablesizes/)	sets table sizes for the component
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
sortingMethod (/api/sortingmethod/)	sets custom sorting for hierarchy members
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

3.2. addCalculatedMeasure

addCalculatedMeasure(measure: Object)

[starting from version: 2.3]

This API call adds a 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 the report is retrieved.

Note that you can add calculated measures only for reports based on "json", "csv", and "api" data source types.

Parameters

measure – the object that describes measure. This object can have the following parameters:

- uniqueName – String. The measure's unique name. This property will be used as an identifier for the measure inside Flexmonster and as an identifier to remove the measure via API.
- formula – String. Represents the formula. It can contain the following operators:+, -, *, / and the following functions:isNaN(), !isNaN() (check a full list (/doc/calculated-values/#!formula-operators)). Other measures can be addressed using the measure's unique name and aggregation function, for example sum("Price") or max("Order"). To see the list of supported aggregation functions for each data source type, refer to Flexmonster's technical specifications (/technical-specifications/#aggregations).
- caption (optional) – String. The measure's caption.
- grandTotalCaption (optional) – String. The measure's grand total caption.
- active (optional) – Boolean. Indicates whether the measure will be selected for the report (true) or not (false). active: false can be useful if the measure has non-default properties, but should not be selected for the grid or the chart.
- individual (optional) – Boolean. Only for "json" and "csv" data source types. Defines whether the formula is calculated using raw values (true) or using aggregated values (false). *Default value: false*.
- calculateNaN (optional) – Boolean. Defines whether the formula is calculated using NaN values (true) or using null values (false). *Default value: true*.
- format (optional) – String. The name of the number formatting that will be applied to the measure. Measure values can be formatted according to the number formatting defined in the report. All available number formattings are stored in the formats array in the report. More information about the number formatting part of the report can be found in the number formatting article.

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/)
Calculated values tutorial (/doc/calculated-values/)

3.3. 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 format to this cell:

```
var condition = {
  id: 1,
  formula: '#value > 400000',
  format: {fontSize : "10px", 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 format to the cell:

```
var condition = {
  id: 2,
  measure: "Sales",
  isTotal: true,
  formula: 'AND(#value > 100000000, #value < 200000000)',
  format: {fontSize : "11px", backgroundColor: "#00FF00"}
};
flexmonster.addCondition(condition);
flexmonster.refresh();
```

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

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

```
var condition = {
  id: 1,
  formula: 'isNaN(#value)',
  format: {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/)
Conditional formatting tutorial (/doc/conditional-formatting/)

3.4. addJSON

DEPRECATED

addJSON method was deprecated in version 2.4. You should use updateData (/api/api/updatedata/) instead.

3.5. addMeasure

[removed]

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

3.6. addStyleToMember

[removed]

addStyleToMember method was removed in version 2.3.

3.7. addUrlToMember

[removed]

addUrlToMember method was removed in version 2.3.

3.8. alert

alert(alertConfigs: Object)

[starting from version: 2.6.5]

This API call shows an alert pop-up window with a custom message.

Parameters

alertConfigs – the object that describes the alert pop-up. This object has the following parameters:

- title – String. The title of the pop-up. *Default value: ""*.
  Note: either title or message is required to show the alert pop-up.
- message – String. The message of the pop-up. *Default value: ""*.
  Note: either title or message is required to show the alert pop-up.
- type (optional) – String. The type of the pop-up. Available types: "alert", "confirmation", "error", "info". *Default type: "alert"*.
- buttons (optional) – Array of objects. By default, theOK button is shown. buttons is used to set custom buttons. Each object in buttons represents one button in the pop-up and has the following properties:
  - label – String. The label of the button.
  - handler – Function. The function that handles clicks on this button.
- blocking (optional) – Boolean. Defines whether the pop-up is blocking other actions until its button is clicked. *Default value: false*.

Example

Show the pop-up:

```
flexmonster.alert({
 title: "Error Title",
 message: "An error message",
 type: "error",
 buttons: [{
  label: "Button 1",
  handler: function() { console.log('Button 1 handler'); }
 }, {
  label: "Button 2",
  handler: function() { console.log('Button 2 handler'); }
 }],
```

```
 blocking: false

});
```

Open the example on JSFiddle (https://jsfiddle.net/flexmonster/a3t67sfr/).

See also

All events (/api/events/)

3.9. 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/).

3.10. clearFilter

clearFilter(hierarchyName: String)

[starting from version: 1.4]

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

Parameters

- hierarchyName – String. The name of the hierarchy.

Example

```
flexmonster.setFilter("Category",
  {
    "members": [
      "category.[bikes]",
      "category.[cars]"
    ]
  }
);
flexmonster.getFilter('Category');
```

```
/*
method getFilter() returns the following Object:
{
  "members": [
    "category.[bikes]",
    "category.[cars]"
  ]
}
*/

flexmonster.clearFilter('Category');

flexmonster.getFilter('Category');
/*
after clearFilter() call, method getFilter() returns null:
null
*/
```

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

See also

getFilter (/api/getfilter/)
setFilter (/api/setfilter/)

3.11. clearXMLACache

clearXMLACache(proxyURL: String, databaseId: String, callbackHandler: Function, 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 – String. The path to proxy URL to the Microsoft Analysis Services data source.
- databaseId – String. The ID of the database on the current connection.
- callbackHandler (optional) – Function. A 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 SSAS }. In case of failure the object will contain only one property: { complete: false }.
- cubeId (optional) – String. Cube ID.
- measuresGroupId (optional) – String. 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) – String. The name of a user account at the server. This parameter is necessary to complete the authentication process.
- password (optional) – String. The password to a user account at the server. This parameter is necessary

to complete the 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/).

3.12. closeFieldsList

closeFieldsList()

[starting from version: 1.4]

Closes the Field List.

Example

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

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

See also

openFieldsList (/api/openfieldslist/)

3.13. 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/)

3.14. collapseData

collapseData(hierarchyName: String)

[starting from version: 1.6]

Collapses all nodes of the specified hierarchy. Please note, this method works only for CSV and JSON data sources.

Example

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

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

See also

collapseAllData (/api/collapsealldata/)
expandAllData (/api/expandalldata/)
expandData (/api/expanddata/)

3.15. connectTo

connectTo(dataSource: Data Source Object (/api/data-source-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.

Parameters

Data Source Object (/api/data-source-object/) which contains connection parameters.

Examples

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

2) Connect to Microsoft Analysis Services:

```
flexmonster.connectTo({
```

```
  type: '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 Elasticsearch:

```
flexmonster.connectTo({
  type: 'elasticsearch',
  node: 'https://olap.flexmonster.com:9200',
  index: 'fm-product-sales'
});
```

4) Connect to CSV data source:

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

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

```
flexmonster.connectTo({
  type: 'csv',
  filename: 'colon-data.csv',
  fieldSeparator: ':'
});
```

6) Open local CSV file:

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

See also

updateData (/api/updatedata/)
open (/api/open/)
load (/api/load/)
save (/api/save/)
getReport (/api/getreport/)

setReport (/api/setreport/)

3.16. customizeAPIRequest

customizeAPIRequest(customizeAPIRequestFunction: Function)

[starting from version: 2.8]

This API call allows customizing the request before it is sent to a server. Only for "api" data source type.

customizeAPIRequest can be defined in two ways:

1. as a regular API call: flexmonster.customizeAPIRequest(customizeAPIRequestFunction);
2. as an initialization parameter: new Flexmonster({customizeAPIRequest: customizeAPIRequestFunction, ...})

Parameters

customizeAPIRequestFunction or null in case you do not need to change anything. The customizeAPIRequestFunction function has the following signature: customizeAPIRequestFunction(requestBody: Object): Object.

Data passed to customizeAPIRequestFunction:

- requestBody – Object. The object containing the request's body and through which the request can be customized. Has the following properties:
    - requestHeaders – Object. Allows you to add custom request headers. This object consists of "key": "value" pairs, where "key" is a header's name and "value" is its value. The requestHeaders object is either empty or containing the request headers added in the Data Source Object (https://www.flexmonster.com/api/data-source-object/) in the report. After customization, Flexmonster adds the contents of requestHeaders to the headers of the request. Changes made in requestBody are added to the request's body.

The customizeAPIRequestFunction function must return requestBody, the changed object with new or updated request headers. If requestBody is null, custom request headers will not be added to the request.

Example

Adding custom request headers:

```
function customizeAPIRequestFunction(req) {
    req.requestHeaders = {
        "customHeader1": "This is the first custom request header",
        "customHeader2": "This is the second custom request header"
    };
    console.log(req);
    return(req);
}
```

Check out a live sample on JSFiddle (http://jsfiddle.net/flexmonster/x6mh3c5k).

3.17. customizeCell

customizeCell(customizeCellFunction: Function)

[starting from version: 2.306]

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

Note that customizeCell can be defined in two ways:

1. as a regular API call: flexmonster.customizeCell(customizeCellFunction);
2. as an initialization parameter: new Flexmonster({customizeCell: customizeCellFunction, ...})

Parameters

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

- cell (starting from v2.4) – Cell builder. The object that contains the current representation of the cell on the grid and through which the cell representation can be customized. It has the following properties and methods:
  - attr – Object. All attributes and their values for the HTML element. Custom attributes can be added to the cell and, for example, used in CSS selectors to identify the cell. Read more info about CSS attribute selectors (https://css-tricks.com/attribute-selectors/).
  - classes – Array of strings. The array of classes assigned to the cell. The addClass() method should be used to add the new class.
  - style – Object. CSS object of the element that will be put in the style attribute of the element for inline styling.
  - tag – String. The tag of the element (each cell has tag: "div").
  - text – String. The text of the element which may also contain HTML, e.g., icons for expand, collapse, drill up and down, sorting, etc.
  - addClass(value: String) – Method. Use this method to add new classes to the element.
  - toHtml() – Method. Returns HTML string that represents the cell. It gathers all the properties of the cell builder object into HTML. This is how the cell will be added to the grid.
- data – Cell Data Object (/api/cell-object/) which contains information about the cell.

Examples

1) Alternating row colors:

```
function customizeCellFunction(cell, data) {
  if (data.type == "value") {
    if (data.rowIndex % 2 == 0) {
      cell.addClass("alter1");
    } else {
      cell.addClass("alter2");
    }
  }
}
```

"alter1" and "alter2" CSS classes are specified additionally. Check out a live sample on JSFiddle (https://jsfiddle.net/flexmonster/4z20ssq4/).

2) Styling subtotals and grand totals:

```
function customizeCellFunction(cell, data) {
  if (data.isClassicTotalRow) cell.addClass("fm-total-classic-r");
  if (data.isGrandTotalRow) cell.addClass("fm-grand-total-r");
  if (data.isGrandTotalColumn) cell.addClass("fm-grand-total-c");
}
```

This example has 3 different CSS classes and they are added to a subtotal row in classic view, to grand total row, and to a grand total column. Try on JSFiddle (https://jsfiddle.net/flexmonster/Lmx4p4os/).

3) Highlighting levels through the entire grid:

```
function customizeCellFunction(cell, data) {
  if (data.level != undefined) {
    cell.addClass("fm-level-"+data.level);
  }
}
```

customizeCellFunction is used here to add different CSS classes for different levels. This live sample on JSFiddle (https://jsfiddle.net/flexmonster/v06jx81y/) highlights the first three levels in rows.

4) Highlighting cells based on their semantics – member, hierarchy, measure:

- JSFiddle – Highlighting Country Canada on any place in rows or columns (https://jsfiddle.net/flexmonster/bbefo4z6/)
- JSFiddle – Highlighting rows with measure Price (https://jsfiddle.net/flexmonster/5mjp55rb/)
- JSFiddle – Highlighting cells based on its semantic (https://jsfiddle.net/flexmonster/1354qwh6/)

5) Styling rows based on conditional formatting:

Demonstrates highlighting of the entire row in the pivot table if the condition of the conditional formatting is true for at least one cell in this row. Also, this sample shows how to use beforegriddraw and aftergriddraw events. Check out on JSFiddle (https://jsfiddle.net/flexmonster/nonte3qv/).

6) Representing numbers by icons:

In this sample (https://jsfiddle.net/flexmonster/6shdmx9u/) cell values are replaced with the images depending on to which interval the value belongs: high, middle, etc.

7) Adding hyperlinks:

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

3.18. customizeChartElement

customizeChartElement(customizeChartElementFunction: Function)

[starting from version: 2.8.8]

This API call allows customizing separate chart elements in Flexmonster Charts. For example, you can add custom attributes or set element colors.

Note that customizeChartElement can be defined in two ways:

1. as a regular API call: flexmonster.customizeChartElement(customizeChartElementFunction);
2. as an initialization parameter: new Flexmonster({customizeChartElement: customizeChartElementFunction, ...})

Parameters

customizeChartElementFunction or null in case you do not need to change anything. The customizeChartElementFunction function has the following signature: customizeChartElementFunction(element: HTMLElement | SVGElement, data: Chart Data Object | Chart Legend Data Object): Object.

Data passed to customizeChartElementFunction:

- element – HTMLElement (https://developer.mozilla.org/en-US/docs/Web/API/HTMLElement) | SVGElement (https://developer.mozilla.org/en-US/docs/Web/API/SVGElement). The object that contains the current representation of the chart element and through which the chart element representation can be customized.
- data – Chart Data Object (/api/chart-data-object/) | Chart Legend Data Object (https://www.flexmonster.com/api/chart-legend-data-object/). Contains information about the chart element.

Example

Highlighting chart elements containing info about the United States regardless of their position:

```
function customizeChartElementFunction(element, data) {
  let newColor = "purple";

  /* contains() is a function that checks whether
     a given member is present in rows or column
  */
  if (contains(data, "country.[united states]")) {
    /* highlight United States in rows or columns */
    if (data.chartType == "pie") {
      if (element.querySelector('path')) {
        element.querySelector('path').style.fill = newColor;
      }
    } else {
      element.style.fill = newColor;
    }
  }

  /* change the legend item color for United States */
  if (data && data.type == 'legend') {
```

```
    if (data.member && !data.isExpanded
    && data.member.uniqueName == "country.[united states]" ) {
      element.querySelector(".fm-icon-display").style
      .backgroundColor = newColor;
    }
    /* isChild() is a function that checks whether
       the current member is a child of the given member
    */
    if (data.tuple && isChild(data.tuple, data.member.uniqueName,
    "country.[united states]") && !data.isExpanded) {
      element.querySelector(".fm-icon-display").style
      .backgroundColor = newColor;
    }
  }
}
```

See the full code on JSFiddle (https://jsfiddle.net/flexmonster/0t8gv2cp/).

3.19. customizeContextMenu

customizeContextMenu(customizeFunction: Function)

[starting from version: 2.6]

This API call allows customizing the context menu. For example, you can create a context menu for a flat table or remove all context menu items for classic view.

customizeContextMenu can be defined in two ways:

1. as a regular API call: flexmonster.customizeContextMenu(customizeFunction: Function);
2. as an initialization parameter: new Flexmonster({customizeContextMenu: customizeFunction, ...})

Parameters

customizeFunction has the following signature: customizeFunction(items: Array): Array.

Data passed to the customizeFunction:

- items – Array of objects. Context menu items created by Flexmonster. Each object can have the following properties:
    - id – String. The ID of the menu item.
    - label – String. The name of the menu item.
    - handler – Function. The function that handles click on this item.
    - submenu (optional) – Array of objects. Array of submenu items. Each submenu item has the same structure as items.
    - isSelected (optional) – Boolean. Specifies whether the menu item is selected.
    - class (optional) – String. Adds a custom CSS class to the specified item.
- data – Object. Information about the right-clicked object. In case of right click on the grid, this is a Cell Data Object (/api/cell-object/) containing information about the cell. In case of right click on the chart element, this is a Chart Data Object (/api/chart-data-object/) containing information about the chart segment.

- viewType – String. View type that was right-clicked. Can have one of the four possible values:
    1. "pivot" – Means that pivot grid was right-clicked, either compact or classic view.
    2. "flat" – Means that flat table view was right-clicked.
    3. "charts" – Means that charts view was right-clicked.
    4. "drillthrough" – Means that drill through view was right-clicked.

The customizeFunction function must return items, the array of new or changed context menu items. If items is null, the default items will be used. If items is [], the context menu will be hidden.

Examples

1) Add a “Switch to charts” option to the flat table context menu:

```
flexmonster.customizeContextMenu(function(items, data, viewType) {
 if (viewType == "flat")
  items.push({
   label: "Switch to charts",
   handler: function() {
    flexmonster.showCharts();
   }
  });
 return items;
});
```

Open the example on JSFiddle (https://jsfiddle.net/flexmonster/1p2qbx2t/).

2) Remove “Aggregation” item from all the context menus:

```
flexmonster.customizeContextMenu(function(items, data, viewType) {
 items = items.filter(function (item){
    return item.label !== "Aggregation"
   })
 return items;
});
```

Open the example on JSFiddle (https://jsfiddle.net/flexmonster/q254Lsef/).

See also

Customizing the context menu (/doc/customizing-context-menu/)

3.20. 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/).

3.21. embedPivotComponent

[removed]

embedPivotComponent method was deprecated in version 2.3. You should use new Flexmonster() (/api/new-flexmonster/) instead.

3.22. 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. Set it to false if you want to expand nodes without the drill-down. When set to true, the drill-down is performed as well. *Default value: true.*

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/)

3.23. expandData

expandData(hierarchyName: String)

[starting from version: 1.6]

Expands all nodes of the specified hierarchy. Please note, this method works only for CSV and JSON data sources.

Example

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

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

See also

expandAllData (/api/expandalldata/)
collapseAllData (/api/collapsealldata/)
collapseData (/api/collapsedata/)

3.24. exportTo

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

[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 – String. A type of export. There are such types available: "csv", "html", "pdf", "image" and "excel".
- params (optional) – Object. It contains export parameters. The object can have the following properties:
    - alwaysEnclose (optional) – Boolean. Indicates whether to enclose all CSV fields in quotes. When set to true, the fields are always enclosed in quotes. Otherwise, they will be enclosed only when necessary (e.g., if a field contains a comma: Bike, "$15,000", blue). Only for CSV export. *Default value: false*.
    - filename – String. A default name of the resulting file.
    - destinationType – String. It 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).
        - "plain" – the component's content will be returned with callbackHandler. Below are the types of the component's content for each type of export:
            - for HTML export: String
            - for CSV export: String
            - for Excel export: Uint8Array
            - for image export: HTMLCanvasElement
            - for PDF export: jsPDF object (https://github.com/MrRio/jsPDF). This destination

type allows modification of the generated PDF file. jsPDF is a library which generates PDFs in client-side JavaScript. After export from Flexmonster, jsPDF object can be modified using jsPDF API (https://rawgit.com/MrRio/jsPDF/master/docs/) and then saved. See the example (https://jsfiddle.net/flexmonster/qhkvmhn5/).

- excelSheetName (starting from v2.2) (optional) – String. To configure the sheet name when exporting to Excel file.
- fieldSeparator (optional) – String. Defines specific fields separator to split CSV row in the export file (only for CSV export). *Default value: ,*.
- fontUrl (from v2.7.7) (optional) – String. The URL to the TTF font file for saving PDF reports in Chinese, Arabic or any other language. Check out the list of ready-to-use Google Noto Fonts (https://www.flexmonster.com/doc/export-and-print/#!cdn-fonts-list) that you can use to support almost any language in the world. Only fonts in standard TTF format are supported.
- footer (starting from v2.211) (optional) – String. For Excel and CSV, there's an option to apply the multirow footer in the following way: "Row1\nRow2\nRow3".
  For PDF, HTML, and Image exports, footer can also be set in HTML format (tags, inline styles, img with base64 src). For Image and 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 and Image exports.
  Note: for CSV, Excel, and Image exports, footer is available starting from version 2.7.24.
- header (starting from v2.211) (optional) – String. For Excel and CSV, there's an option to apply the multirow header in the following way: "Row1\nRow2\nRow3".
  For PDF, HTML, and Image exports, header can also be set in HTML format (tags, inline styles, img with base64 src). For Image and 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 and Image exports.
  Note: for CSV, Excel, and Image exports, header is available starting from version 2.7.24.
- pageFormat (optional) – String. It defines the page format for a PDF file. There are such types available: "A0", "A1", "A2", "A3", "A4", "A5". *Default value: "A4"*.
- pageOrientation (optional) – String. It 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.
- requestHeaders (starting from v2.6.13) (optional) – Object. It allows adding custom request headers when exporting the file to a server. The object consists of "key": "value" pairs, where "key" is a header name and "value" is its value.
- 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: false*.
- url – String. A path to a server-side script that can save the file to the server. Use this parameter only if the destinationType parameter is "server".
- 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).
- useCustomizeCellForData (optional) – Boolean. Excel only. Specifies how cells modified by customizeCell are exported: as formatted strings (true) or as numbers without formatting (false). *Default value: true*.

- callbackHandler (optional) – Function. A JS function that is called when the data is ready to be exported. It has the following parameters:
  - result – Object. Describes the result of the export. If the export fails, the result will be null. The result object can have the following properties:
    - data – String | Uint8Array. The data to be exported. The data's type depends on the type of the export: String for "csv" and "html", Uint8Array for "excel", "image", and "pdf".
      Note that the types are different for the "plain" destination type – learn more here (#plain).

        - filename – String. The name of the resulting file.
        - response (optional) – String. The server's response. This property is defined in the result only when the destinationType is set to "server".
        - type – String. The export type.
    - error – Object. Describes the error with export. If the export is successful, the error will be null. Check out how to handle errors when exporting to a server (https://jsfiddle.net/flexmonster/L29r8dmv/).
      The error object can have the following properties:
        - message – String. The error message.
        - response (optional) – String. The server's response. This property is defined in the error only when the destinationType is set to "server".
        - status (optional) – Number. The response status code. This property is defined in the error only when the destinationType is set to "server".

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 PDF, modify generated file and save locally:

```
flexmonster.exportTo("pdf", { destinationType: "plain" }, function(res) {
  var pdf = res.data;
  pdf.addPage();
  pdf.text('Hello world!', 10, 10);
  pdf.save(res.filename);
});
```

3) 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); }
);
```

4) Export to HTML and save as local file:

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

5) 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);
```

6) Export to Excel and save as local file:

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

7) Export to PDF and set TTF font file:

```
flexmonster.exportTo('pdf', {
  fontUrl: 'https://cdn.flexmonster.com/fonts/NotoSansCJKtc-Regular.ttf'
});
```

See also

print (/api/print/)
Export and print tutorial (/doc/export-and-print/)

3.25. fullScreen

fullScreen()

[removed]

fullscreen method was removed in version 2.402. You should use Fullscreen from the Toolbar instead.

3.26. 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 the array is a 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/)

3.27. getAllHierarchies

getAllHierarchies(): Array

[starting from version: 1.4]

Returns a list of all available hierarchies.

Returns

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

- caption – String. Hierarchy caption.
- uniqueName – String. Unique hierarchy name.
- levels – Array. The levels of the hierarchy.
- type – String. Hierarchy type. The following types are supported: "string", "number", "date", "date string", "year/month/day", "year/quarter/month/day", "datetime", "time".

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", type: "string"},
{caption: "Category", uniqueName: "Category", type: "string"},
{caption: "Country", uniqueName: "Country", type: "string"}
]
*/
```

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

See also

getAllMeasures (/api/getallmeasures/)
getColumns (/api/getcolumns/)
getRows (/api/getrows/)
getReportFilters (/api/getreportfilters/)
getMeasures (/api/getmeasures/)

3.28. getAllMeasures

getAllMeasures(): Array

[starting from version: 1.4]

Returns a list of all available measures.

Returns

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

- name – String. The measure's name.
- uniqueName – String. The measure's unique name.
- aggregation – String. The name of the aggregation applied to the measure. If the measure is calculated, aggregation is set to "none".
- availableAggregations – Array of strings. Represents the list of aggregation functions that can be applied to the current measure. If the measure is calculated, availableAggregations is set to [].
- availableAggregationsCaptions – Array of strings. Represents the list of available aggregations' captions. If the measure is calculated, availableAggregationsCaptions is set to [].
- caption – String. The measure's caption.
- format – String. The name of the number formatting that will be applied to the measure.
- formula (optional) – String. It represents the formula. For calculated measures.
- grandTotalCaption– String. The measure's grand total caption.
- groupName (optional) – String. The measure's group defined in SSAS. Only for the "microsoft analysis services" data source type.
- type – String. The measure's type. The following types are supported: "string", "number", "date", "date string", "year/month/day", "year/quarter/month/day", "datetime", "time".

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
", "min", "max", "percent", "percentofcolumn"],
        availableAggregationsCaptions: ["Sum", "Count", "Distinct Count", "Average",
 "Product", "Min", "Max", "Percent", "Percent of Column"],
        caption: "Sum of Sales",
        format: "currency",
        grandTotalCaption: "Total Sum of Sales",
        name: "Sales",
        type: "number",
        uniqueName: "Sales"
    },
    {
        aggregation: "none",
        availableAggregations: [ ],
        availableAggregationsCaptions: [ ],
        caption: "Test",
        format: "",
        formula: "(SUM("Price") / count("Price")) * 100",
        grandTotalCaption: "Total Test",
        name: "Test",
```

```
        type: "number",
        uniqueName: "Test"
    }
]
*/
```

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

See also

getMeasures (/api/getmeasures/)
getAllHierarchies (/api/getallhierarchies/)
getColumns (/api/getcolumns/)
getRows (/api/getrows/)
getReportFilters (/api/getreportfilters/)

3.29. getCell

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

[starting from version: 1.4]

Returns information about the cell by row and column indexes.

Parameters

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

Returns

Cell Data 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/)

3.30. 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 the array contains the following properties:

- caption – String. Hierarchy caption.
- uniqueName – String. Unique hierarchy name.
- sort – String. Sorting type ("asc" or "desc").
- type – String. Hierarchy type. The following types are supported: "string", "number", "date", "date string", "year/month/day", "year/quarter/month/day", "datetime", "time".

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", type: "string"
},
{caption: "Category", uniqueName: "Category", sort: "asc", type: "string"}
]
*/
```

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

See also

getAllHierarchies (/api/getallhierarchies/)
getRows (/api/getrows/)
getReportFilters (/api/getreportfilters/)
getAllMeasures (/api/getallmeasures/)
getMeasures (/api/getmeasures/)

3.31. getColumnWidth

[removed]

getColumnWidth method was removed in version 2.3.

3.32. 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 the existing conditional formatting rule.

Parameters

- id – String. The 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/)

3.33. getData

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

[starting from version: 2.3]

Note! This method is only for integration with 3rd party charting libraries.

getData 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. Note: this property is available for CSV and JSON data sources only.
- callbackHandler – Function. Gets two input parameters – rawData and error. Tracks when the data is ready.
- updateHandler (optional) – Function. Gets two input parameters – rawData and error. Tracks if the data in the pivot table was filtered/sorted/etc or a number format was changed.

Response

rawData is an 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. 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.

error is an object that is returned if getData() gets terminated if the web page is likely to crash due to the too large dataset uploaded. Unless the object is undefined, it's asynchronously passed to callbackHandler and updateHandler. It has the following structure:

- dataHeight – Number. The number of rows from the report that failed to be retrieved by getData().
- dataWidth – Number. The number of columns that failed to be retrieved by getData().
- errorMessage – String. An error message. Its text description can be changed via the localization file (https://github.com/flexmonster/pivot-localizations/blob/586186fba870bd7fa9a43b7b76a3d9e86d7e96e7/en.json#L286). *Default value: "Dataset is too large. Some fields cannot be expanded. Please narrow down the dataset."*.

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: "",
            negativeCurrencyFormat: "-$1",
            positiveCurrencyFormat: "$1",
            decimalPlaces: -1,
            decimalSeparator: ".",
            divideByZeroValue: "Infinity",
            infinityValue: "Infinity",
            isPercent: false,
            maxDecimalPlaces: -1,
            maxSymbols: 20,
            negativeNumberFormat: "-1",
            nullValue: "",
            textAlign: "right",
            thousandsSeparator: " ",
            beautifyFloatingPoint: true
        }],
        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/).

Read a tutorial how getData is used for Integration with any charting library (/doc/integration-with-any-charting-library/).

3.34. getFilter

getFilter(hierarchyName: String): Filter Object (/api/filter-object/)

[starting from version: 1.4]

Returns the Filter Object (/api/filter-object/) for the specified hierarchy.

Parameters

- hierarchyName – String. The name of the hierarchy.

Returns

Filter Object (/api/filter-object/) that contains filtering information.

Example

```
flexmonster.setFilter("Category",
  {
    "members": [
      "category.[bikes]",
      "category.[cars]"
    ]
  }
);
flexmonster.getFilter('Category');

/*
method getFilter() returns the following Object:
{
  "members": [
    "category.[bikes]",
    "category.[cars]"
  ]
}
*/
```

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

See also

clearFilter (/api/clearfilter/)
setFilter (/api/setfilter/)

3.35. getFilterProperties

[removed]

getFilterProperties method was removed in version 2.7. Use getFilter() (/api/getFilter/) instead.

3.36. getFlatSort

getFlatSort(): Array

[starting from version: 2.8.2]

Returns an array of objects defining the sorting on the flat grid.

Note that the getFlatSort method is available only for reports based on "csv", "json", and "api" data source types.

Returns

Array of objects. Each object has the following properties:

- uniqueName – String. The unique name of the hierarchy being sorted.
- sort – String. The sorting type ("asc", "desc", or "undefined").

If the flat sorting isn't specified, or all the sorted hierarchies have the "undefined" sorting type, an empty array is returned.

Example

```
flexmonster.getFlatSort();

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

Try the example on JSFiddle (https://jsfiddle.net/flexmonster/wrqLg3pv/).

See also

setFlatSort (/api/setflatsort/)
getSort (https://www.flexmonster.com/api/getsort/)
setSort (https://www.flexmonster.com/api/setsort/)
sortingMethod (https://www.flexmonster.com/api/sortingmethod/)

3.37. 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 a report or via the 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) – String. The unique name of the measure. If the measure's unique name is not specified or 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/)

3.38. getLabels

[removed]

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

3.39. 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 the array contains the following parameters:

- name – String. The measure’s name.
- uniqueName – String. The measure’s unique name.
- aggregation – String. The name of the aggregation applied to the measure. If the measure is calculated, aggregation is set to "none".
- availableAggregations – Array of strings. Represents the list of aggregation functions that can be applied to the current measure. If the measure is calculated, availableAggregations is set to [].
- availableAggregationsCaptions – Array of strings. Represents the list of available aggregations’ captions. If the measure is calculated, availableAggregationsCaptions is set to [].
- caption – String. The measure’s caption.
- format – String. The name of the number formatting that will be applied to the measure.
- formula (optional) – String. It represents the formula. For calculated measures.
- grandTotalCaption– String. The measure’s grand total caption.
- groupName (optional) – String. The measure’s group defined in SSAS. Only for the "microsoft analysis

services" data source type.
- type – String. The measure's type. The following types are supported: "string", "number", "date", "date string", "year/month/day", "year/quarter/month/day", "datetime", "time".

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",
        type: "number",
        uniqueName: "Sales"
    },
    {
        aggregation: "sum",
        availableAggregations: ["sum", "count", "distinctcount", "average", "product
", "min", "max", "percent", "percentofcolumn", "percentofrow", "index"],
        availableAggregationsCaptions: ["Sum", "Count", "Distinct Count", "Average",
 "Product", "Min", "Max", "% of Grand Total", "% of Column", "% of Row", "Index"],
        caption: "Sum of Orders",
        format: "",
        grandTotalCaption: "Total Sum of Orders",
        name: "Orders",
        type: "number",
        uniqueName: "Orders"
    }
]
*/
```

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

See also

getAllMeasures (/api/getallmeasures/)
getAllHierarchies (/api/getallhierarchies/)
getColumns (/api/getcolumns/)
getRows (/api/getrows/)
getReportFilters (/api/getreportfilters/)

3.40. getMembers

getMembers(hierarchyName: String, memberName: String, callbackHandler: Function): 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 – String. The name of the hierarchy.
- memberName (optional) – String. The 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) – Function. A callback handler becomes useful when you work 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. A 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

An 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 an array of objects
[
  {caption: "Accessories", hierarchyName: "Category",
   uniqueName: "category.[accessories]", children: [],
   isLeaf:true, parentMember:"(All)"
  },
  {caption: "Cars", hierarchyName: "Category",
   uniqueName: "category.[cars]", children: [],
   isLeaf:true, parentMember:"(All)"
  },
  {caption: "Clothing", hierarchyName: "Category",
   uniqueName: "category.[clothing]", children: [],
   isLeaf:true, parentMember:"(All)"
  }
]
*/
```

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

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

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

/* method returns an 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:
[{
    caption: "Accessories",
    dimensionName: "[Product]",
    hierarchyCaption: "Product Categories",
    hierarchyName: "[Product].[Product Categories]",
    isLeaf: false,
    parentMember: "[Product].[Product Categories].[All Products]",
    uniqueName: "[Product].[Product Categories].[Category].&[4]"
 }
 {
    caption: "Bikes",
    dimensionName: "[Product]",
    hierarchyCaption: "Product Categories",
    hierarchyName: "[Product].[Product Categories]",
    isLeaf: false,
```

```
    parentMember: "[Product].[Product Categories].[All Products]",
    uniqueName: "[Product].[Product Categories].[Category].&[1]"
 }
 {
    caption: "Clothing",
    dimensionName: "[Product]",
    hierarchyCaption: "Product Categories",
    hierarchyName: "[Product].[Product Categories]",
    isLeaf: false,
    parentMember: "[Product].[Product Categories].[All Products]",
    uniqueName: "[Product].[Product Categories].[Category].&[3]"
 }
 {
    caption: "Components",
    dimensionName: "[Product]",
    hierarchyCaption: "Product Categories",
    hierarchyName: "[Product].[Product Categories]",
    isLeaf: false,
    parentMember: "[Product].[Product Categories].[All Products]",
    uniqueName: "[Product].[Product Categories].[Category].&[2]"
 }]

now you can ask for '[Product].[Product Categories].[Category].&[4]' members:
flexmonster.getMembers('[Product].[Product Categories]',
'[Product].[Product Categories].[Category].&[4]', onGetMembers);
*/
```

Open the example on JSFiddle (https://jsfiddle.net/flexmonster/0b8fm965/).

3.41. 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 = "off";
flexmonster.setOptions(options);
flexmonster.refresh();
```

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

See also

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

3.42. getPages

[renamed]

getPages method was deprecated in version 2.4. You should use getReportFilters() (/api/getreportfilters/) instead.

3.43. getReport

getReport(options: Object): Report Object (/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 the 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). *Default value: false*.
- withGlobals (optional) – Boolean. Indicates whether the options defined in the global object will be included in the report (true) or not (false). *Default value: false*.

Example

1) Get report:

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

<script>
var pivot = new Flexmonster({
    container: "pivotContainer",
    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/flexmonster.js"></script>

<script>
var pivot1 = new Flexmonster({
    container: "firstPivotContainer",
    toolbar: true,
    report: {
        dataSource: {
            filename: "data1.csv"
        }
    }
});
var pivot2 = new Flexmonster({
    container: "secondPivotContainer",
    toolbar: true,
    report: {
        dataSource: {
            filename: "data2.csv"
        }
    }
});
</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 a 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/)

3.44. getReportFilters

getReportFilters(): Array

[starting from version: 2.4]

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

Returns

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

- caption – String. Hierarchy caption.
- uniqueName – String. Unique hierarchy name.
- sort – String. Sorting type ("asc" or "desc").
- type – String. Hierarchy type. The following types are supported: "string", "number", "date", "date string", "year/month/day", "year/quarter/month/day", "datetime", "time".

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

Example

```
flexmonster.getReportFilters();

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

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/)

3.45. getRowHeight

[removed]

getRowHeight method was removed in 2.3 version.

3.46. 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 the array contains the following properties:

- caption – String. Hierarchy caption.
- uniqueName – String. Unique hierarchy name.
- sort – String. Sorting type ("asc" or "desc").
- type – String. Hierarchy type. The following types are supported: "string", "number", "date", "date string", "year/month/day", "year/quarter/month/day", "datetime", "time".

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", type: "str
ing"},
    {caption: "Category", uniqueName: "Category", sort: "asc", type: "string"}
]
*/
```

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

See also

getAllHierarchies (/api/getallhierarchies/)
getColumns (/api/getcolumns/)
getReportFilters (/api/getreportfilters/)
getAllMeasures (/api/getallmeasures/)
getMeasures (/api/getmeasures/)

3.47. getSelectedCell

getSelectedCell(): Cell Data Object (/api/cell-object/) | Array of Cell Data Objects (/api/cell-object/)

[starting from version: 1.4]

Returns information about the selected cell/cells.

Returns

Cell Data Object (/api/cell-object/) which contains information about selected cell. If multiple cells are selected, getSelectedCell returns an array of Cell Data Objects (/api/cell-object/).

Example

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

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

See also

getCell (/api/getcell/)
removeSelection (/api/removeselection/)

3.48. getSort

getSort(hierarchyName: String): String

[starting from version: 1.4]

Returns the sort type which is applied to the hierarchy.

Parameters

- hierarchyName – String. 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/)
sortingMethod (/api/sortingmethod/)
sortValues (/api/sortvalues/)

3.49. getTableSizes

getTableSizes(): Table Sizes Object (https://www.flexmonster.com/api/table-sizes-object/)

[starting from version 2.8.19]

Returns table sizes that are set for the component.

Returns

Table Sizes Object (https://www.flexmonster.com/api/table-sizes-object/) that contains information about table

sizes.

Example

```
flexmonster.getTableSizes()
```

Open the example on JSFiddle (https://jsfiddle.net/flexmonster/e9oruwsx/).

See also

setTableSizes (/api/settablesizes/)
getReport (https://www.flexmonster.com/api/getreport/)
setReport (https://www.flexmonster.com/api/setreport/)

3.50. getValue

[removed]

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

3.51. getXMLACatalogs

getXMLACatalogs(proxyURL: String, dataSourceInfo: String, callbackHandler: Function, 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 – String. The path to proxy URL to the OLAP data source, such as Microsoft Analysis Services and Mondrian.
- dataSourceInfo – String. The service info of the OLAP data source.
- callbackHandler – Function. A JS function which will be called when the component obtains a list of all available catalogs.
- username (optional) – String. The name of the user account at the server. This parameter is necessary to complete the authentication process.
- password (optional) – String. 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 type. 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 type = '';
var catalog = '';
var cube = '';

function diagnostic() {
  getDataSources();
}

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

function onDataSourceLoaded(result) {
  dataSourceInfo = result[0];
  type = 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({
    type: type,
    proxyUrl: proxyUrl,
    dataSourceInfo: dataSourceInfo,
    catalog: catalog,
    cube: cube
  });
}

diagnostic();
```

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

See also

getXMLADataSources (/api/getxmladatasources/)
getXMLAProviderName (/api/getxmlaprovidername/)
getXMLACubes (/api/getxmlacubes/)

3.52. getXMLACubes

getXMLACubes(proxyURL: String, dataSourceInfo: String, catalog: String, callbackHandler: Function, 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 – String. The path to proxy URL to the OLAP data source, such as Microsoft Analysis Services and Mondrian.
- dataSourceInfo – String. The service info of the OLAP data source.
- catalog – the data source catalog name of the OLAP data source.
- callbackHandler – Function. A JS function which will be called when the component obtains a list of all available cubes.
- username (optional) – String. The name of the user account at the server. This parameter is necessary to complete the authentication process.
- password (optional) – String. 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 type. 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 type = '';
var catalog = '';
var cube = '';

function diagnostic() {
  getDataSources();
}

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

function onDataSourceLoaded(result) {
  dataSourceInfo = result[0];
  type = 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({
    type: type,
    proxyUrl: proxyUrl,
    dataSourceInfo: dataSourceInfo,
    catalog: catalog,
    cube: cube
  });
}

diagnostic();
```

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

See also

getXMLADataSources (/api/getxmladatasources/)
getXMLAProviderName (/api/getxmlaprovidername/)
getXMLACatalogs (/api/getxmlacatalogs/)

3.53. getXMLADataSources

getXMLADataSources(proxyURL: String, callbackHandler: Function, 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 – String. The path to proxy URL to the OLAP data source, such as Microsoft Analysis Services and Mondrian.
- callbackHandler – Function. A JS function which will be called when the component obtains a list of all data sources.
- username (optional) – String. The name of the user account at the server. This parameter is necessary to complete the authentication process.
- password (optional) – String. 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 type. 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 type = '';
var catalog = '';
var cube = '';

function diagnostic() {
  getDataSources();
}

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

function onDataSourceLoaded(result) {
  dataSourceInfo = result[0];
  type = 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({
    type: type,
    proxyUrl: proxyUrl,
    dataSourceInfo: dataSourceInfo,
    catalog: catalog,
    cube: cube
  });
}

diagnostic();
```

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

See also

getXMLAProviderName (/api/getxmlaprovidername/)
getXMLACatalogs (/api/getxmlacatalogs/)
getXMLACubes (/api/getxmlacubes/)

3.54. getXMLAProviderName

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

[starting from version: 1.5]

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

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

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

Parameters

- proxyUrl – String. The path to proxy URL to the OLAP data source, such as Microsoft Analysis Services and Mondrian.
- callbackHandler (optional) – Function. A JS function which will be called when the component is ready to give type.
- username (optional) – String. The name of the user account at the server. This parameter is necessary to complete the authentication process.
- password (optional) – String. 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 type. 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 type = '';
var catalog = '';
var cube = '';

function diagnostic() {
  getDataSources();
}

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

function onDataSourceLoaded(result) {
  dataSourceInfo = result[0];
  type = 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({
    type: type,
```

```
    proxyUrl: proxyUrl,
    dataSourceInfo: dataSourceInfo,
    catalog: catalog,
    cube: cube
  });
}

diagnostic();
```

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

See also

getXMLADataSources (/api/getxmladatasources/)
getXMLACatalogs (/api/getxmlacatalogs/)
getXMLACubes (/api/getxmlacubes/)

3.55. gridColumnCount

[removed]

gridColumnCount method was removed in version 2.3.

3.56. gridRowCount

[removed]

gridRowCount method was removed in version 2.3.

3.57. load

load(url: String, requestHeaders: Object)

[starting from version: 1.4]

Loads the report file from the specified URL.

Parameters

- url – String. The URL to the report file.
- requestHeaders (optional) – Object. Allows you to add custom request headers when loading a report from a server. This object consists of "key": "value" pairs, where "key" is a header's name and "value" is its value.

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/).

3) Load a report from a resource that requires specific request headers (e.g., authorization headers):

```
flexmonster.load("http://yourserverwithauth.com/report_json.php",
    {
        AuthToken: "XXXX"
    }
);
```

See also

open (/api/open/)
save (/api/save/)
getReport (/api/getreport/)
setReport (/api/setreport/)

3.58. off

off(eventName: String, functionName: String)

[starting from version: 2.3]

Removes JS handlers for specified event.

Parameters

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

//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/)

3.59. on

on(eventName: String, function: Function)

[starting from version: 2.3]

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

Parameters

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

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/)

3.60. open

open()

[starting from version: 1.6]

Opens local report file. Use this API call to open the report JSON file from the local file system.

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/)

3.61. openCalculatedValueEditor

openCalculatedValueEditor(uniqueName: String, callbackHandler: Function)

[starting from version: 2.7.9]

Opens the calculated value pop-up window editor. Calling openCalculatedValueEditor results in opening an empty editor for creating a new value. It is also possible to open this editor for editing or deleting an existing calculated value.

Parameters

- uniqueName (optional) – String. The unique name of the calculated value. It is used for editing or deleting an existing calculated value.
- callbackHandler (optional) – Function. Called after a particular action (clicking the Apply or Delete button). The callbackHandler takes the response object which has the following properties:
    - uniqueName – String. The unique name of the calculated value.
    - isRemoved – Boolean. Specifies whether the calculated value was removed (true) or not (false).

Examples

1) How to open the calculated value editor without parameters:

```
flexmonster.openCalculatedValueEditor();
```

Check out on JSFiddle (https://jsfiddle.net/flexmonster/m2tkrc8s/).

2) How to open the calculated value editor for the previously defined calculated value (Revenue):

```
flexmonster.openCalculatedValueEditor("Revenue");
```

Check out on JSFiddle (https://jsfiddle.net/flexmonster/m2tkrc8s/).

3) How to open the calculated value editor for the previously defined calculated value (Revenue) and send a function as a callback:

```
flexmonster.openCalculatedValueEditor("Revenue", function(response) {
    console.log(response);
});
```

Check out on JSFiddle (https://jsfiddle.net/flexmonster/m2tkrc8s/).

See also

Calculated values tutorial (https://www.flexmonster.com/doc/calculated-values/)
addCalculatedMeasure (https://www.flexmonster.com/api/addcalculatedmeasure/)
getAllMeasures (https://www.flexmonster.com/api/getallmeasures/)
getMeasures (https://www.flexmonster.com/api/getmeasures/)
removeAllCalculatedMeasures (https://www.flexmonster.com/api/removeallcalculatedmeasures/)
removeCalculatedMeasure (https://www.flexmonster.com/api/removecalculatedmeasure/)
(https://www.flexmonster.com/doc/calculated-values/)

3.62. openFieldsList

openFieldsList()

[starting from version: 1.4]

Opens the Field List.

Example

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

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

See also

closeFieldsList (/api/closefieldslist/)

3.63. openFilter

openFilter(hierarchyUniqueName: String)

[starting from version: 2.6.7]

This API call opens the filter pop-up window for the specified hierarchy.

Parameters

hierarchyUniqueName – String. The unique name of the hierarchy for which the filter pop-up window should be opened.

Example

Show the filter pop-up window for Category:

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

Open the example on JSFiddle (https://jsfiddle.net/flexmonster/jofLbat5/).

See also

filteropen (/api/filteropen/)
filterclose (/api/filterclose/)

3.64. percentZoom

[removed]

percentZoom method was removed in version 2.3.

3.65. print

print(options: Object)

[starting from version: 1.4]

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

Parameters

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

- header (starting from v2.211) (optional) – String. The 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/)

3.66. 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/)

3.67. removeAllCalculatedMeasures

removeAllCalculatedMeasures()

[starting from version: 2.3]

Note that the removeAllCalculatedMeasures method is available only for reports based on "csv", "json", and "api" data source types.

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/)

3.68. 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/)

3.69. removeAllMeasures

[removed]

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

3.70. removeCalculatedMeasure

removeCalculatedMeasure(measureName: String)

[starting from version: 2.3]

Note that the removeCalculatedMeasure method is available only for reports based on "json", "csv", and "api" data source types.

Removes the calculated measure by the measure’s unique name.

Parameters

- measureName – String. 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/)

3.71. 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 – String. 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/)

3.72. removeMeasure

[removed]

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

3.73. 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/)

3.74. runQuery

runQuery(query: Query Object)

[starting from version: 1.6]

Runs a query with specified rows, columns, measures and reportFilters from 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

- query – Query Object. It contains rows, columns, measures, and reportFilters from Slice Object (/api/slice-object/).

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/)

3.75. save

save(params: Object)

[starting from version: 2.302]

Saves the current report to a specified location (e.g., to a server or to the local file system). The report is saved in JSON format.

Parameters

params – Object. It can have the following properties:

- filename – String. A default name of the file.
- destination (optional) – String. Defines where to save the generated file. The destination can be either "server" (the file will be saved to the server) or "file" (the file will be saved to the local file system). When saving to a server, Flexmonster creates an XMLHttpRequest (https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest) and sends it as a POST request.
  Default value: "file".
- callbackHandler (optional) – Function. A JS function that is called when the report is saved. It has the following parameters:
  - result – Object. Describes the result of saving. If saving fails, the result will be null. The result object can have the following properties:
    - report – Report Object (https://www.flexmonster.com/api/report-object/). The saved report.
    - response (optional) – String. The server's response. This property is defined in the result only when the destination is set to "server".
    - status (optional) – Number. The response status code. This property is defined in the result only when the destination is set to "server".
  - error – Object. Describes the error with saving. If saving is successful, the error will be null. The error object can have the following properties:
    - message – String. The error message.
    - response (optional) – String. The server's response. This property is defined in the error only when the destination is set to "server".
    - status (optional) – Number. The response status code. This property is defined in the error only when the destination is set to "server".
- url (optional) – String. A URL to the server-side script which saves the generated file. The file is sent as a POST parameter. Use this parameter only if the destination parameter is "server".
- embedData (optional) – Boolean. Specifies whether to save CSV data within the report or not. *Default value: false*.
- reportType (optional) – String. The report can be saved in "json" or "xml" format. Starting from version 2.6.0, the "xml" format is deprecated. *Default value: "json"*.
- requestHeaders (optional) – Object. Allows you to add custom request headers when saving the report to

a server. This object consists of "key": "value" pairs, where "key" is a header's name and "value" is its value.
- withDefaults (optional) – Boolean. Indicates whether the default values for options will be included in the report (true) of not (false). *Default value: false*.
- withGlobals (optional) – Boolean. Indicates whether the options defined in the global object will be included in the report (true) or not (false). *Default value: 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'
});
```

Note: the server-side script should be created on your back end to save reports to the server. And the url parameter is the path to this server-side script.

3) How to handle errors when saving to a server:

```
flexmonster.save(
    {
        filename: 'report.json',
        destination: 'server',
        url: 'https://httpstat.us/404',
        callbackHandler: function(result, error) {
            if (error) {
                alert("response: " + error.response + "\nstatus: "
                + error.status + "\nmessage: "+ error.message);
            }
        }
    }
);
```

See the live sample on JSFiddle (https://jsfiddle.net/flexmonster/q0prh49f/).

See also

load (/api/load/)
getReport (/api/getreport/)
setReport (/api/setreport/)

3.76. scrollToColumn

scrollToColumn(columnIndex: Number)

[starting from version: 2.8.27]

Scrolls the grid to the specified column.

Parameters

- columnIndex – Number. The index of the column to which the grid will be scrolled. Indexes start from 0.

Example

```
flexmonster.scrollToColumn(10);
```

See the full code on JSFiddle (https://jsfiddle.net/flexmonster/odyqw4zv/).

See also

scrollToRow (https://www.flexmonster.com/api/scrolltorow/)

3.77. scrollToRow

scrollToRow(rowIndex: Number)

[starting from version: 2.8.27]

Scrolls the grid to the specified row.

Parameters

- rowIndex – Number. The index of the row to which the grid will be scrolled. Indexes start from 0.

Example

```
flexmonster.scrollToRow(10);
```

See the full code on JSFiddle (https://jsfiddle.net/flexmonster/odyqw4zv/).

See also

scrollToColumn (https://www.flexmonster.com/api/scrolltocolumn/)

3.78. setBottomX

[removed]

setBottomX method was removed in version 2.7. Use setFilter() (/api/setFilter/) instead.

3.79. setChartTitle

[removed]

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

3.80. setColumnWidth

[removed]

setColumnWidth method was removed in version 2.3.

3.81. setFilter

setFilter(hierarchyName: String, filter: Filter Object (/api/filter-object/))

[starting from version: 1.4]

Sets the filter for the specified hierarchy.

Parameters

- hierarchyName – String. The name of the hierarchy
- filter – Filter Object (/api/filter-object/). It contains filtering information.

Examples

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

```
flexmonster.setFilter("Category",
  {
    "members": [
      "category.[bikes]",
      "category.[cars]"
    ]
  }
);
```

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 exclude property is used:

```
flexmonster.setFilter("Category",
  {
    "exclude": [
      "category.[accessories]"
    ]
  }
);
```

instead of

```
flexmonster.setFilter("Category",
  {
    "members": [
      "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/)

3.82. setFlatSort

setFlatSort(flatSort: Array)

[starting from version: 2.8.2]

Sets the flat table multicolumn sorting.

Note that the setFlatSort method is available only for reports based on "csv", "json", and "api" data source types.

Parameters

Array of objects defining the sorting on the flat grid. Each object has the following properties:

- uniqueName – String. The unique name of the hierarchy being sorted.
- sort – String. The sorting type ("asc", "desc", or "undefined").

Note: the hierarchies are sorted in the order they were specified (i.e., the first hierarchy is sorted first, and so on). Therefore, take the element’s order into account when defining the flat table multicolumn sorting. See the example on JSFiddle (https://jsfiddle.net/flexmonster/3u1o2mry/).

Example

```
var sort = [
  {
      uniqueName: "Category",
      sort: "desc"
  },
  {
      uniqueName: "Price",
      sort: "asc"
  }
];
flexmonster.setFlatSort(sort);
```

Try the example on JSFiddle (https://jsfiddle.net/flexmonster/wrqLg3pv/).

See also

getFlatSort (/api/getflatsort/)
getSort (https://www.flexmonster.com/api/getsort/)
setSort (https://www.flexmonster.com/api/setsort/)
sortingMethod (https://www.flexmonster.com/api/sortingmethod/)

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

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

Parameters

- format – Format Object (/api/format-object/). It contains the number format parameters.
- measureName (optional) – String. The unique name of the measure. Specify the 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 the 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.name = "PriceFormat";
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/)
Number formatting tutorial (/doc/number-formatting/)

3.84. setGridTitle

[removed]

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

3.85. setHandler

[removed]

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

3.86. setLabels

[removed]

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

3.87. 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 – Options Object (/api/options-object/). It 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 = "off";
flexmonster.setOptions(options);
flexmonster.refresh();
```

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

3) How to set a chart title:

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

See also

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

refresh (/api/refresh/)
Options tutorial (/doc/options/)

3.88. 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 – Report Object (/api/report-object/). It describes the report and contains all its properties.

Example

1) Set report:

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

<script>
var pivot = new Flexmonster({
    container: "pivotContainer"
});
</script>

<button onclick="setReport()">Set Report</button>
<script>
function setReport() {
    var report = {
        dataSource: {
            filename: "http://cdn.flexmonster.com/data/data.csv"
        }
    }
    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/flexmonster.js"></script>

<script>
var pivot1 = new Flexmonster({
    container: "firstPivotContainer",
    toolbar: true,
```

```
    report: {
        dataSource: {
            filename: "data1.csv"
        }
    }
});
var pivot2 = new Flexmonster({
    container: "secondPivotContainer",
    toolbar: true,
    report: {
        dataSource: {
            filename: "data2.csv"
        }
    }
});
</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/)

3.89. setRowHeight

[removed]

setRowHeight method was removed in version 2.3.

3.90. setSelectedCell

[removed]

setSelectedCell method was removed in version 2.3.

3.91. 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/)
sortingMethod (/api/sortingmethod/)
sortValues (/api/sortvalues/)

3.92. setStyle

[removed]

setStyle method was removed in version 2.3.

3.93. setTableSizes

setTableSizes(tableSizes: Table Sizes Object (https://www.flexmonster.com/api/table-sizes-object/))

[starting from version 2.8.19]

Sets table sizes for the component. Use this method to set table sizes without updating the report.

When called, the setTableSizes() method overwrites previously set table sizes.

Parameters

- tableSizes – Table Sizes Object (https://www.flexmonster.com/api/table-sizes-object/). Contains new values for table sizes.

Examples

1) Set width of a column with a certain index:

```
flexmonster.setTableSizes({
  "columns": [
    {
      "idx": 0,
      "width": 160
    }
  ]
});
```

See the full code on JSFiddle (https://jsfiddle.net/flexmonster/e9oruwsx/).

2) Set height of a row containing a certain member:

```
flexmonster.setTableSizes({
  "rows": [
    {
      "tuple": ["category.[cars]"],
      "height": 80
    }
  ]
});
```

See the live sample on JSFiddle (https://jsfiddle.net/flexmonster/e9oruwsx/).

3) Set width of a column containing a certain member and measure:

```
flexmonster.setTableSizes({
  "columns": [
    {
      "tuple": ["country.[germany]"],
      "measure": {
        "uniqueName": "Price",
        "aggregation": "sum"
      },
      "width": 160
    }
  ]
});
```

See the full code on JSFiddle (https://jsfiddle.net/flexmonster/e9oruwsx/).

See also

getTableSizes (/api/gettablesizes/)
getReport (https://www.flexmonster.com/api/getreport/)
setReport (https://www.flexmonster.com/api/setreport/)

3.94. setTopX

[removed]

setTopX method was removed in version 2.7. Use setFilter() (/api/setFilter/) instead.

3.95. 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: "column", "bar_h", "line", "scatter", "pie", "stacked_column" and "column_line".
After showCharts() API call options.viewType property in the report will be "charts".

Parameters

- type (optional) – String. The type of charts to show. *Default value: "column".*
- multiple (optional) – Boolean. To show one measure on the chart (false) or multiple (true) – as many measures as selected in the slice. Available since version 1.9. *Default value: false.*

Examples

1) Show default chart type:

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

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

2) Show the pie chart:

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

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

See also

showGrid (/api/showgrid/)
showGridAndCharts (/api/showgridandcharts/)

3.96. 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/)

3.97. 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: "column", "bar_h", "line", "scatter", "pie", "stacked_column" and "column_line".
After showGridAndCharts() API call options.viewType property in the report will be "grid_charts".

Parameters

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

3.98. sortingMethod

sortingMethod(hierarchyName: String, compareFunction: Function)

[starting from version: 2.6]

Sets custom sorting for hierarchy members. For more details about defining custom sort please refer to Custom sorting tutorial (/doc/custom-sorting/).

Parameters

- hierarchyName – String. The unique name of the hierarchy to which the sorting is applied.
- compareFunction – Function. Defines the sort order. The input parameters are the same as for compareFunction of Array.sort() method (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/sort).

Example

The following example shows sorting with really simple compareFunction which sorts elements in reverse alphabetical order:

```
flexmonster.sortingMethod("Category", function (a, b) {
    return a < b ? 1 : -1;
});
```

Try the example on JSFiddle (https://jsfiddle.net/flexmonster/j85qom2p/).

See also

Custom sorting (/doc/custom-sorting/)
setSort (/api/setsort/)
getSort (/api/getSort/)
sortValues (/doc/sortValues/)

3.99. sortValues

sortValues(axisName: String, type: String, tuple: Array, measure: Object)

[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 the member's unique names.
- measure – Object. Identifies the measure on which sorting will be applied. Has the following properties:
    - uniqueName – String. Unique measure name.
    - aggregation (optional) – String. Measure the aggregation type.

Example

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

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

See also

getSort (/api/getsort/)
sortingMethod (/api/sortingmethod/)
setSort (/api/setsort/)

3.100. updateData

updateData(connectionParameters: Object, options: 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. For all data sources, updateData allows connecting to a new data source. For JSON data source it is also possible to update only some part of the data.

Parameters

- connectionParameters – Object. It contains connection parameters. The connectionParameters object has the same structure as the Data Source Object (https://www.flexmonster.com/api/data-source-object/).
- options (optional) – Object. Contains additional options:
    - partial (optional) – Boolean. For JSON data source only. Allows partial updating of the data. When partial is set to true, you can add, update, or remove certain records of JSON array. For more information, check example number 4 (#example-4). *Default value: false*.
    - ignoreSorting (optional) – Boolean. Set ignoreSorting: true and current sorting defined in the report will be ignored when you update the data. *Default value: false*.
    - ignoreScroll (optional) – Boolean. Set ignoreScroll: true and current scroll position in the pivot table will be ignored when you update the data. *Default value: false*.

Examples

1) Update data from Microsoft Analysis Services:

```
flexmonster.updateData({
    type: 'microsoft analysis services',
    proxyUrl: 'https://olap.flexmonster.com/olap/msmdpump.dll',
    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({
    type: 'csv',
    filename: 'data/data.csv'
});
```

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

3) Update data from JSON inline data:

```
var jsonData = [{
    "Category": "Accessories",
    "Size": "277 oz",
    "Color": "red",
    "Destination": "United Kingdom",
    "Business Type": "Warehouse",
    "Country": "United Kingdom",
    "Price": 1222,
    "Quantity": 730,
    "Discount": 38
  },
  {
    "Category": "Accessories",
    "Size": "47 oz",
    "Color": "white",
    "Destination": "United States",
    "Business Type": "Warehouse",
    "Country": "United States",
    "Price": 7941,
    "Quantity": 73,
    "Discount": 53
  },
  {
    "Category": "Bikes",
    "Size": "264 oz",
    "Color": "white",
    "Destination": "Australia",
    "Business Type": "Specialty Bike Shop",
    "Country": "Australia",
```

```
    "Price": 6829,
    "Quantity": 19,
    "Discount": 56
}];
flexmonster.updateData({ data: jsonData });
```

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

4) How to use updateData for adding/updating/removing partial data:

First of all, we add id and delete types to the first object of JSON array:

```
{
      "Category": { "type": "string" },
      "Price": { "type": "number" },
      "RowId": { "type": "id" },
      "DeleteRow": { "type": "delete" }
}
```

Then, when defining the data, we specify an id field for every object the following way:

```
{
      "Category": "Accessories",
      "Price": 242,
      "RowId": 1
}
```

To add or update only some of the records use partial: true:

```
var dataForUpdate = [{
      "Category": "Cars",
      "Price": 51844,
      "RowId": 4
}]
flexmonster.updateData({data: dataForUpdate}, {partial: true});
```

To delete the records specify id and delete fields:

```
var dataForUpdate = [{
      "RowId": 3,
      "DeleteRow": true
}]
flexmonster.updateData({data: dataForUpdate}, {partial: true});
```

Live example on JSFiddle (http://jsfiddle.net/flexmonster/n9hw8ab0/).

See also

connectTo (/api/connectto/)

3.101. zoomTo

[removed]

zoomTo method was removed in version 2.3.

3.102. jsCellClickHandler

[removed]

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

3.103. jsFilterOpenHandler

[removed]

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

3.104. jsFieldsListCloseHandler

[removed]

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

3.105. jsFieldsListOpenHandler

[removed]

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

3.106. jsFullScreenHandler

[removed]

jsFullScreenHandler event was removed in version 2.3.

3.107. jsPivotCreationCompleteHandler

[removed]

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

3.108. jsPivotUpdateHandler

[removed]

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

3.109. jsReportChangeHandler

[removed]

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

3.110. jsReportLoadedHandler

[removed]

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

4. Events

4.1. All 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)
afterchartdraw (/api/afterchartdraw/)	triggered after chart rendering
aftergriddraw (/api/aftergriddraw/)	triggered after grid rendering
beforegriddraw (/api/beforegriddraw/)	triggered before grid rendering
beforetoolbarcreated (/api/beforetoolbarcreated/)	triggered before the creation of the Toolbar
cellclick (/api/cellclick/)	triggered when a cell is clicked on the grid
chartclick (/api/chartclick/)	triggered when a chart element is clicked
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
drillthroughclose (/api/drillthroughclose/)	triggered when the drill-through view is closed
drillthroughopen (/api/drillthroughopen/)	triggered when the drill-through view is opened
exportcomplete (/api/exportcomplete/)	triggered after the export process has finished
exportstart (/api/exportstart/)	triggered when the export of grid or chart starts
fieldslistclose (/api/fieldslistclose/)	triggered when the built-in Field List is closed
fieldslistopen (/api/fieldslistopen/)	triggered when the built-in Field List is opened
filterclose (/api/filterclose/)	triggered when the filter pop-up window is closed
filteropen (/api/filteropen/)	triggered when the filter pop-up window is opened
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
unauthorizederror (/api/unauthorizederror/)	triggered when the Accelerator sends the 401 Unauthorized error in response to the Flexmonster request
update (/api/update/)	triggered when the change occurred in the component

4.2. afterchartdraw

[starting from version: 2.6.6]

It is triggered after chart rendering.

Example

How to use afterchartdraw:

```
flexmonster.on('afterchartdraw', function () {
 console.log('After chart draw!');
```

```
});
```

Open the example on JSFiddle (https://jsfiddle.net/flexmonster/32w7eLgc/).

See also

aftergriddraw (/api/aftergriddraw/)

4.3. aftergriddraw

[starting from version: 2.4]

It is triggered after grid rendering. Use beforegriddraw right before rendering.

Data passed to the handler

params – Object. It has the following parameter:

- smooth – Boolean. There are two types of grid draw: grid draw that adds row/column scroll and grid draw that adds pixel scroll. As a result,
  aftergriddraw is called twice. The first time it is called with smooth: false and it means that grid draw with row/column scroll has ended. The second time aftergriddraw is called with smooth: true and it means that grid draw with pixel scroll has ended.

Examples

1) How to use aftergriddraw:

```
flexmonster.on('aftergriddraw', function (e) {
 if (e.smooth == false)
     console.log('After grid draw with row/column scroll!');
    else
     console.log('After grid draw with pixel scroll!');
});
```

Open the example on JSFiddle (https://jsfiddle.net/flexmonster/hm2gt8v1/).

2) How to use aftergriddraw for styling rows based on the conditional formatting: live demo (http://jsfiddle.net/flexmonster/nonte3qv/).

See also

beforegriddraw (/api/beforegriddraw/)

4.4. beforegriddraw

[starting from version: 2.4]

It is triggered before grid rendering. To know when rendering is finished use aftergriddraw.

Data passed to the handler

params – Object. It has the following parameter:

- smooth – Boolean. There are two types of grid draw: grid draw that adds row/column scroll and grid draw that adds pixel scroll. As a result,
  beforegriddraw is called twice. The first time it is called with smooth: false and it means that grid draw with row/column scroll is starting. The second time beforegriddraw is called with smooth: true and it means that grid draw with pixel scroll is starting.

Examples

1) How to use beforegriddraw:

```
flexmonster.on('beforegriddraw', function (e) {
 if (e.smooth == false)
     console.log('Before grid draw with row/column scroll!');
    else
     console.log('Before grid draw with pixel scroll!');
});
```

Open the example on JSFiddle (https://jsfiddle.net/flexmonster/hm2gt8v1/).

2) How to use beforegriddraw for styling rows based on the conditional formatting: live demo (http://jsfiddle.net/flexmonster/nonte3qv/).

See also

aftergriddraw (/api/aftergriddraw/)

4.5. beforetoolbarcreated

[starting from version: 2.3]

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

- title – String. The tab's label.
- id – String. Id used in CSS styles.
- handler – Function. The function that handles clicks on this tab.
- icon – String. The HTML tag containing your custom icon for this new tab. You can choose one of the basic vector icons defined in the flexmonster.toolbar.js file.
- args (optional) – Any. Arguments to pass to the handler.
- menu (optional) – Array. Dropdown menu items.
- mobile (optional) – Boolean. When set to false, the tab does not show on mobile devices. *Default value: true*.
- ios (optional) – Boolean. When set to false, the tab does not show on iOS devices. *Default value: true*.

- android (optional) – Boolean. When set to false false, the tab does not show on Android devices. *Default value: true*.
- rightGroup (optional) – Boolean. When set to true, the tab is positioned on the right side of the Toolbar. *Default value: false*.

Example

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

```
var pivot = new Flexmonster({
    container: "pivot-container",
    toolbar: true,
    beforetoolbarcreated: customizeToolbar
});

function customizeToolbar(toolbar) {
    // override tabs
    toolbar.getTabs = function() {
        var tabs = [];
        tabs.push({
            id: "fm-tab-connect",
            icon: this.icons.connect,
            title: this.Labels.connect,
            handler: function() {
                this.pivot.connectTo({
                    type: "csv",
                    filename: "http://cdn.flexmonster.com/2.3/data/data.csv"
                });
            }
        });
        return tabs;
    }
}
```

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

See also

Customizing toolbar tutorial (/doc/customizing-toolbar/)

4.6. cellclick

[starting from version: 2.3]

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

Data passed to the handler

cell – Cell Data Object (/api/cell-object/). It 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

chartclick (/api/chartclick/)
on (/api/on/)
off (/api/off/)

4.7. celldoubleclick

[starting from version: 2.3]

It is triggered when a cell is double clicked on the grid. For CSV and JSON data, the event is also triggered for the drill-through view.

Data passed to the handler

cell – Cell Data Object (/api/cell-object/). It 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/).

4.8. chartclick

[starting from version: 2.7.17]

Triggered once any chart element is clicked.

Data passed to the handler

chart – Chart Data Object (https://www.flexmonster.com/api/chart-data-object/). It contains information about the clicked chart segment.

Example

```
flexmonster.on('chartclick', function(chart) {
    alert(
        "Click on chart segment with label: " +
        chart.label + ", measure: " +
        chart.measure.uniqueName +
        ", value: " +
        chart.value);
});
```

Open the example on JSFiddle (https://jsfiddle.net/flexmonster/cf3pb4ox/).

See also

cellclick (/api/cellclick/)
on (/api/on/)
off (/api/off/)

4.9. datachanged

[starting from version: 2.306]

It is triggered after the user edits data. Editing is available through the flat table or the drill-through pop-up. This event should be used when editing is enabled (editing: true). When datachanged is triggered it gets as input the object with data property. This property is an 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.
- oldValue – String. Old 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 Field 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 - field: "
            + e.data[0].field + ", id: "
            + e.data[0].id
            + ", new value: "
            + e.data[0].value
            + ", previous value: "
            + e.data[0].oldValue);
```

```
});
```

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

See also

Report Object (/api/report-object/)

4.10. dataerror

[starting from version: 2.3]

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

Data passed to the handler

params – Object. It has the following parameter:

- error – String. The error message from the response returned from the server-side.

Example

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

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

See also

loadingdata (/api/loadingdata/)
dataloaded (/api/dataloaded/)
olapstructureerror (/api/olapstructureerror/)

4.11. datafilecancelled

[starting from version: 2.3]

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

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

4.12. dataloaded

[starting from version: 2.3]

It is triggered when the component loaded data. To track possible errors, use dataerror (/api/dataerror/).

Note: use olapstructureloaded (/api/olapstructureloaded/) for "microsoft analysis services", "api", and "mondrian" data source types.

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/)

4.13. drillthroughclose

[starting from version: 2.8]

It is triggered when the drill-through view is closed. To know when the drill-through view is opened, use the drillthroughopen event.

Example

```
flexmonster.on('drillthroughclose', function() {
    alert("The drill-through view is closed!");
});
```

See also

drillthroughopen (https://www.flexmonster.com/api/drillthroughopen/)

4.14. drillthroughopen

[starting from version: 2.8]

It is triggered when the drill-through view is opened. To know when the drill-through view is closed, use the drillthroughclose event.

Data passed to the handler

- data – Cell Data Object | Chart Data Object. Information about the object opened in the drill-through view. If the drill-through view is opened on the grid, this is a Cell Data Object (https://www.flexmonster.com/api/cell-object/) containing information about the cell. If the drill-through view is opened on charts, this is a Chart Data Object (https://www.flexmonster.com/api/chart-data-object/) containing information about the chart segment.

Examples

1. Example with Cell Data Object passed to the handler:

```
flexmonster.on('drillthroughopen', function (cell) {
    alert("The drill-through view is opened for the cell - row: "
          + cell.rowIndex + ", column: "
          + cell.columnIndex
          + ", label: "
          + cell.label);
});
```

2. Example with Chart Data Object passed to the handler:

```
flexmonster.on('drillthroughopen', function(chart) {
    alert("The drill-
through view is opened for the chart segment with label: "
          + chart.label + ", measure: "
          + chart.measure.uniqueName
          + ", value: "
          + chart.value);
});
```

3. Choosing the handler's action depending on where the drill-through view is opened:

```
flexmonster.on('drillthroughopen', function(data) {
    if (flexmonster.getOptions().viewType == "grid"){
        alert("The drill-through view is opened for the cell - row: "
              + data.rowIndex + ", column: "
              + data.columnIndex
              + ", label: "
              + data.label);
    }
```

```
        else if (flexmonster.getOptions().viewType == "charts") {
            alert("The drill-
    through view is opened for the chart segment with label: "
                  + data.label + ", measure: "
                  + data.measure.uniqueName
                  + ", value: "
                  + data.value);
        }
    });
```

See also

drillthroughclose (https://www.flexmonster.com/api/drillthroughclose/)

4.15. exportcomplete

[starting from version: 2.3]

It is triggered after the export process has finished. 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/)

4.16. exportstart

[starting from version: 2.3]

It is triggered when the export of a grid or chart starts. Export is possible to CSV, HTML, PDF, Image, or Excel format. Use exportTo(type:String, params:Object, callbackHandler: Function) for more export options. To make sure that the 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/)

4.17. fieldslistclose

[starting from version: 2.3]

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

Example

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

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

See also

fieldslistopen (/api/fieldslistopen/)

4.18. fieldslistopen

[starting from version: 2.3]

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

Example

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

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

See also

fieldslistclose (/api/fieldslistclose/)

4.19. filterclose

[starting from version: 2.6.8]

It is triggered when the filter pop-up window is closed.

Example

```
flexmonster.on('filterclose', function () {
  alert('The filter is closed!');
});
```

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

See also

filteropen (/api/filteropen/)
openFilter (/api/openFilter/)

4.20. filteropen

[starting from version: 2.3]

It is triggered when the filter pop-up window is opened.

Data passed to the handler

params – Object. It contains the following parameters of the filter pop-up window:

- hierarchy – Object. It describes hierarchy.

Example

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

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

See also

reportcomplete (/api/reportcomplete/)
filterclose (/api/filterclose/)
openFilter (/api/openFilter/)

4.21. loadingdata

[starting from version: 2.3]

It is triggered when data starts loading from local or remote CSV, JSON, or after the report was loaded. To track possible errors, use dataerror (/api/dataerror/). To make sure that loading of the data was successful, use

dataloaded (/api/dataloaded/).

Note: use loadingolapstructure (/api/loadingolapstructure) for "microsoft analysis services", "api", and "mondrian" data source types.

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/)

4.22. loadinglocalization

[starting from version: 2.3]

It is triggered when a 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/)

4.23. loadingolapstructure

[starting from version: 2.3]

It is triggered for "microsoft analysis services", "api", and "mondrian" data source types.

For OLAP data sources, loadingolapstructure is triggered when the structure of an OLAP cube starts loading.

Once the connection was established, the component begins loading the OLAP structure.

For the custom data source API, the event is triggered when the component sends the request for the schema (https://www.flexmonster.com/api/fields-request/) to the server.

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/)

4.24. loadingreportfile

[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/)

4.25. localizationerror

[starting from version: 2.3]

It is triggered when some error appeared while loading a 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/)

4.26. localizationloaded

[starting from version: 2.3]

It is triggered when a 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/)

4.27. olapstructureerror

[starting from version: 2.3]

It is triggered when some error appeared while loading the 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/)
unauthorizederror (/api/unauthorizederror/)

4.28. olapstructureloaded

[starting from version: 2.3]

It is triggered for "microsoft analysis services", "api", and "mondrian" data source types.

For OLAP data sources, olapstructureloaded is triggered after the OLAP structure is loaded.

For the custom data source API, the event is triggered after the component receives the response to the /fields request (https://www.flexmonster.com/api/fields-request/).

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/)

4.29. openingreportfile

[starting from version: 2.3]

It is triggered when:

1. a 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/)

4.30. printcomplete

[starting from version: 2.3]

It is triggered after the 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 the 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/)

4.31. printstart

[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 the 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/)

4.32. querycomplete

[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/)

4.33. queryerror

[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/)

4.34. ready

[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 the 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 the report parameter was specified in new Flexmonster(). To retrieve information about data source structure use reportcomplete. If you want to track changes in a report object, we recommend using reportchange.

Example

How to define a report from JS:

Defining a report from JS after the component is created:

```
var pivot = new Flexmonster({
 container: "pivotContainer",
 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/new-flexmonster/)
reportcomplete (/api/reportcomplete/)

reportchange (/api/reportchange/)

4.35. reportchange

[starting from version: 2.3]

It is triggered when a report is changed in the component. The change can be performed by a user or programmatically. The list of API calls which trigger the reportchange event:

- expandAllData()
- expandData()
- collapseData()
- setFilter()
- clearFilter()
- sortValues()
- setSort()
- setFormat()
- runQuery()
- addCondition()
- removeCondition()
- removeAllConditions()
- addCalculatedMeasure()
- removeCalculatedMeasure()
- removeAllCalculatedMeasures()

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/)

4.36. reportcomplete

[starting from version: 2.3]

It is triggered when OLAP structure or data from the 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/)

4.37. reportfilecancelled

[starting from version: 2.3]

It is triggered when a customer uses the 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/)

4.38. reportfileerror

[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/)

4.39. reportfileloaded

[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/)

4.40. runningquery

[starting from version: 2.3]

It is triggered before a 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/)

4.41. unauthorizederror

[starting from version: 2.8.17]

It is triggered when the Accelerator or the custom data source API server sends the 401 Unauthorized (https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/401) error in response to a Flexmonster's request. Only for "api" and "microsoft analysis services" data source types.

Data passed to the handler

- callbackHandler – Function. It can be used to resend the request to the Accelerator or to the custom data source API server. callbackHandler accepts an Object that can have the following parameters:
    - requestHeaders – Object. New request headers can be specified here.

Example

unauthorizederror can be used to automatically refresh authorization headers and resend the request in case of the 401 Unauthorized error:

```
flexmonster.on('unauthorizederror', function (callbackHandler) {
    // other actions
    // pass new request headers and resend the request
    callbackHandler({
        requestHeaders: {
            AuthToken: "XXX"
        }
    });
});
```

See also

olapstructureerror (https://www.flexmonster.com/api/olapstructureerror/)
queryerror (https://www.flexmonster.com/api/queryerror/)
dataerror (https://www.flexmonster.com/api/dataerror/)

4.42. update

[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/)

5. Custom data source API

5.1. All requests

The Flexmonster custom data source API (https://www.flexmonster.com/doc/introduction-to-custom-data-source-api/) is a communication protocol that allows retrieving already aggregated data from a server to Flexmonster Pivot.

To fetch the data from the server, Flexmonster sends POST requests to the API endpoints in JSON format.

List of requests:

<url>/handshake (/api/handshake-request/)	the first request to establish a connection between the client and server sides
<url>/fields (/api/fields-request/)	request for all fields with their types
<url>/members (/api/members-request)	request for all members of the field
<url>/select (/api/select-request-for-pivot-table/)	request for data, can be for pivot table (/api/select-request-for-pivot-table/), flat table (/api/select-request-for-flat-table/), and drill-through view (/api/select-request-for-drill-through-view/)

5.2. /handshake request

[starting from version: 2.8]

An optional request to establish a connection between the client and server sides. Sends the version of the custom data source API implemented by the front end. Expects the version of the custom data source API implemented by the back end in return.

/handshake request allows checking whether the server and the client implement the same version of the custom data source API. If the implemented versions are incompatible, you will get an appropriate notification.

Note that you need to handle the /handshake request properly to receive compatibility notifications. Find the recommended way of implementing the /handshake request here (/doc/implement-custom-data-source-api/#!step4).

Request

```
{
    "type": "handshake"
    "version": string
}
```

The request has the following parameters:

- type – String. The type of the request. In this case, it is "handshake".
- version – String. The version of the custom data source API implemented by the front end.

Response

```
{
    "version": "2.8.0"
}
```

The response has the following parameters:

- version – String. The version of the custom data source API implemented by the back end.

See also

/fields request (https://www.flexmonster.com/api/fields-request/)
/members request (https://www.flexmonster.com/api/members-request/)
/select request for pivot table (https://www.flexmonster.com/api/select-request-for-pivot-table/)
/select request for flat table (https://www.flexmonster.com/api/select-request-for-flat-table/)
/select request for drill-through view (https://www.flexmonster.com/api/select-request-for-drill-through-view/)

5.3. /fields request

[starting from version: 2.8]

A request for all fields with their types (i.e., meta-object or schema).

Request

```
{
    "type": "fields"
    "index": string
}
```

The request has the following parameters:

- type — String. The type of the request. In this case, it is "fields".
- index – String. The dataset identifier.

Response

```
{
    "fields"[]: {
        "uniqueName": string,
        "type": "string" | "number" | "date",
        "caption": string,
        "hierarchy": string,
        "parent": string,
        "folder": string,
        "interval": string,
        "aggregations"[]: string,
        "filters": boolean | {
            "members": boolean,
            "query": boolean | string[],
            "valueQuery": boolean | string[]
        }
    },
    "aggregations": string[] | {
        "any"[]: string,
        "date"[]: string,
        "number"[]: string,
        "string"[]: string
    },
    "filters": boolean | {
        "any": boolean | {
            "members": boolean,
            "query": boolean | string[],
            "valueQuery": boolean | string[]
        },
        "date": boolean | {
            "members": boolean,
            "query": boolean | string[],
            "valueQuery": boolean | string[]
        },
        "number": boolean | {
            "members": boolean,
            "query": boolean | string[],
            "valueQuery": boolean | string[]
        },
        "string": boolean | {
            "members": boolean,
            "query": boolean | string[],
            "valueQuery": boolean | string[]
        },
        "advanced": boolean
    },
    "sorted": boolean
}
```

The response has the following parameters:

- fields – Array of objects. Contains the following information about fields:
  - uniqueName – String. The field's unique name. Will later be used in /members and /select requests.
  - type – String. The field's type. Supported values are: "string", "number", "date".
  - caption (optional) – String. The field's caption to appear on the UI.
  - hierarchy (optional) – String. The hierarchy's name. Used to configure multilevel hierarchies. Specify this property to set the field as a level of a hierarchy.
    See how to configure multilevel hierarchies (https://www.flexmonster.com/doc/supporting-multilevel-hierarchies/).
  - parent (optional) – String. The parent level's unique name. This property is necessary to specify if the field is a level of a hierarchy and has a parent level.
    See how to configure multilevel hierarchies (https://www.flexmonster.com/doc/supporting-multilevel-hierarchies/).
  - folder (optional) – String. The field's folder. Folders are used to organize groups of fields in the Field List. folder supports nesting via / (e.g., "Folder/Subfolder").
  - interval (optional) – String. A date's aggregation interval to group dates on the server. The component will automatically send it in /members and /select requests. Possible values depend on how the server handles date intervals. Only for fields of the "date" type.
  - aggregations (optional) – Array of strings. A list of supported aggregation functions for the field. To define supported aggregations for all fields, use the root aggregations property (#aggregations). Supported values include: "sum", "count", "distinctcount", "average", "median", "product", "min", "max", "stdevp", "stdevs", "none", or a custom aggregation (https://www.flexmonster.com/doc/support-more-aggregations/#!custom-aggs). Note: for the fields of the "number" type, Flexmonster Pivot supports built-in front-end aggregations (/doc/support-more-aggregations/#built-in-aggs).
  - filters (optional) – Boolean|Object. Supported filters for the field. To define supported filters for all fields, specify the root filters property. (#filters)
    Filters for the field can be enabled all at once by setting this property to true, or each filter type can be turned on separately:
    - members (optional) – Boolean. A configuration of an include/exclude members filter. If true, the members filter is enabled for the field in Flexmonster.
    - query (optional) – Boolean|Array of strings. A configuration of a conditional filter on members. To turn on this filter, either set query to true or specify an array of supported conditions.
      If the property is set to true, this enables all the conditions that exist in Flexmonster for the members filter. See the list of supported conditions for "string" (/doc/implementing-filters/#string), "number" (/doc/implementing-filters/#number), and "date" (/doc/implementing-filters/#date) field types.
    - valueQuery (optional) – Boolean|Array of strings. A configuration of a conditional filter on values. To turn on this filter, either set valueQuery to true or specify an array of supported conditions.
      If the property is set to true, this enables all the conditions that exist in Flexmonster for the values filter. See the list of supported conditions (/doc/implementing-filters/#values).
- aggregations (optional) – Array of strings|Object. Supported aggregation functions for all fields or fields of a certain type. Supported aggregations for a certain field can be defined in the fields.aggregations property (#field-aggregations).
  To define supported aggregations for all fields, specify aggregations as an array of strings. Supported values include: "sum", "count", "distinctcount", "average", "median", "product", "min", "max","stdevp", "stdevs", "none", or a custom aggregation (https://www.flexmonster.com/doc/support-more-aggregations/#!custom-aggs). Note: for the fields of the "number" type, Flexmonster Pivot supports built-in front-end aggregations (/doc/support-more-aggregations/#built-in-aggs).
  To define supported aggregations for certain field types, specify the aggregations property as an object with the following properties:

    - any (optional) – Array of strings. Supported aggregation functions for any field type.
    - date (optional) – Array of strings. Supported aggregation functions for the "date" field type.
    - number (optional) – Array of strings. Supported aggregation functions for the "number" field type. Note: for the fields of this type, Flexmonster Pivot supports built-in front-end aggregations (/doc/support-more-aggregations/#built-in-aggs).
    - string (optional) – Array of strings. Supported aggregation functions for the "string" field type.
- filters (optional) – Boolean|Object. Supported filters for all fields or for fields of a certain type. To define filters for a specific field, use the fields.filters property (#field-filters).
  The filters for all fields are turned off by default. They can be turned on all at once by setting this property to true, or each filter type can be configured separately for fields of a certain type:
    - any (optional) – Boolean|Object. Supported filters for any field type.
        - members (optional) – Boolean. A configuration of an include/exclude members filter. If true, the members filter is enabled in Flexmonster.
        - query (optional) – Boolean|Array of strings. A configuration of a conditional filter on members. To turn on this filter, either set query to true or specify an array of supported conditions.
          If the property is set to true, this enables all the conditions that exist in Flexmonster for the members filter. There are 4 common conditions for all field types that can be specified here: "equal", "not_equal", "between", "not_between".
        - valueQuery (optional) – Boolean|Array of strings. A configuration of a conditional filter on values. To turn on this filter, either set valueQuery to true or specify an array of supported conditions.
          If the property is set to true, this enables all the conditions that exist in Flexmonster for the values filter. See the list of supported conditions (/doc/implementing-filters/#values).
    - date (optional) – Boolean|Object. Supported filters for the "date" field type.
        - members (optional) – Boolean. A configuration of an include/exclude members filter. If true, the members filter is enabled in Flexmonster.
        - query (optional) – Boolean|Array of strings. A configuration of a conditional filter on members. To turn on this filter, either set query to true or specify an array of supported conditions.
          If the property is set to true, this enables all the conditions that exist in Flexmonster for the members filter. See the list of supported conditions for the “date” field type (/doc/implementing-filters/#date).
        - valueQuery (optional) – Boolean|Array of strings. A configuration of a conditional filter on values. To turn on this filter, either set valueQuery to true or specify an array of supported conditions.
          If the property is set to true, this enables all the conditions that exist in Flexmonster for the values filter. See the list of supported conditions (/doc/implementing-filters/#values).
    - number (optional) – Boolean|Object. Supported filters for the "number" field type.
        - members (optional) – Boolean. A configuration of an include/exclude members filter. If true, the members filter is enabled in Flexmonster.
        - query (optional) – Boolean|Array of strings. A configuration of a conditional filter on members. To turn on this filter, either set query to true or specify an array of supported conditions.
          If the property is set to true, this enables all the conditions that exist in Flexmonster for the members filter. See the list of supported conditions for the “number” field type (/doc/implementing-filters/#number).
        - valueQuery (optional) – Boolean|Array of strings. A configuration of a conditional filter on values. To turn on this filter, either set valueQuery to true or specify an array of supported conditions.
          If the property is set to true, this enables all the conditions that exist in Flexmonster for the values filter. See the list of supported conditions (/doc/implementing-filters/#values).
    - string (optional) – Boolean|Object. Supported filters for the "string" field type.
        - members (optional) – Boolean. A configuration of an include/exclude members filter. If true, the members filter is enabled in Flexmonster.

        - query (optional) – Boolean|Array of strings. A configuration of a conditional filter on members. To turn on this filter, either set query to true or specify an array of supported conditions.
          If the property is set to true, this enables all the conditions that exist in Flexmonster for the members filter. See the list of supported conditions for the “string” field type (/doc/implementing-filters/#string).
        - valueQuery (optional) – Boolean|Array of strings. A configuration of a conditional filter on values. To turn on this filter, either set valueQuery to true or specify an array of supported conditions.
          If the property is set to true, this enables all the conditions that exist in Flexmonster for the values filter. See the list of supported conditions (/doc/implementing-filters/#values).
    - advanced – Boolean. Indicates whether the server has advanced hierarchical filters implemented. When the advanced parameter is set to true, it means that the server supports hierarchies and can filter them, so multilevel hierarchies can be configured in the component. When the parameter is false, the server cannot filter the hierarchical data. In this case, if multilevel hierarchies were configured in the Mapping Object (/api/mapping-object/), these configurations will be ignored. *Default value: false.*
- sorted (optional) – Boolean. If true, the fields’ order from the response will be used to display fields in the Field List.

See also

/handshake request (https://www.flexmonster.com/api/handshake-request/)
/members request (https://www.flexmonster.com/api/members-request/)
/select request for pivot table (https://www.flexmonster.com/api/select-request-for-pivot-table/)
/select request for flat table (https://www.flexmonster.com/api/select-request-for-flat-table/)
/select request for drill-through view (https://www.flexmonster.com/api/select-request-for-drill-through-view/)

5.4. /members request

[starting from version: 2.8]

A request for all members of the field.

Request

```
{
    "type": "members",
    "index": string,
    "field": FieldObject,
    "filter": FilterObject[] | FilterGroupObject,
    "page": number
}
```

The request has the following parameters:

- type — String. The type of the request. In this case, it is "members".
- index – String. The dataset identifier.
- field – Field Object (/api/field-object/). The field which members should be sent in the response.
- filter – Array of Filter Objects (/api/filter-object-for-requests/)|Filter Group Object (https://www.flexmonster.com/api/filter-group-object/). Filters that should be applied to members. filter is

present in the request when multilevel hierarchies are configured in Flexmonster Pivot.
The filter can be:
    - An array of Filter Objects (https://www.flexmonster.com/api/filter-object-for-requests/) – when the version sent in the /handshake response (https://www.flexmonster.com/api/handshake-request/#response) is 2.8.5 or /handshake is not implemented. Only include and exclude filtering parameters of the Filter Object (https://www.flexmonster.com/api/filter-object-for-requests/) are used.
    - The Filter Group Object (https://www.flexmonster.com/api/filter-group-object/) – when the version sent in the /handshake response (https://www.flexmonster.com/api/handshake-request/#response) is 2.8.22 or later.
- page – Number. The page number. It can be used to load members by parts. If the response contains pageTotal parameter, additional requests will be performed to load the remaining pages. Starts from 0.

Response

```
{
    "members"[]: {
        "value": string | number,
        "id": string
    },
    "sorted": boolean,
    "page": number,
    "pageTotal": number
}
```

The response has the following parameters:

- members – Array of objects. It contains all the members. Each member has the following properties:
    - value – String|Number. The member's value. In the case of a number field, it should be of type number. In the case of a date field, the members should be passed to Flexmonster in the Unix timestamp (https://www.unixtimestamp.com/) format, then dates are recognized correctly. For example, the date "2016-02-07" will be "1454803200" in the form of a Unix timestamp.
    - id (optional) – String. The member's id. Supported only for string fields. If defined, it is used in queries and in responses to identify the member.
- sorted (optional) – Boolean. If true, the members' order from the response will be used as AZ order on the UI.
- page (optional) – Number. The current page number. Starts from 0.
- pageTotal (optional) – Number. The total number of pages. It can be used to load members by parts.

Examples

1. Example with a string field
   Request:

   ```
   {
       "index": "data-set-123",
       "type": "members",
       "field": {
           "uniqueName": "city"
       },
       "page": 0
   }
   ```

Response:

```
{
    "members": [
        { "value": "Toronto" },
        { "value": "Montreal" },
        { "value": "New York" }
    ]
}
```

2. Example with a number field
   Request:

```
{
    "index": "data-set-123",
    "type": "members",
    "field": {
        "uniqueName": "price"
    },
    "page": 0
}
```

Response:

```
{
    "members": [
        { "value": 10 },
        { "value": 28 },
        { "value": 30 }
    ]
}
```

3. Example with a date field
   Request:

```
{
    "index": "data-set-123",
    "type": "members",
    "field": {
        "uniqueName": "order_date"
    },
    "page": 0
}
```

Response:

```
{
    "members": [
        { "value": 1562889600000 },
```

```
            { "value": 1564617600000 },
            { "value": 1564963200000 }
        ]
    }
```

See also

/handshake request (https://www.flexmonster.com/api/handshake-request/)
/fields request (https://www.flexmonster.com/api/fields-request/)
/select request for pivot table (https://www.flexmonster.com/api/select-request-for-pivot-table/)
/select request for flat table (https://www.flexmonster.com/api/select-request-for-flat-table/)
/select request for drill-through view (https://www.flexmonster.com/api/select-request-for-drill-through-view/)

5.5. /select request for the pivot table

[starting from version: 2.8]

A request for data.

Request

```
{
    "type": "select"
    "index": string,
    "query": {
        "aggs": {
            "values"[]: {
                "field": FieldObject,
                "func": string
            },
            "by": {
                "rows"[]:FieldObject,
                "cols"[]: FieldObject
            }
        },
        "filter": FilterObject[] | FilterGroupObject
    },
    "page": number
}
```

The request has the following parameters:

- type — String. The type of the request. In this case, it is "select".
- index – String. The dataset identifier.
- query – Object. A query object. Contains the following properties:
    - aggs – Object. Query aggregations. The part of the query that specifies which data should be aggregated and how. Contains the following properties:
        - values – Array of objects. Values to aggregate. Fields with at least one supported aggregation defined in the schema can be selected for the query as values. Each object in

the array has the following properties:
- field – Field Object (/api/field-object/). The field selected as a measure.
- func – String. The aggregation function name. For each field, the list of supported aggregations is defined in the response to the /fields request (https://www.flexmonster.com/api/fields-request/). Supported values may include: "sum", "count", "distinctcount", "average", "median", "product", "min", "max", "stdevp", "stdevs", "none", or a custom aggregation (https://www.flexmonster.com/doc/support-more-aggregations/#!custom-aggs). Note: for the fields of the "number" type, Flexmonster Pivot supports built-in front-end aggregations (/doc/support-more-aggregations/#built-in-aggs).

- by – Object. Fields by which the data should be aggregated:
  - rows – Array of Field Objects (/api/field-object/). Fields in rows.
  - cols – Array of Field Objects (/api/field-object/). Fields in columns.

- filter (optional) – Array of Filter Objects (https://www.flexmonster.com/api/filter-object-for-requests/)|Filter Group Object (https://www.flexmonster.com/api/filter-group-object/). Query filters. The part of a query that specifies which filters should be applied to the data.
  If the server does not support multilevel hierarchies (i.e., the filters.advanced property (https://www.flexmonster.com/api/fields-request/#advanced) is set to false), the filter's structure is an array of Filter Objects (https://www.flexmonster.com/api/filter-object-for-requests/).
  If multilevel hierarchies are supported, the filter can be:
  - An array of Filter Objects (https://www.flexmonster.com/api/filter-object-for-requests/) – when the version sent in the /handshake response (https://www.flexmonster.com/api/handshake-request/#response) is 2.8.5 or /handshake is not implemented.
  - The Filter Group Object (https://www.flexmonster.com/api/filter-group-object/) – when the version sent in the /handshake response (https://www.flexmonster.com/api/handshake-request/#response) is 2.8.22 or later.
- page – Number. The page number. It can be used to load data by parts. If a response contains the pageTotal parameter, additional requests will be performed to load the remaining pages. Starts from 0.

Response

```
{
    "aggs"[]: {
        "values": {
            (uniqueName): {
                (func): number
            }
        },
        "keys": {
            (uniqueName): string | number
        }
    },
    "page": number,
    "pageTotal": number
}
```

The response has the following parameters:

- aggs – Array of objects. Aggregated data. Each object in the array has the following properties:
  - values – Object. Numeric values that are calculated for a specific tuple.
    - (uniqueName) – Object. The field's unique name.

        - (func) – Number. The result of the calculation, where (func) is an aggregation function.
    - keys (optional) – Object. Field's keys that describe a specific tuple. In case it is not defined, values are treated as totals.
        - (uniqueName) – String|Number. Field's member, where (uniqueName) is a field's unique name. Note: totals for each field should be included in the response even if they are disabled on the client side.
- page (optional) – Number. The current page number. Starts from 0.
- pageTotal (optional) – Number. The total number of pages. It can be used to load members by parts.

Examples

1. Example with one value
   Request:

```
{
    "index": "data-set-123",
    "type": "select",
    "query": {
        "aggs": {
            "values": [{
                "func": "sum",
                "field": {
                    "uniqueName": "price"
                }
            }]
        }
    },
    "page": 0
}
```

   Response:

```
{
    "aggs": [{
        "values": {
            "price": {
                "sum": 123
            }
        }
    }]
}
```

2. Example with two values
   Request:

```
{
    "index": "data-set-123",
    "type": "select",
    "query": {
        "aggs": {
            "values": [{
                "func": "sum",
                "field": {
```

```
                    "uniqueName": "price"
                }
            }, {
                "func": "sum",
                "field": {
                    "uniqueName": "quantity"
                }
            }]
        }
    },
    "page": 0
}
```

Response:

```
{
    "aggs": [{
        "values": {
            "price": {
                "sum": 123
            },
            "quantity": {
                "sum": 5
            }
        }
    }]
}
```

3. Example with a field in rows
   Request:

```
{
    "index": "data-set-123",
    "type": "select",
    "query": {
        "aggs": {
            "values": [{
                "func": "sum",
                "field": {
                    "uniqueName": "price"
                }
            }],
            "by": {
                "rows": [{
                    "uniqueName": "city"
                }]
            }
        }
    },
    "page": 0
}
```

Response:

```
{
    "aggs": [{
        "values": {
            "price": {
                "sum": 123
            }
        }
    }, {
        "keys": {
            "city": "Toronto"
        },
        "values": {
            "price": {
                "sum": 100
            }
        }
    }, {
        "keys": {
            "city": "New York"
        },
        "values": {
            "price": {
                "sum": 23
            }
        }
    }]
}
```

4. Example with fields in rows and columns
   Request:

```
{
    "index": "data-set-123",
    "type": "select",
    "query": {
        "aggs": {
            "values": [{
                "func": "sum",
                "field": {
                    "uniqueName": "price"
                }
            }],
            "by": {
                "rows": [{
                    "uniqueName": "city"
                }],
                "cols": [{
                    "uniqueName": "color"
                }]
            }
        }
    },
```

```
    "page": 0
}
```

Response:

```
{
    "aggs": [{
        "values": {
            "price": {
                "sum": 48
            }
        }
    }, {
        "keys": {
            "city": "New York"
        },
        "values": {
            "price": {
                "sum": 20
            }
        }
    }, {
        "keys": {
            "city": "Toronto"
        },
        "values": {
            "price": {
                "sum": 28
            }
        }
    }, {
        "keys": {
            "color": "blue"
        },
        "values": {
            "price": {
                "sum": 38
            }
        }
    }, {
        "keys": {
            "color": "red"
        },
        "values": {
            "price": {
                "sum": 10
            }
        }
    }, {
        "keys": {
            "city": "New York",
            "color": "blue"
        },
        "values": {
```

```
            "price": {
                "sum": 20
            }
        }
    }, {
        "keys": {
            "city": "Toronto",
            "color": "blue"
        },
        "values": {
            "price": {
                "sum": 18
            }
        }
    }, {
        "keys": {
            "city": "Toronto",
            "color": "red"
        },
        "values": {
            "price": {
                "sum": 10
            }
        }
    }]
}
```

5. Example with exclude members filter
   Request:

```
{
    "index": "data-set-123",
    "type": "select",
    "query": {
        "filter": [{
            "field": {
                "uniqueName": "city"
            },
            "exclude": [
                {
                    "member": "New York"
                },
                {
                    "member": "Montreal"
                }
            ]
        }],
        "aggs": {
            "values": [{
                "func": "sum",
                "field": {
                    "uniqueName": "price"
                }
            }],
```

```
            "by": {
                "rows": [{
                    "uniqueName": "city"
                }]
            }
        }
    }
}
```

Response:
Format is the same as above.

6. Example with include/exclude members filter on several fields
   Request:

```
{
    "index": "data-set-123",
    "type": "select",
    "query": {
        "filter": [{
                "field": {
                    "uniqueName": "color"
                },
                "include": [
                    {
                        "member": "blue"
                    }
                ]
            },
            {
                "field": {
                    "uniqueName": "city"
                },
                "exclude": [
                    {
                        "member": "New York"
                    },
                    {
                        "member": "Montreal"
                    }
                ]
        }],
        "aggs": {
            "values": [{
                "func": "sum",
                "field": {
                    "uniqueName": "price"
                }
            }],
            "by": {
                "rows": [{
                    "uniqueName": "city"
                }]
            }
        }
    }
```

```
}
```

Response:
Format is the same as above.

7. Example with a conditional filter on members for string field
   Request:

```
{
    "index": "data-set-123",
    "type": "select",
    "query": {
        "filter": [{
            "field": {
                "uniqueName": "city"
            },
            "query": {
                "begin": "toro"
            }
        }],
        "aggs": {
            "values": [{
                "func": "sum",
                "field": {
                    "uniqueName": "price"
                }
            }],
            "by": {
                "rows": [{
                    "uniqueName": "city"
                }]
            }
        }
    }
}
```

Response:
Format is the same as above.

8. Example with a conditional filter on members for number field
   Request:

```
{
    "index": "data-set-123",
    "type": "select",
    "query": {
        "filter": [{
            "field": {
                "uniqueName": "quantity"
            },
            "query": {
                "greater": 2
            }
        }],
        "aggs": {
```

```
            "values": [{
                "func": "sum",
                "field": {
                    "uniqueName": "price"
                }
            }],
            "by": {
                "rows": [{
                    "uniqueName": "quantity"
                }]
            }
        }
    }
}
```

Response:
Format is the same as above.

9. Example with a conditional filter on dates for date field
   Request:

```
{
    "index": "data-set-123",
    "type": "select",
    "query": {
        "filter": [{
            "field": {
                "field": "order_date"
            },
            "query": {
                "between": [1564610400000, 1564696799999]
            }
        }],
        "aggs": {
            "values": [{
                "func": "sum",
                "field": {
                    "field": "price"
                }
            }],
            "by": {
                "rows": [{
                    "field": "order_date"
                }]
            }
        }
    }
 }
```

Response:
Format is the same as above.

10. Example with a conditional filter on values
    Request:

```
{
    "index": "data-set-123",
    "type": "select",
    "query": {
        "filter": [{
            "field": {
                "uniqueName": "city"
            },
            "query": {
                "top": 3
            },
            "value": {
                "func": "sum",
                "field": {
                    "uniqueName": "price"
                }
            }
        }],
        "aggs": {
            "values": [{
                "func": "sum",
                "field": {
                    "uniqueName": "price"
                }
            }],
            "by": {
                "rows": [{
                    "uniqueName": "city"
                }]
            }
        }
    }
}
```

Response:
Format is the same as above.

See also

/handshake request (https://www.flexmonster.com/api/handshake-request/)
/fields request (https://www.flexmonster.com/api/fields-request/)
/members request (https://www.flexmonster.com/api/members-request/)
/select request for flat table (https://www.flexmonster.com/api/select-request-for-flat-table/)
/select request for drill-through view (https://www.flexmonster.com/api/select-request-for-drill-through-view/)

5.6. /select request for the flat table

[starting from version: 2.8]

A request for data.

Request

```
{
    "type": "select"
    "index": string,
    "query": {
        "fields"[]: FieldObject,
        "filter": FilterObject[] | FilterGroupObject,
        "aggs": {
            "values"[]: {
                "field": FieldObject,
                "func": string
            }
        }
    },
    "page": number
}
```

The request has the following parameters:

- type — String. The type of the request. In this case, it is "select".
- index – String. The dataset identifier.
- query – Object. A query object. Contains the following properties:
    - fields – Array of Field Objects (/api/field-object/). An array of fields (columns) to include in the response.
    - filter (optional) – Array of Filter Objects (https://www.flexmonster.com/api/filter-object-for-requests/)|Filter Group Object (https://www.flexmonster.com/api/filter-group-object/). Query filters. The part of a query that specifies which filters should be applied to the data.
      If the server does not support multilevel hierarchies (i.e., the filters.advanced property (https://www.flexmonster.com/api/fields-request/#advanced) is set to false), the filter's structure is an array of Filter Objects (https://www.flexmonster.com/api/filter-object-for-requests/).
      If multilevel hierarchies are supported, the filter can be:
        - An array of Filter Objects (https://www.flexmonster.com/api/filter-object-for-requests/) – when the version sent in the /handshake response (https://www.flexmonster.com/api/handshake-request/#response) is 2.8.5 or /handshake is not implemented.
        - The Filter Group Object (https://www.flexmonster.com/api/filter-group-object/) – when the version sent in the /handshake response (https://www.flexmonster.com/api/handshake-request/#response) is 2.8.22 or later.
    - aggs (optional) – Object. Query column totals. Contains the following properties:
        - values – Array of objects. Columns to aggregate totals for.
          Fields with at least one supported aggregation defined in the schema can be selected for this part of the query. Each object in the array has the following properties:
            - field – Field Object (/api/field-object/). The field selected as a measure.
            - func – String. The aggregation function name. For each field, the list of supported aggregations is defined in the response to the /fields request (https://www.flexmonster.com/api/fields-request/). Supported values may include: "sum", "count", "distinctcount", "average", "median", "product", "min", "max", "stdevp", "stdevs", "none", or a custom aggregation (https://www.flexmonster.com/doc/support-more-aggregations/#!custom-aggs).
              Note: for the fields of the "number" type, Flexmonster Pivot supports built-in front-end aggregations (https://www.flexmonster.com/doc/support-more-aggregations/#built-in-aggs).
- page – Number. The page number. It can be used to load data by parts. If the response contains the

pageTotal parameter, additional requests will be performed to load the remaining pages. Starts from 0.

Response

```
{
    "fields"[]: {
        "uniqueName": string
    },
    "hits"[]: [
        (index): string | number
    ],
    "aggs"[]: {
        "values": {
            (uniqueName): {
                (func): number
            }
        }
    },
    "page": number,
    "pageTotal": number
}
```

The response has the following parameters:

- fields – Array of objects. Fields (columns) included in the response. Each object in the array has the following properties:
  - uniqueName – String. The field's unique name.
- hits – Array. A two-dimensional array containing data:
  - (index) – String | Number. A data row, where (index) is a field (column) index.
- aggs (optional) – Array of objects. Column totals. Each object in the array has the following properties:
  - values – Object. Numeric values that are calculated for column totals. Contains the following properties:
    - (uniqueName) – Object. The field's unique name.
      - (func) – Number. The result of the calculation, where (func) is the aggregation function.
- page (optional) – Number. The current page number. Starts from 0.
- pageTotal (optional) – Number. The total number of pages. It can be used to load members by parts.

Example

Request:

```
{
    "index": "data-set-123",
    "type": "select",
    "query": {
        "fields": [
            {
                "uniqueName": "country"
            },
            {
                "uniqueName": "city"
```

```
            },
            {
                "uniqueName": "price"
            },
            {
                "uniqueName": "quantity"
            }
        ],
        "aggs": {
            "values": [{
                "func": "sum",
                "field": {
                    "uniqueName": "price"
                }
            }, {
                "func": "sum",
                "field": {
                    "uniqueName": "quantity"
                }
            }]
        }
    },
    "page": 0
}
```

Response:

```
{
    "fields": [
        { "uniqueName": "country" },
        { "uniqueName": "city" },
        { "uniqueName": "price" },
        { "uniqueName": "quantity" }
    ],
    "hits": [
        ["Canada", "Toronto", 53, 2],
        ["...", "...", 1, 1]
    ],
    "aggs": [{
        "values": {
            "price": {
                "sum": 123
            },
            "quantity": {
                "sum": 5
            }
        }
    }]
}
```

See also

/handshake request (/api/handshake-request/)
/fields request (/api/fields-request/)
/members request (/api/members-request/)
/select request for pivot table (/api/select-request-for-pivot-table/)
/select request for drill-through view (/api/select-request-for-drill-through-view/)

5.7. /select request for the drill-through view

[starting from version: 2.8]

A request for data.

Request

```
{
    "type": "select"
    "index": string,
    "query": {
        "fields"[]: FieldObject,
        "filter": FilterObject[] | FilterGroupObject,
        "limit": number
    },
    "page": number
}
```

The request has the following parameters:

- type — String. The type of the request. In this case, it is "select".
- index – String. The dataset identifier.
- query – Object. A query object. Contains the following properties:
    - fields – Array of Field Objects (/api/field-object/). An array of fields (columns) to include in the response.
    - filter (optional) – Array of Filter Objects (https://www.flexmonster.com/api/filter-object-for-requests/)|Filter Group Object (https://www.flexmonster.com/api/filter-group-object/). Query filters. The part of a query that specifies which filters should be applied to the data.
      If the server does not support multilevel hierarchies (i.e., the filters.advanced property (https://www.flexmonster.com/api/fields-request/#advanced) is set to false), the filter's structure is an array of Filter Objects (https://www.flexmonster.com/api/filter-object-for-requests/).
      If multilevel hierarchies are supported, the filter can be:
        - An array of Filter Objects (https://www.flexmonster.com/api/filter-object-for-requests/) – when the version sent in the /handshake response (https://www.flexmonster.com/api/handshake-request/#response) is 2.8.5 or /handshake is not implemented.
        - The Filter Group Object (https://www.flexmonster.com/api/filter-group-object/) – when the version sent in the /handshake response (https://www.flexmonster.com/api/handshake-request/#response) is 2.8.22 or later.
    - limit – Number. The maximum number of records that should be included in the response. Configurable on the client. *Default value: 1000*.
- page – Number. The page number. It can be used to load data by parts. If the response contains the

pageTotal parameter, additional requests will be performed to load the remaining pages. Starts from 0.

Response

```
{
    "fields"[]: {
        "uniqueName": string
    },
    "hits"[]: [
        (index): string | number
    ],
    "page": number,
    "pageTotal": number
}
```

The response has the following parameters:

- fields – Array of objects. Fields (columns) included in the response. Each object in the array has the following properties:
    - uniqueName – String. The field's unique name.
- hits – Array. A two-dimensional array containing data:
    - (index) – String | Number. A data row, where (index) is a field (column) index.
- page (optional) – Number. The current page number. Starts from 0.
- pageTotal (optional) – Number. The total number of pages. It can be used to load members by parts.

Example

Request:

```
{
    "index": "data-set-123",
    "type": "select",
    "query": {
        "fields": [
            {
                "uniqueName": "country"
            },
            {
                "uniqueName": "city"
            },
            {
                "uniqueName": "price"
            },
            {
                "uniqueName": "quantity"
            }
        ],
        "filter"[]: {
            "field": {
                "uniqueName": "country"
            },
            "include": [
```

```
                {
                    "member": "Canada"
                }
            ]
        },
        "limit": 1000
    },
    "page": 0
}
```

Response:

```
{
    "fields": [
        { "uniqueName": "country" },
        { "uniqueName": "city" },
        { "uniqueName": "price" },
        { "uniqueName": "quantity" }
    ],
    "hits": [
        ["Canada", "Toronto", 53, 2],
        ["Canada", "...", 1, 1]
    ]
}
```

See also

/handshake request (https://www.flexmonster.com/api/handshake-request/)
/fields request (https://www.flexmonster.com/api/fields-request/)
/members request (https://www.flexmonster.com/api/members-request/)
/select request for pivot table (https://www.flexmonster.com/api/select-request-for-pivot-table/)
/select request for flat table (https://www.flexmonster.com/api/select-request-for-flat-table/)
Field Object (https://www.flexmonster.com/api/field-object/)

5.8. Field Object

An object widely used in custom data source API requests. Represents a field with its properties.

Field Object has the following parameters:

- uniqueName – String. The field’s unique name.
- interval (optional) – String. A date’s aggregation interval to group dates on the server. The component will automatically send it in /members and /select requests. Possible values depend on how the server handles date intervals. Only for fields of the "date" type.

Example

An example of the request with the Field Object:

```
{
    "uniqueName": string,
    "interval": string
}
```

See also

/handshake request (https://www.flexmonster.com/api/handshake-request/)
/fields request (https://www.flexmonster.com/api/fields-request/)
/members request (https://www.flexmonster.com/api/members-request/)
/select request for pivot table (https://www.flexmonster.com/api/select-request-for-pivot-table/)
/select request for flat table (https://www.flexmonster.com/api/select-request-for-flat-table/)
/select request for drill-through view (https://www.flexmonster.com/api/select-request-for-drill-through-view/)

5.9. Filter Object

An object used in the custom data source API requests. It contains a query filter.

The Filter Object has the following parameters:

- field – Field Object (/api/field-object/). The field to apply the filter to.
- include (optional) – Array of objects. Field members to include. Each object has the following parameters:
    - member – String | Number. The field's member. For string field type it is string. For number and date field types it is number.
    - filter (optional) – Filter Object (/api/filter-object-for-requests/). Allows filtering multilevel hierarchies. If the member is a parent level of the hierarchy, specify filter to filter the lower levels of the hierarchy. Otherwise, the filter property is not necessary.
- exclude (optional) – Array of objects. Field members to exclude. Each object has the following parameters:
    - member – String | Number. The field's member. For string field type it is string. For number and date field types it is number.
    - filter (optional) – Filter Object (/api/filter-object-for-requests/). Allows filtering multilevel hierarchies. If the member is a parent level of the hierarchy, specify filter to filter the lower levels of the hierarchy. Otherwise, the filter property is not necessary.
- query (optional) – Object. A conditional filter.
    - (condition) – String|Number|Array of strings| Array of numbers. Value for the condition, where condition to apply is (condition).
- value (optional) – Object. The value to which a conditional filter is applied. Contains the following properties:
    - field – Field Object (https://www.flexmonster.com/api/field-object/). The value by which the data should be filtered.
    - func – String. The aggregation function name. For each field, the list of supported aggregations is defined in the response to the /fields request (https://www.flexmonster.com/api/fields-request/). Supported values may include: "sum", "count", "distinctcount", "average", "median", "product", "min", "max", "stdevp", "stdevs", "none". Note: for the fields of the "number" type, Flexmonster Pivot supports built-in front-end aggregations (/doc/support-more-aggregations/#built-in-aggs).

Examples

Example of the request with the Filter Object:

```
[{
```

```
    "field": FieldObject,
    "include"[]: {
        "member": string | number,
        "filter": FilterObject
    },
    "exclude"[]: {
        "member": string | number,
        "filter": FilterObject
    },
    "query": {
        (condition): string | number | string[] | number[]
    },
    "value": {
        "field": FieldObject,
        "func": string
    }
}]
```

See also

/handshake request (/api/handshake-request/)
/fields request (/api/fields-request/)
/members request (/api/members-request/)
/select request for pivot table (/api/select-request-for-pivot-table/)
/select request for flat table (/api/select-request-for-flat-table/)
/select request for drill-through view (/api/select-request-for-drill-through-view/)

5.10. Filter Group Object

The Filter Group Object describes filters for hierarchical data. It appears in /members (https://www.flexmonster.com/api/members-request/) and /select (https://www.flexmonster.com/api/select-request-for-pivot-table/) requests when the server supports multilevel hierarchies.

Refer to our guide to learn more about supporting multilevel hierarchies (https://www.flexmonster.com/doc/supporting-multilevel-hierarchies/).

The Filter Group Object has the following parameters:

- type – String. The filter's type. Possible values: "and", "or".
- value – Array of objects. Filters to apply to the data. Each element in the array can be either the Filter Object (https://www.flexmonster.com/api/filter-object-for-requests/) or the Filter Group Object (https://www.flexmonster.com/api/filter-group-object/). Filters are combined using the operator specified in the type property.

Examples

The code below demonstrates the request with the Filter Group Object. Notice that the filter.value array contains two objects: the first one is a Filter Group Object (https://www.flexmonster.com/api/filter-group-object/), and the second one is a Filter Object (https://www.flexmonster.com/api/filter-object-for-requests/).

```
{
    "type": "and",
    "value": [
        {
            "type": "or",
             "value": [
                {
                    "field": {
                        "uniqueName": "country"
                    },
                    "query": {
                        "begin": "c"
                    }
                },
                {
                    "field": {
                        "uniqueName": "state"
                    },
                    "query": {
                        "begin": "c"
                    }
                }
            ]
        },
        {
            "field": {
                "uniqueName": "W"
            },
            "query": {
                "begin": "m"
            }
        }
    ]
}
```

See also

Supporting multilevel hierarchies (https://www.flexmonster.com/doc/supporting-multilevel-hierarchies/)
Filter Object (https://www.flexmonster.com/api/filter-object-for-requests/)
/members request (https://www.flexmonster.com/api/members-request/)
/select request for pivot table (https://www.flexmonster.com/api/select-request-for-pivot-table/)
/select request for flat table (https://www.flexmonster.com/api/select-request-for-flat-table/)
/select request for drill-through view (https://www.flexmonster.com/api/select-request-for-drill-through-view/)

6. MongoDB Connector API

6.1. All methods

Flexmonster Connector for MongoDB (/doc/mongodb-connector/) is a special server-side tool allowing you to retrieve the data from a MongoDB database in a fast and comfortable way.

The Connector has three methods for handling Flexmonster's requests for data and retrieving the needed data from MongoDB.

List of Connector's methods:

getSchema (/api/getschema/)	allows getting the list of all fields with their types from a MongoDB database
getMembers (/api/getmembers-mongo/)	allows getting all members of the field from a MongoDB database
getSelectResult (/api/getselectresult)	allows getting the data from a MongoDB database

6.2. getSchema

getShema(mongoDBInstance: Db instance, index: String)

[starting from version: 2.8]

This API call allows getting the list of all fields with their types from a MongoDB database. Used in the handler of the Flexmonster /fields request. The getShema method is a part of the Flexmonster MongoDB Connector API.

Parameters:

- mongoDBInstance – Db instance (https://mongodb.github.io/node-mongodb-native/api-generated/db.html). The instance of the needed MongoDB database.
- index – String. The collection's name. index is sent in the body of the Flexmonster request.

Returns

Array of field objects, which contains all the fields and information about them. The response format is the same as in the /fields request (https://www.flexmonster.com/api/fields-request/).

Example

```
mongo.post("/fields", async (req, res) => {
    try {
        const result = await MongoDataAPI.getSchema(mongoDBInstance, req.body.index)
;
        res.json(result.toJSON());
    } catch (err) { }
});
```

See also

getMembers (/api/getmembers-mongo/)
getSelectResult (/api/getselectresult)

6.3. getMembers

getMembers(mongoDBInstance: Db instance, index: String, fieldObject: Field Object, page: Object)

[starting from version: 2.8]

This API call allows getting all members of the field from a MongoDB database. Used in the handler of the Flexmonster /members request. The getMembers method is a part of the Flexmonster MongoDB Connector API.

Parameters:

- mongoDBInstance – Db instance (https://mongodb.github.io/node-mongodb-native/api-generated/db.html). The instance of the needed MongoDB database.
- index – String. The collection's name. index is sent in the body of the Flexmonster request.
- fieldObject – Field Object (/api/field-object/). Represents a field with its properties. fieldObject is sent in the body of the Flexmonster request.
- page – Object. Has the following properties:
    - pageNumber – Number. Page number. Used to load members by parts. Starts from 0. pageNumber is sent in the body of the Flexmonster request.
    - pageToken – String. A key calculated on the server that defines which part of the data should be loaded next. pageToken is sent in the body of the Flexmonster request.

Returns

Array of objects, which contains all the members. The response format is the same as in the /members request (https://www.flexmonster.com/api/members-request/).

Example

```
mongo.post("/members", async (req, res) => {
    try {
        const result = await MongoDataAPI.getMembers(mongoDBInstance, req.body.index
, req.body.field, {pageNumber: req.body.page, pageToken: req.body.pageToken });
        res.json(result);
    } catch (err) { }
});
```

See also

getSchema (/api/getschema)
getSelectResult (/api/getselectresult)

6.4. getSelectResult

getSelectResult(mongoDBInstance: Db instance, index: String, query: Object, page: Object)

[starting from version: 2.8]

This API call allows getting the data from a MongoDB database. Used in the handler of the Flexmonster /select request. The getSelectResult method is a part of the Flexmonster MongoDB Connector API.

Parameters:

- mongoDBInstance – Db instance (https://mongodb.github.io/node-mongodb-native/api-generated/db.html). The instance of the needed MongoDB database.
- index – String. The collection's name. index is sent in the body of the Flexmonster request.
- query – Object. query is sent in the body of the Flexmonster request.
- page – Object. Has the following properties:
    - pageNumber – Number. Page number. Used to load members by parts. Starts from 0. pageNumber is sent in the body of the Flexmonster request.
    - pageToken – String. A key calculated on the server that defines which part of the data should be loaded next. pageToken is sent in the body of the Flexmonster request.

Returns

Array of objects, which contains the aggregated data. The response format for compact and classic tables is the same as in the /select request for the pivot table (https://www.flexmonster.com/api/select-request-for-pivot-table/). The response for the flat table corresponds to the response of the /select request for the flat table (https://www.flexmonster.com/api/select-request-for-flat-table/).

Example

```
mongo.post("/select", async (req, res) => {
    try {
        const result = await MongoDataAPI.getSelectResult(mongoDBInstance, req.body.
index, req.body.query,{ page: req.body.page, pageToken: req.body.pageToken });
        res.json(result);
    } catch (err) { }
});
```

See also

getSchema (/api/getschema)
getMembers (/api/getmembers-mongo/)

7. Flexmonster Connector for amCharts

7.1. All methods

Flexmonster Connector for amCharts has methods for requesting data from Flexmonster Pivot, passing it to amCharts, and getting the number format defined in Flexmonster.

This API reference provides descriptions for all the Connector's methods. For integration details, refer to the guide on integration with amCharts (/doc/integration-with-amcharts/).

List of the Connector's methods

amcharts.getData() (/api/amcharts-getdata/)	requests data from Flexmonster and preprocesses it for amCharts

amcharts.getCategoryName() (/api/amcharts-getcategoryname/)	returns the name of a field representing a category in data for the chart
amcharts.getMeasureNameByIndex() (/api/amcharts-getmeasurenamebyindex/)	returns the name of a measure with the specified index from data for the chart
amcharts.getNumberOfMeasures() (/api/amcharts-getnumberofmeasures/)	returns the total amount of available measures in the slice
amcharts.getNumberFormatPattern() (/api/amcharts-getnumberformatpattern/)	converts the component's number format to the amCharts number formatting string

7.2. amcharts.getData

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

Requests data from the component and preprocesses it to the format required by amCharts (https://www.amcharts.com/docs/v4/concepts/data/#Structure_of_data), namely to an array of objects.

Parameters

amcharts.getData() has the following parameters:

- options – Object. Allows setting options for data preprocessing. Has the following properties:
  - slice (optional) – Object. Defines the data slice to be used for the chart. If not defined, the Connector prepares the data based on the current slice in Flexmonster Pivot.
    Note: if amcharts.getData() gets the slice as a parameter, the chart will not respond to further slice changes on the grid, which means data shown on the chart will be static.
  - prepareDataFunction (optional) – Function. Allows you to override the Connector's default prepareDataFunction and perform custom data preprocessing if needed.
    If prepareDataFunction is not specified, the Connector uses the built-in method to preprocess the data.
    prepareDataFunction takes two input parameters:
    - rawData – Object. Raw data to preprocess (check out the structure of rawData in getData() (/api/getdata/));
    - options – Object. It contains options set in the amcharts.getData() function.
- callbackHandler – Function. Used to create the chart once the data is ready to be passed to it. Takes two input parameters:
  - chartData – data preprocessed by either the Connector or prepareDataFunction (if it is defined).
  - rawData – raw data from the component. It can be used to get the number formatting specified in Flexmonster. Check out the structure of rawData (/api/getdata/).
- updateHandler (optional) – Function. Used to update the chart when the report is updated. It takes the same input parameters as the callbackHandler function: chartData and rawData.

Returns

Returns an array of objects that contains the data for the chart. For example:

```
[
    {
        "categoryName": "value",
        "measureName 1": "value",
         …
        "measureName n": "value",
    }
    …
]
```

Learn more about how the Connector prepares data (/doc/integration-with-amcharts/#data-preparation) for the chart.

Example

Here is an example of passing the slice to the amcharts.getData() method:

```
flexmonster.amcharts.getData(
    {
        slice: {
            rows: [{uniqueName: "Country"}],
            columns: [{uniqueName: "[Measures]"}],
            measures: [{uniqueName: "Quantity"}]
        }
    },
    drawChart,
    updateChart
);
```

See also

amcharts.getCategoryName (https://www.flexmonster.com/api/amcharts-getcategoryname/)
amcharts.getMeasureNameByIndex (https://www.flexmonster.com/api/amcharts-getmeasurenamebyindex/)
amcharts.getNumberOfMeasures (https://www.flexmonster.com/api/amcharts-getnumberofmeasures/)
amcharts.getNumberFormatPattern (https://www.flexmonster.com/api/amcharts-getnumberformatpattern/)

7.3. amcharts.getCategoryName

amcharts.getCategoryName(rawData: Object)

Returns the name of a field that represents a category in data for the chart. It is needed to set up the amCharts Category axis (https://www.amcharts.com/docs/v4/concepts/axes/category-axis/).

Parameters
amcharts.getCategoryName() has the following parameter:

- rawData – Object. Raw data that contains the category (check out the structure of rawData in getData() (/api/getdata/)).

Returns

Returns a string, which is the name of a field representing a category in the chart's data.

Example

Here is an example of setting the name for the amCharts Category axis

(https://www.amcharts.com/docs/v4/concepts/axes/category-axis/):

```
var categoryAxis = chart.xAxes.push(new am4charts.CategoryAxis());
categoryAxis.dataFields.category = pivot.amcharts.getCategoryName(rawData);
```

See also

amcharts.getData (/api/amcharts-getdata/)
amcharts.getMeasureNameByIndex (https://www.flexmonster.com/api/amcharts-getmeasurenamebyindex/)
amcharts.getNumberOfMeasures (https://www.flexmonster.com/api/amcharts-getnumberofmeasures/)
amcharts.getNumberFormatPattern (https://www.flexmonster.com/api/amcharts-getnumberformatpattern/)

7.4. amcharts.getMeasureNameByIndex

amcharts.getMeasureNameByIndex(rawData: Object, index: Number)

Returns the name of a measure with the specified index from data for the chart. Use amcharts.getMeasureNameByIndex() to set up the amCharts Value axis (https://www.amcharts.com/docs/v4/concepts/axes/value-axis/).

Parameters

amcharts.getMeasureNameByIndex() has the following parameters:

- rawData – Object. The raw data that contains the measure name (check out the structure of rawData in getData() (/api/getdata/)). The measures in rawData have the same order as in the slice.
- index – Number. The index of the measure to get.

Returns

Returns a string representing the name of a measure with the given index.

Example

Here is an example of specifying the amCharts Value axis (https://www.amcharts.com/docs/v4/concepts/axes/value-axis/):

```
var series = chart.series.push(new am4charts.PieSeries());
series.dataFields.value = pivot.amcharts.getMeasureNameByIndex(rawData, 0);
```

See also

amcharts.getData (/api/amcharts-getdata/)
amcharts.getCategoryName (https://www.flexmonster.com/api/amcharts-getcategoryname/)
amcharts.getNumberOfMeasures (https://www.flexmonster.com/api/amcharts-getnumberofmeasures/)
amcharts.getNumberFormatPattern (https://www.flexmonster.com/api/amcharts-getnumberformatpattern/)

7.5. amcharts.getNumberOfMeasures

amcharts.getNumberOfMeasures(rawData: Object)

Returns the total number of available measures in the slice.

This method can be used to create a chart with multiple series (https://www.amcharts.com/docs/v4/chart-types/xy-chart/#Mixing_series) (e.g., the stacked column chart).

Parameters

amcharts.getNumberOfMeasures() has the following parameters:

- rawData – Object. Raw data containing the measures to count (check out the structure of rawData in getData() (/api/getdata/)).

Returns

Returns a number representing the quantity of measures in the slice.

Example

Here is an example of creating a 100% stacked column chart with amcharts.getNumberOfMeasures():

```
for (var s = 0; s < pivot.amcharts.getNumberOfMeasures(rawData); s++) {
    var series = chart.series.push(new am4charts.ColumnSeries());
    series.dataFields.categoryX = pivot.amcharts.getCategoryName(rawData);
    series.dataFields.valueY = pivot.amcharts.getMeasureNameByIndex(rawData, s);
    series.name = pivot.amcharts.getMeasureNameByIndex(rawData, s);
    series.stacked = true;
}
```

See the full code on JSFiddle (https://jsfiddle.net/flexmonster/ujp9n4sk/).

See also

amcharts.getData (/api/amcharts-getdata/)
amcharts.getCategoryName (https://www.flexmonster.com/api/amcharts-getcategoryname/)
amcharts.getMeasureNameByIndex (https://www.flexmonster.com/api/amcharts-getmeasurenamebyindex/)
amcharts.getNumberFormatPattern (https://www.flexmonster.com/api/amcharts-getnumberformatpattern/)

7.6. amcharts.getNumberFormatPattern

amcharts.getNumberFormatPattern(format: Format Object)

Converts the Format Object (https://www.flexmonster.com/api/format-object/) to the amCharts number formatting string.

To learn how the number format is set in amCharts, see the number formatting section (/doc/integration-with-

amcharts/#number-formatting).

Parameters

amcharts.getNumberFormatPatter() has the following parameter:

- format – Format Object (https://www.flexmonster.com/api/format-object/). Contains the number format set in the component.

Returns

This method returns a string representing the amCharts number formatting string. For example: '$'#,###.00|'($'#,###.00')'.

Example

1) Applying the formatting to the Value axis of the chart:

```
// Create a number formatter for the value axis
valueAxis.numberFormatter = new am4core.NumberFormatter();
// Get a format object from Flexmonster
var numberFormat = flexmonster.amcharts
                   .getNumberFormatPattern(rawData.meta.formats[0]);
// Apply number formatting to the value axis
valueAxis.numberFormatter.numberFormat = numberFormat;
```

2) Applying the formatting to the chart's tooltip text:

```
var numberFormat = flexmonster.amcharts
                   .getNumberFormatPattern(rawData.meta.formats[0]);
series.columns.template.tooltipText =
"{valueY.value.formatNumber('" + numberFormat + "')}";
```

See also

amcharts.getData (/api/amcharts-getdata/)
amcharts.getCategoryName (https://www.flexmonster.com/api/amcharts-getcategoryname/)
amcharts.getMeasureNameByIndex (https://www.flexmonster.com/api/amcharts-getmeasurenamebyindex/)
amcharts.getNumberOfMeasures (https://www.flexmonster.com/api/amcharts-getnumberofmeasures/)