API Reference

by Flexmonster

1. API reference

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

This API documentation describes calls and callbacks.

1.1. addCondition

addCondition(condition:Object)

Version: 1.5

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

You can create as many conditions with different formatting for one report as you need, there is no limit towards the number of conditions. If there are more than one conditional formatting rules for the report, they will be applied one by one in the order they have been created.

The conditional formatting may be added to all pivot table cells, to the cell specifying row and column indexes, to totals and sub-totals only, to regular cells only, or to the cells of selected measure, hierarchy, and hierarchy's member. All these parameters can be specified in condition object.

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

When you save the report all the conditional formatting will be saved as well and loaded when report is retrieved.

Parameters

condition – the object that describes the conditional formatting rule. Object has the following properties:

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

Examples

If cell value is more than 100, then apply trueStyle to this cell:

```
var condition = {
  id: 1,
  formula: 'if(#value > 100, "trueStyle")',
  trueStyle: {fontSize : 10, backgroundColor: "#33BB33"},
};
flexmonster.addCondition(condition);
flexmonster.refresh();
```

This rule will be applied only to Sales measure totals and sub-totals cells. If Sales is between 100000 and 200000, then apply trueStyle to the cell, else apply false style.

```
var condition = {
  id: 2,
  measure: "[Measures].[Sales]",
  isTotal: true,
  formula: 'if(AND(#value > 100000, #value < 200000), "trueStyle", "falseStyle")',
  trueStyle: {fontSize : 11, fontWeight : "bold", backgroundColor: "#00FF00"},
  falseStyle: {fontSize : 11, color: "#000000"}
};
flexmonster.addCondition(condition);
flexmonster.refresh();
```

See also

getCondition
getAllConditions
removeCondition
removeAllConditions
refresh

1.2. addJSON

addJSON(json:Array)

Version: 2.2

Sets JSON data source.

Parameters

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

- type – data type. Can be:
  - "string" – field contains string data. You will be able to aggregate it only with Count and Distinct Count aggregations. It will be sorted as string data.
  - "number" – field contains numeric data. You will be able to aggregate it with all different

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

Examples

To add JSON data using embedPivotComponent() API call:

```
/**
* The data can be set directly in embedPivotComponent()
*/
flexmonster.embedPivotComponent("flexmonster/", "pivotContainer", "100%", "515", {
  data: [
    {"Category" : "Accessories", "Color" : "green", "Quantity" : 22}
  ]
});
```

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

```
var jsonData = [
  {
    "Color":{"type": "string"},
    "M":{"type": "month",
        "dimensionUniqueName": "Days",
        "dimensionCaption": "Days",
        "caption":"Month"},
    "W":{"type": "weekday",
        "dimensionUniqueName": "Days",
```

```
      "dimensionCaption": "Days",
      "caption":"Week Day"},
  "country":{"type":"level",
      "hierarchy": "Geography",
      "level":"Country"},
  "state":{"type":"level",
      "hierarchy": "Geography",
      "level":"State",
      "parent": "Country"},
  "city":{"type":"level",
      "hierarchy": "Geography",
      "parent": "State"},
  "Price":0,
  "Quantity":{"type": "number"}
},
{
  "Color":"green",
  "M":"September",
  "W":"Wed",
  "country" : "Canada",
  "state" : "Ontario",
  "city" : "Toronto",
  "Price":174,
  "Quantity":22
},
{
  "Color":"red",
  "M":"March",
  "W":"Mon",
  "Time":"1000",
  "country" : "USA",
  "state" : "California",
  "city" : "Los Angeles",
  "Price":1664,
  "Quantity":19
},
{
  "Color":"red",
  "M":"January",
  "W":"Mon",
  "country" : "Canada",
  "state" : "Quebec",
  "city" : "Montreal",
  "Price":1190,
  "Quantity":292
},
{
  "Color":"green",
  "D":"04/08/2014",
  "M":"August",
  "W":"Fri",
  "Time":"1000",
  "country" : "USA",
  "state" : "California",
  "city" : "Los Angeles",
```

```
  "Price":1222,
  "Quantity":730
},
{
  "Color":"white",
  "M":"March",
  "W":"Wed",
  "country" : "USA",
  "state" : "California",
  "city" : "Los Angeles",
  "Price":7941,
  "Quantity":73
},
{
  "Color":"red",
  "M":"August",
  "W":"Wed",
  "country" : "Canada",
  "state" : "Ontario",
  "city" : "Toronto",
  "Price":6829,
  "Quantity":19
},
{
  "Color":"green",
  "M":"January",
  "W":"Wed",
  "country" : "Canada",
  "state" : "Quebec",
  "city" : "Montreal",
  "Price":2995,
  "Quantity":98
},
{
  "Color":"white",
  "M":"February",
  "W":"Mon",
  "country" : "USA",
  "state" : "California",
  "city" : "Los Angeles",
  "Price":2471,
  "Quantity":93
},
{
  "Color":"yellow",
  "M":"March",
  "W":"Fri",
  "country" : "Canada",
  "state" : "Ontario",
  "city" : "Toronto",
  "Price":6650,
  "Quantity":40
},
{
  "Color":"blue",
```

```
    "M":"February",
    "W":"Wed",
    "country" : "Canada",
    "state" : "Ontario",
    "city" : "Toronto",
    "Price":865,
    "Quantity":45
  },
  {
    "Color":"purple",
    "M":"August",
    "W":"Wed",
    "country" : "Canada",
    "state" : "Quebec",
    "city" : "Montreal",
    "Price":511,
    "Quantity":46
  },
  {
    "Color":"blue",
    "M":"September",
    "W":"Mon",
    "country" : "Canada",
    "state" : "Quebec",
    "city" : "Montreal",
    "Price":981,
    "Quantity":18
  },
  {
    "Color":"blue",
    "M":"September",
    "W":"Fri",
    "country" : "Canada",
    "state" : "Ontario",
    "city" : "Toronto",
    "Price":284,
    "Quantity":24
  }
];

/**
* The data can be set using addJSON() and setReport() API calls
* only after creationComplete event is dispatched by Flexmonster Component
*/
function pivotCreationCompleteHandler() {
  flexmonsterAddJSON();
}

flexmonster.embedPivotComponent("flexmonster/", "pivotContainer", "100%", "515", {
  jsPivotCreationCompleteHandler: "pivotCreationCompleteHandler"
});

/**
* flexmonsterAddJSON() function illustrates how to set JSON data
* that is already on the page and define a slice based on this data.
```

```
*/
function flexmonsterAddJSON() {
  flexmonster.addJSON(jsonData);
  var slice = {
    rows: [{ uniqueName: "Color" }],
    columns: [{ uniqueName: "W" }, { uniqueName: "[Measures]" }],
    measures: [{ uniqueName: "Price"}]};
  flexmonster.runQuery(slice);
}
```

To add JSON data using setReport() API call:

```
/**
* The data can be set using addJSON() and setReport() API calls
* only after creationComplete event is dispatched by Flexmonster Component
*/
function pivotCreationCompleteHandler() {
  flexmonsterSetReport();
}

flexmonster.embedPivotComponent("flexmonster/", "pivotContainer", "100%", "515", {
  jsPivotCreationCompleteHandler: "pivotCreationCompleteHandler"
});

/**
* flexmonsterSetReport() function illustrates how to compose report based on JSON da
ta that is already on the page.
*/
function flexmonsterSetReport() {
  var report = {
    dataSourceType: "json",
    data: jsonData,
    rows: [{ uniqueName: "M" }],
    columns: [{ uniqueName: "Geography" }, { uniqueName: "[Measures]" }],
    measures: [{ uniqueName: "Quantity"}],
    configuratorActive: true,
    viewType: "grid"
  };
  flexmonster.setReport(report);
}
```

Open local JSON file:

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

Connect to remote JSON file:

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

1.3. addMeasure

addMeasure(measure:Object)

Version: 1.5

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

Parameters

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

- uniqueName – measure unique name, this property will be used as an identifier for the measure inside Pivot component and as the identifier to remove calculated measure via API.
- caption – measure caption.
- calculated — boolean value that defines whether measure is calculated.
- formula – string that represents the formula that can contain the following operations: +, -, *, /; other measures can be addressed using measure unique name and aggregation function, for example sum("Price") or max("Order"). Pivot supports the following aggregation functions for CSV data source: "sum", "count", "distinctcount", "average", "product", "min", "max", "percent", "percentofcolumn", "percentofrow", "index".
- grandTotalCaption (optional) – measure grand total caption.
- availableAggregations (optional) — array of strings that represents the list of aggregation functions which can be applied to the current measure. If it is calculated measure, it will be [].
- aggregation (optional) — type of aggregation function that will be applied to the measure ("sum", "count", "distinctcount", "average", "product", "min", "max", "percent", "percentofcolumn", "percentofrow", "index"). If it is calculated measure, it will be "none".
- active (optional) – boolean value that defines whether the calculated measure will be added to the list of available values but not selected (false) or will be selected for the report (true).
- format (optional) – number formatting name.

Examples

The following example shows the calculated measure Average Price which is calculated as

sum("[Measures].[Price]")/count("[Measures].[Price]") and will be added to the list of available measures but not selected for the report (active: false):

```
var measure = {
  formula: 'sum("[Measures].[Price]")/count("[Measures].[Price]")',
  calculated : true,
  uniqueName: "[Measures].[Average Price]",
  caption: "Average Price",
  grandTotalCaption: "Total Average",
  active: false
};
flexmonster.addMeasure(measure);
```

See also

removeMeasure
getAllMeasures
getMeasures

1.4. addStyleToMember

addStyleToMember(hierarchyName:String, memberName:String, style:String)

Version: 1.6

Applies style to the member of the specific hierarchy.

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

Examples

```
flexmonster.addStyleToMember("[Color].[Color]", "[Color].[Color].[red]", '{"backgrou
ndColor": "#AA3333"}');
flexmonster.addStyleToMember("[Country].[Country]", "[Country].[Country].[Canada]",
'{"rightIcon" : "./images/flags/ca.png"}');
flexmonster.addStyleToMember("[Category].[Category]", "", '{"color": "#AAAA33"}');
flexmonster.addStyleToMember("[Measures]", "[Measures].[Discount]", '{"underline": "
true"}');
flexmonster.refresh();
```

See also

addUrlToMember
refresh

1.5. addUrlToMember

addUrlToMember(hierarchyName:String, memberName:String, url:String)

Version: 1.6

Adds link to the member of the specific hierarchy.

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

Examples

```
flexmonster.addUrlToMember("[Color].[Color]", "", "http://flexmonster.com?color");
flexmonster.addUrlToMember("[Category].[Category]", "[Category].[Category].[Cars]",
"http://flexmonster.com?cars");
flexmonster.addUrlToMember("[Measures]", "[Measures].[Price]", "http://flexmonster.c
om?price");
flexmonster.refresh();
```

See also

addStyleToMember
refresh

1.6. clear

clear()

Version: 1.6

Clears the component's data and view.

Examples

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

1.7. clearFilter

clearFilter(hierarchyName:String)

Version: 1.4

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

Parameters

- hierarchyName – the name of the hierarchy.

Example

```
var filter = [
```

```
  '[Product].[Color].[red]',
  '[Product].[Color].[green]',
  '[Product].[Color].[blue]'
];
flexmonster.setFilter('[Product].[Color]', filter);

flexmonster.getFilter('[Product].[Color]');
/*
method getFilter() returns the following Array
[
  {'caption' : 'red', 'hierarchyName' : '[Product].[Color]', 'uniqueName' : '[Produc
t].[Color].[red]'},
  {'caption' : 'green', 'hierarchyName' : '[Product].[Color]', 'uniqueName' : '[Prod
uct].[Color].[green]'},
  {'caption' : 'blue', 'hierarchyName' : '[Product].[Color]', 'uniqueName' : '[Produ
ct].[Color].[blue]'}
]
*/

flexmonster.getFilterProperties('[Product].[Color]');
/*
method getFilterProperties() returns the following object, where type is 'members' a
nd members property has 3 elements
{
  type : 'members',
  members : [
   {'caption' : 'red', 'hierarchyName' : '[Product].[Color]', 'uniqueName' : '[Produ
ct].[Color].[red]'},
   {'caption' : 'green', 'hierarchyName' : '[Product].[Color]', 'uniqueName' : '[Pro
duct].[Color].[green]'},
   {'caption' : 'blue', 'hierarchyName' : '[Product].[Color]', 'uniqueName' : '[Prod
uct].[Color].[blue]'}
  ]
}
*/

flexmonster.clearFilter('[Product].[Color]');

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

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

See also

getFilter
getFilterProperties
setFilter
setTopX
setBottomX

1.8. clearXMLACache

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

Version: 2.2

API call that requests Microsoft Analysis Services to clear the cache. Available for HTML5 version only. Please visit official documentation from Microsoft for more details – https://msdn.microsoft.com/en-US/LIbrary/hh230974.aspx?f=255&MSPPError=-2147217396

Parameters

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

Example

```
<button onclick="clearCache()">Clear XMLA Cache</button>
<div id="pivotContainer"></div>
<script type="text/javascript" src="flexmonster/flexmonster.js"></script>
<script type="text/javascript">
  flexmonster.embedPivotComponent("flexmonster/", "pivotContainer", "100%", "615", {
configUrl : "config.msolap.xml"}, true);

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

1.9. closeFieldsList

closeFieldsList()

Version: 1.4

Closes Fields List.

Examples

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

See also

openFieldsList

1.10. collapseAllData

Version: 1.4

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

Examples

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

See also

collapseData
expandAllData
expandData

1.11. collapseData

collapseData(hierarchyName:String)

Version: 1.6

Collapses all nodes of the specified hierarchy.

Examples

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

See also

collapseAllData
expandAllData
expandData

1.12. connectTo

connectTo(params:Object, callbackHandler:String)

Version: 1.4

Connects the component to the specified data source.

There are three possible data sources:

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

Parameters

- params – the object which contains connection parameters. List of possible parameters:
    - dataSourceType – type of data source. The component supports the following types: "microsoft analysis services", "mondrian", "iccube", "csv", "ocsv", "json".
    - proxyUrl – the path to proxy URL to the OLAP data source, such as Microsoft Analysis Services, Mondrian, icCube (only for "microsoft analysis services", "mondrian", "iccube" data source types)
    - dataSourceInfo – the service info of the OLAP data source (only for "microsoft analysis services", "mondrian", "iccube" data source types)
    - catalog – the data source catalog name of the OLAP data source (only for "microsoft analysis services", "mondrian", "iccube" data source types)
    - cube – given catalog's cube's name of the OLAP data source (only for "microsoft analysis services", "mondrian", "iccube" data source types)
    - filename – the URL to CSV file or to server side script which generates CSV data (only for "csv", "ocsv" and "json" data source type)
    - browseForFile – this boolean parameter defines whether you want to load CSV file from the local file system (true) or not (false). It is *false* by default. (only for "csv", "ocsv" and "json" data source type)
    - fieldSeparator – defines specific fields separator to split CSV row (only for "csv" data source type). There is no need to define it if CSV fields are separated by , or ;. This property is used if another char separates fields. For example, if you use TSV, where tab char is used to separate fields in row, fieldSeparator parameter should be defined explicitly.
- callbackHandler (starting from v2.1) (optional) – Callback handler is called when Pivot connected to the data source or failed to connect to it. It is called with the following object { success: true } if the connection was made successfully or with { success: false } if it failed.

Examples

Connect to Microsoft Analysis Services:

```
flexmonster.connectTo({
  dataSourceType: 'microsoft analysis services',
  proxyUrl: 'http://olap.flexmonster.com/olap/msmdpump.dll',
  dataSourceInfo: 'Provider=MSOLAP; Data Source=extranet;',
```

```
  catalog: 'Adventure Works DW Standard Edition',
  cube: 'Adventure Works'
});
```

Connect to Mondrian:

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

Connect to icCube:

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

Connect to CSV data source:

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

Connect to TSV file (where tab char is used to separate fields in row). You have to define fieldSeparator explicitly:

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

Open local CSV file:

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

See also

open
load
save
getReport
setReport

1.13. embedPivotComponent

embedPivotComponent(path:String, container:String, width:Number, height:Number, params:Object, withToolbar:Boolean, toolbarLocalization:Object)

Version: 1.4

Embeds the component into HTML page.

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 v2.0 you can:

1. Embed JavaScript toolbar from All-in-One demo by setting withToolbar parameter in embedPivotComponent() function. Also, new toolbar adapts to mobile browsers to become more user-friendly. By default flexmonster.js does not embed the toolbar.
2. Localize JavaScript toolbar using toolbarLocalization parameter in embedPivotComponent() function. Read more about localization in Localizing component article.

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

Parameters

- path – 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.
- container – id of the HTML element you would like to have as a container for the component (HTML5 version); replaced by the component's content (Flash version).
- width – width of the component on the page (pixels or percent).
- height – height of the component on the page (pixels or percent).
- params – report parameters and licenseKey as JSON. Among all report parameters, the following parameters for setting handlers are supported as well:
    - jsPivotCreationCompleteHandler – parameter to specify JavaScript handler which will be called when the component creation is complete and it is ready for API calls.
    - jsReportLoadedHandler – parameter to specify JavaScript handler which will be called when the component loaded report.
    - jsReportChangeHandler – parameter to specify JavaScript handler which will be called when a report is changed. Use it to track changes in a report object.
    - jsPivotUpdateHandler – parameter to specify JavaScript handler which will be called when the component loaded data, updated data slice, filter or sort. Use it to retrieve information about data source structure.
    - jsFullScreenHandler – parameter to specify JavaScript handler which will be called when the component enters or leaves full-screen display mode.
    - jsFieldsListOpenHandler – parameter to specify JavaScript handler which will be called when the

built-in fields list is opened.
    - jsFieldsListCloseHandler – parameter to specify JavaScript handler which will be called when the built-in fields list is closed.
    - jsFilterOpenHandler – parameter to specify JavaScript handler which will be called when a filter icon is pressed.
    - jsCellClickHandler – parameter to specify JavaScript handler which will be called when a cell is clicked on the grid.
- withToolbar (optional) – parameter to embed JavaScript toolbar (the one that is shown in All-in-One demo) or not. Default value is false – without toolbar.
- toolbarLocalization (optional) – object with toolbar localization. Read more about localization in Localizing component article.

Returns

Object. If you want to work with multiple instances on the same page use objects returned by the calls of this method. These objects have all methods described in this manual.

Examples

Add the component instance to your web page:

a) the component without toolbar:

```
<div id="pivotContainer">The component will appear within this DIV.</div>
<script type="text/javascript" src="flexmonster/flexmonster.js"></script>
<script type="text/javascript">
  flexmonster.embedPivotComponent("flexmonster/", "pivotContainer","100%", "500",
    {configUrl : "config.xml"});
</script>
```

b) the component with toolbar:

```
<div id="pivotContainer">The component will appear within this DIV.</div>
<script type="text/javascript" src="flexmonster/flexmonster.js"></script>
<script type="text/javascript">
  flexmonster.embedPivotComponent("flexmonster/", "pivotContainer","100%", "500",
    {configUrl : "config.xml"}, true);
</script>
```

Add and operate with multiple instances:

```
<div id="firstPivotContainer">The component will appear within this DIV.</div>
<div id="secondPivotContainer">The component will appear within this DIV.</div>
<script type="text/javascript" src="flexmonster/flexmonster.js"></script>
<script type="text/javascript">
  var pivot1 = flexmonster.embedPivotComponent(
    "flexmonster/", "firstPivotContainer","100%", "500",
    {configUrl : "firstconfig.xml"}
  );
  var pivot2 = flexmonster.embedPivotComponent(
    "flexmonster/", "secondPivotContainer","100%", "500",
```

```
    {configUrl : "secondconfig.xml"}
  );
</script>

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

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

All params which can be set via embedPivotComponent():

```
var pivot = flexmonster.embedPivotComponent("", "pivotContainer", "100%", "600",
 {
  configUrl: "config.xml",
  jsPivotCreationCompleteHandler: "pivotCreationCompleteHandler",
  jsReportLoadedHandler: "pivotReportLoadedHandler",
  jsReportChangeHandler: "pivotReportChangeHandler",
  jsPivotUpdateHandler: "pivotUpdateHandler",
  jsFullScreenHandler: "fullScreenHandler",
  jsFieldsListOpenHandler: "fieldsListOpenHandler",
  jsFieldsListCloseHandler: "fieldsListCloseHandler",
  jsFilterOpenHandler: "filterOpenHandler",
  jsCellClickHandler: "cellClickHandler"
 },
 true);

function pivotCreationCompleteHandler() {
 console.log("The component was created");
}

function pivotReportLoadedHandler() {
 console.log("Report is ready");
}

function pivotReportChangeHandler() {
 console.log("Report is changed");
}

function pivotUpdateHandler() {
 console.log("Pivot updated");
}

function fullScreenHandler(fullScreenMode) {
 console.log("Fullscreen", fullScreenMode);
}
```

```
function fieldsListOpenHandler() {
 console.log("Fields List opened");
}

function fieldsListCloseHandler() {
 console.log("Fields List closed");
}

function filterOpenHandler(params) {
 console.log("Filter cell was clicked", params);
}

function cellClickHandler(cell) {
 console.log("Cell was clicked", cell);
}
```

See also

jsPivotCreationCompleteHandler
jsReportLoadedHandler
jsReportChangeHandler
jsPivotUpdateHandler
jsFullScreenHandler
jsFieldsListOpenHandler
jsFieldsListCloseHandler
jsFilterOpenHandler
jsCellClickHandler

1.14. expandAllData

expandAllData(withAllChildren:Boolean)

Version: 1.4

Expands all nodes and drills down all levels of all hierarchies in 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 slice. Please set it to false if you want just expand nodes. Default value is true, which means that the drill down will be performed as well.

Examples

Expands all nodes and drills down all hierarchies in slice.

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

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

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

See also

expandData
collapseAllData
collapseData

1.15. expandData

expandData(hierarchyName:String)

Version: 1.6

Expands all nodes of the specified hierarchy.

Examples

```
flexmonster.expandData('[Product].[Color]');
```

See also

expandAllData
collapseAllData
collapseData

1.16. exportTo

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

Version: 1.4

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

Parameters

- type – type of export. There are such types available: "csv", "html", "pdf", "image" and "excel".
- params (optional) – export parameters. Object params can contain the following properties:
  - filename – default name of the resulting file.
  - destinationType – defines where the component's content will be exported. Destination type can be the following:
    - "file" – the component's content will be exported to the file to the local computer.
    - "server" – the component's content will be exported to the server (server-side script is required).
    - "clipboard" (Flash version only, no in HTML5) – the component's content will be exported to clipboard. Please note that binary files (image, PDF, Excel) cannot be saved to the

clipboard due to the Flash Player security policy.
        - "plain" (starting from v2.1) – The component's content will be exported to plain data (for CSV only).
    - excelSheetName (starting from v2.2) (optional) – String. To configure the sheet name when exporting to Excel file.
    - fontUrl (starting from v2.1) (optional) (Flash version only, no in HTML5) – String. PDF only. Specifies the path to the external font (TTF) which will be used during exporting. This is optional. Default value is "".
    - footer (starting from v2.1 for Flash version; starting from v2.211 for HTML5 version) (optional) – String.
        - HTML5 version: PDF only. Footer is set in HTML format (tags, inline styles, img with base64 src), rendered in browser and added as image to the exported PDF file. The following tokens can be used for PDF export: ##CURRENT-DATE##, ##PAGE-NUMBER##. They will be replaced by appropriate data.
        - Flash version: PDF and HTML only. Footer in HTML format which is added to exported file. Only limited amount of tags are supported for PDF export: div, p, span, br, img with base64 src, some basic inline styles. The following tokens can be used for PDF export: ##TITLE##, ##CURRENT-DATE##, ##PAGE-NUMBER##, ##TOTAL-PAGES##. They will be replaced by appropriate data. The following tokens can be used for HTML export: ##TITLE##, ##CURRENT-DATE##. They will be replaced by appropriate data.
    - header (starting from v2.1 for Flash version; starting from v2.211 for HTML5 version) (optional) – String.
        - HTML5 version: PDF only. Header is set in HTML format (tags, inline styles, img with base64 src), rendered in browser and added as image to the exported PDF file. The following tokens can be used for PDF export: ##CURRENT-DATE##, ##PAGE-NUMBER##. They will be replaced by appropriate data.
        - Flash version: PDF and HTML only. Header in HTML format which is added to exported file. Only limited amount of tags are supported for PDF export: div, p, span, br, img with base64 src, some basic inline styles. The following tokens can be used for PDF export: ##TITLE##, ##CURRENT-DATE##, ##PAGE-NUMBER##, ##TOTAL-PAGES##. They will be replaced by appropriate data. The following tokens can be used for HTML export: ##TITLE##, ##CURRENT-DATE##. They will be replaced by appropriate data.
    - pageOrientation (optional) – defines the page orientation for a PDF file. Page orientation can be the following:
        - "portrait" (by default) – defines portrait page orientation for a PDF file.
        - "landscape" – defines landscape page orientation for a PDF file.
    - showFilters (starting from v2.1) (optional) – Boolean. Excel only. Indicates whether the filters info will be shown (true) in exported Excel file or not (false). Default value is false.
    - url – to save file to the server you should provide the component with a path to the server-side script which can save this file.
    - useOlapFormattingInExcel (starting from v2.2) (optional) – Boolean. To configure how to export grid cells in Excel file if formatting is taken from OLAP cube – as a formatted string (true) or as numbers without formatting (false). Previously it was not configurable and the cells were exported as formatted strings.
- callbackHandler (starting from v2.1) (optional) – Callback handler is called only if type is "csv" and destinationType is "plain". It is called with the following object { data: result }, where result is plain CSV data (CSV string).

Examples

Export to CSV and save as local file:

```
flexmonster.exportTo('CSV', {filename : 'flexmonster.csv'});
```

Export to HTML file and save it to clipboard:

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

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 : 'http://flexmonster.com/save.php'
};
flexmonster.exportTo('PDF', params);
```

Export to Excel and save as local file:

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

See also

print

1.17. fullScreen

fullScreen()

Version: 1.4

Switches the component to full-screen mode.

Examples

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

See also

showCharts
showGrid

1.18. getAllConditions

getAllConditions():Array

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 object in array has the following properties:

- formula – IF statement with 3 arguments: IF(CONDITION, TRUE STYLE, FALSE STYLE), where the false style is optional. Condition can contain AND, OR, ==, !=, >, <, >=, <=, -, +, *, /. #value is used to address the cell value in condition. Example: if(#value > 2, "trueStyle", "falseStyle").
- trueStyle – style object that is applied to a cell if the condition for the cell value is met.
- falseStyle – style object. If it is set it is applied to a cell if the condition for the cell value is not met.
- id – id of the conditional formatting rule.
- row – row index to which the condition is applied. The default value is -1, which means that the row index was not specified for this rule.
- column – column index to which the condition is applied. The default value is -1, which means that the column index was not specified for this rule.
- measure – measure unique name to which the condition is applied. If the measure was not specified for this rule, the property is empty string.
- hierarchy – hierarchy unique name to which the condition is applied. If the hierarchy was not specified for this rule, the property is empty string.
- member – hierarchy's member unique name to which the condition is applied. If the hierarchy's member was not specified for this rule, the property is empty string.
- isTotal – Number. If it is -1 (not defined) – the condition is applied for all cells. If it is 0 (false) – the condition is applied to regular cells only. If it is 1 (true) – the condition will be applied to totals and sub-totals cells only.

Examples

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

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

See also

addCondition
getCondition
removeCondition
removeAllConditions

1.19. getAllHierarchies

getAllHierarchies():Array

Version: 1.4

Returns a list of all available hierarchies.

Returns

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

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

Examples

```
flexmonster.getAllHierarchies();

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

See also

getAllMeasures
getColumns
getRows
getPages
getMeasures

1.20. getAllMeasures

getAllMeasures():Array

Version: 1.4

Returns a list of all available measures.

Returns

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

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

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

Examples

```
flexmonster.getAllMeasures();

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

See also

getMeasures
getAllHierarchies
getColumns
getRows
getPages

1.21. getCell

getCell(rowIdx:Number, colIdx:Number):Object

Version: 1.4

Returns information about cell by row and column indexes.

Parameters

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

Returns

Object which contains information about the requested cell. Object has the following parameters:

HTML5 version:

- columnIndex
- columns
- isTotal
- label
- measure
- rowIndex
- rows
- type – it can be "header" or "value"
- value

Flash/Flex version:

- columnIndex
- columns
- height
- isTotal
- label
- measure
- rowIndex
- rows
- styleName
- type – it can be "header" or "value"
- value
- width
- x
- y

See also

getSelectedCell
gridRowCount
gridColumnCount

1.22. getColumns

getColumns():Array

Version: 1.4

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

Returns

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

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

Examples

```
flexmonster.getColumns();

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

See also

getAllHierarchies
getRows
getPages
getAllMeasures
getMeasures

1.23. getColumnWidth

getColumnWidth(columnIndex:Number):Number

Version: 1.4

Returns specified column's width.

Parameters

- columnIndex – index of the column.

Returns

The width of specified column.

Examples

How to find max column width in the grid

```
var maxWidth = 0;
var columnCount = flexmonster.gridColumnCount();
for (var i = 0; i < columnCount; i++) {
  var width = flexmonster.getColumnWidth(i);
  if (maxWidth < width) maxWidth = width;
}
alert("Max width is: " + maxWidth);
```

See also

setColumnWidth

getRowHeight
setRowHeight

1.24. getCondition

getCondition(id:String):Object

Version: 1.6

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

Parameters

- id – id of the condition.

Returns

The object that describes the conditional formatting rule. It has the following properties:

- formula – IF statement with 3 arguments: IF(CONDITION, TRUE STYLE, FALSE STYLE), where the false style is optional. Condition can contain AND, OR, ==, !=, >, <, >=, <=, -, +, *, /. #value is used to address the cell value in condition. Example: if(#value > 2, "trueStyle", "falseStyle").
- trueStyle – style object that is applied to a cell if the condition for the cell value is met.
- falseStyle – style object. If it is set it is applied to a cell if the condition for the cell value is not met.
- id – id of the conditional formatting rule.
- row – row index to which the condition is applied. The default value is -1, which means that the row index was not specified for this rule.
- column – column index to which the condition is applied. The default value is -1, which means that the column index was not specified for this rule.
- measure – measure unique name to which the condition is applied. If the measure was not specified for this rule, the property is empty string.
- hierarchy – hierarchy unique name to which the condition is applied. If the hierarchy was not specified for this rule, the property is empty string.
- member – hierarchy's member unique name to which the condition is applied. If the hierarchy's member was not specified for this rule, the property is empty string.
- isTotal – Number. If it is -1 (not defined) – the condition is applied for all cells. If it is 0 (false) – the condition is applied to regular cells only. If it is 1 (true) – the condition will be applied to totals and sub-totals cells only.

Examples

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

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

See also

addCondition
getAllConditions
removeCondition
removeAllConditions

1.25. getFilter

getFilter(hierarchyName:String):Array

Version: 1.4

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

Parameters

- hierarchyName – the name of the hierarchy.

Returns

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

Example

```
var filter = [
  '[Product].[Color].[red]',
  '[Product].[Color].[green]',
  '[Product].[Color].[blue]'
];
flexmonster.setFilter('[Product].[Color]', filter);
flexmonster.getFilter('[Product].[Color]');

/*
method getFilter() returns the following Array
[
  {'caption' : 'red', 'hierarchyName' : '[Product].[Color]', 'uniqueName' : '[Produc
t].[Color].[red]'},
  {'caption' : 'green', 'hierarchyName' : '[Product].[Color]', 'uniqueName' : '[Prod
uct].[Color].[green]'},
  {'caption' : 'blue', 'hierarchyName' : '[Product].[Color]', 'uniqueName' : '[Produ
ct].[Color].[blue]'}
]
*/

flexmonster.clearFilter('[Product].[Color]');
flexmonster.getFilter('[Product].[Color]');

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

See also

clearFilter

1.26. getFilterProperties

getFilterProperties(hierarchyName:String):Object

Version: 1.6

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

Parameters

- hierarchyName – the name of the hierarchy.

Returns

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

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

Example

```
var filter = [
  '[Product].[Color].[red]',
  '[Product].[Color].[green]',
  '[Product].[Color].[blue]'
];
flexmonster.setFilter('[Product].[Color]', filter);

flexmonster.getFilterProperties('[Product].[Color]');
/*
method getFilterProperties() returns the following object, where type is 'members' a
nd members property has 3 elements
{
  type : 'members',
  members : [
   {'caption' : 'red', 'hierarchyName' : '[Product].[Color]', 'uniqueName' : '[Produ
ct].[Color].[red]'},
```

```
   {'caption' : 'green', 'hierarchyName' : '[Product].[Color]', 'uniqueName' : '[Pro
duct].[Color].[green]'},
   {'caption' : 'blue', 'hierarchyName' : '[Product].[Color]', 'uniqueName' : '[Prod
uct].[Color].[blue]'}
  ]
}
*/

flexmonster.clearFilter('[Product].[Color]');

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

See also

clearFilter
getFilter
setFilter
setTopX
setBottomX

1.27. getFormat

getFormat(measureName:String):Object

Version: 1.4

Returns a default number format or the number format for the specified measure. The number format can be defined via report or via setFormat() API method. Each measure has only one format but a format can be applied to more than one measure.

Parameters

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

Returns

Object that contains the number format parameters:

- name – String. It identifies the format in the report, thus, it should be unique. The default is "", which means that this number format is a default one and it is applied to all the measures for which the specific number format is not set.
- thousandsSeparator – String. The default is " " (space).

- decimalSeparator – String. The default is ".".
- decimalPlaces – Number. The exact number of decimals to show in the fractional part of a number after the decimal separator. The default is -1, which means that the number will be shown as is.
- maxDecimalPlaces – Number. The maximum number of decimals to show in the fractional part of a number after the decimal separator. The default is -1, which means the number will be shown as is.
- maxSymbols – Number. The maximum number of symbols in a cell. The default is 20.
- currencySymbol – String. The symbol which is shown near the value (currency symbol, hours, percent, etc.). The default is "".
- currencySymbolAlign – String. The alignment of the currency symbol. It can be "left" or "right". The default is "left".
- nullValue – String. It defines how to show null values in the grid. The default is "".
- infinityValue – String. It defines how to show infinity values in the grid. The default is "Infinity".
- divideByZeroValue – String. It defines how to show divided by zero values in the grid. The default is "Infinity".
- textAlign – String. The alignment of formatted values in cells on the grid: "right" or "left". The default is "right".

Examples

How to get a precision:

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

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();
```

See also

setFormat

1.28. getLabels

getLabels():Object

Version: 1.6

Returns an object which represents current localization. The object contains pairs of key and value of all localized labels.

Returns

Object which contains a list of available labels.

Examples

Change labels on “OK” and “Cancel” buttons

```
var labels = flexmonster.getLabels();
labels["BUTTON_OK"] = "Accept";
labels["BUTTON_CANCEL"] = "Reject";
flexmonster.setLabels(labels);
```

See also

setLabels

1.29. getMeasures

getMeasures():Array

Version: 1.4

Returns a list of the selected measures in the report.

Returns

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

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

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

Examples

```
flexmonster.getMeasures();

/* method returns array of objects
[
  {aggregation: "sum",
   availableAggregations: ["sum", "average", "percent"],
   availableAggregationsCaptions: ["Sum", "Count", "Percent"],
   calculated: false,
   caption: "Sum of Sales",
   format: "currency",
   formula: "",
   grandTotalCaption: "Total Sum of Sales",
```

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

See also

getAllMeasures
getAllHierarchies
getColumns
getRows
getPages

1.30. getMembers

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

Version: 1.6

Returns a list of members for the specified hierarchy.

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

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

Parameters

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

Returns

Array of objects with member's caption, uniqueName, and hierarchyName for both CSV data source and OLAP data source. Each object will contain children property for CSV data source. Each object will contain isLeaf and parentMember properties for OLAP data source.

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

Examples

CSV data source, where [Category].[Category] hierarchy has only one level:

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

/* method returns array of objects
[
  {caption: "Accessories", hierarchyName: "[Category].[Category]", uniqueName: "[Cat
egory].[Category].[Accessories]", children: []},
  {caption: "Cars", hierarchyName: "[Category].[Category]", uniqueName: "[Category].
[Category].[Cars]", children: []},
  {caption: "Clothing", hierarchyName: "[Category].[Category]", uniqueName: "[Catego
ry].[Category].[Clothing]", children: []}
]
*/
```

CSV data source, where [Date].[Date] hierarchy has more than one level:

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

/* method returns array of objects
[
  {caption: "2003",
  hierarchyName: "[Date].[Date]",
  uniqueName: "[Date].[Date].[2003]",
  children: [
   {caption: "Quarter 1",
   hierarchyName: "[Date].[Date]",
   uniqueName: "[Date].[Date].[Quarter].[2003/Quarter 1]",
   children: [
    {caption: "January",
    hierarchyName: "[Date].[Date]",
    uniqueName: "[Date].[Date].[Month].[2003/Quarter 1/January]",
    children: [ ]
    }
   },
   {caption: "Quarter 2",
   hierarchyName: "[Date].[Date]",
   uniqueName: "[Date].[Date].[Quarter].[2003/Quarter 2]",
   children: [
    {caption: "April",
    hierarchyName: "[Date].[Date]",
    uniqueName: "[Date].[Date].[Month].[2003/Quarter 2/April]",
```

```
    children: [ ]
    }
   }
  ]},
  {caption: "2004",
  hierarchyName: "[Date].[Date]",
  uniqueName: "[Date].[Date].[2004]",
  children: [
   {caption: "Quarter 3",
   hierarchyName: "[Date].[Date]",
   uniqueName: "[Date].[Date].[Quarter].[2004/Quarter 3]",
   children: [
    {caption: "July",
    hierarchyName: "[Date].[Date]",
    uniqueName: "[Date].[Date].[Month].[2004/Quarter 3/July]",
    children: [ ]
    },
    {caption: "August",
    hierarchyName: "[Date].[Date]",
    uniqueName: "[Date].[Date].[Month].[2004/Quarter 3/August]",
    children: [ ]
    }
   }
  ]}
]
*/
```

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

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

function onGetMembers(members) {
  for (int = 0; i < members.length; i++) {
   console.log(members[i]);
  }
}

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

```
  hierarchyName = "[Product].[Product Categories]"
  isLeaf = false
  parentMember = "[Product].[Product Categories].[All]"
  uniqueName = "[Product].[Product Categories].[Category].&[3]"
(Object)#0
  caption = "Components"
  hierarchyName = "[Product].[Product Categories]"
  isLeaf = false
  parentMember = "[Product].[Product Categories].[All]"
  uniqueName = "[Product].[Product Categories].[Category].&[2]"

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

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

1.31. getOptions

getOptions():Object

Version: 1.6

Returns an object with component's options.

Returns

Object which contains a list of component's options. The following options are available:

- chartAutoRange – Boolean. Indicates whether the range of values on the charts is selected automatically or not. Default value is false.
- chartMultipleMeasures – Boolean. Starting from v1.9. Indicates whether to show multiple measures on charts. Default value is false.
- chartOneLevel (starting from v2.1 in HTML5 only) – Boolean. In case of drillable chart, defines whether chart shows elements only for the lowest hierarchy (true). Default value is false. Bar, Line, Scatter, Bar stack and Bar line charts support this feature in HTML5 version.
- chartType – String. Selected chart type: "bar", "line", "scatter", "pie", "bar_stack" and "bar_line" (starting from v1.9). Default value is "bar".
- classicView (starting from v2.1) – Boolean. Defines whether pivot table should be shown in the classic view (true) or not (false). Default value is false, which means that classic view is turned off.
- configuratorActive – Boolean. Indicates whether the Fields List is opened (true) or closed (false). Default value is false.
- configuratorButton – Boolean. Indicates whether the Fields List toggle button is visible (true) or not (false). Default value is true.
- configuratorMatchHeight (starting from v2.1) – Boolean. Indicates whether the Fields List will be the same height as the component (true) or it's height will be defined by its content amount (false). Default value is false.
- drillThrough (starting from v2.1) – Boolean. Indicates whether the drill through feature is enabled (true) or disabled (false). User can drill through by double-clicking the cell with value. This feature is available for CSV data sources only. Default value is true.
- editing (starting from v2.1) – Boolean. Indicates whether the editing feature is enabled (true) or disabled (false) on the Drill Through popup for CSV data sources. User will be able to double-click the cell and enter new value in it if the editing feature is enabled.
- fitGridlines – Boolean. Indicates whether the gridlines are shown for all cells (false) or only non-empty

(true). Default value is false.
- flatView (starting from v2.2) – Boolean. Defines whether data should be shown in the flat view (true) or not (false). Default value is false, which means that flat view is turned off. Report parameter tableType (deprecated in 2.2) was replaced with the new flatView property.
- ignoreQuotedLineBreaks (starting from v2.1) – Boolean. Indicates whether the line breaks in quotes will be ignored (true) in CSV files or not (false). Default value is true, which makes CSV parsing faster. Set it to false only if your data source has valuable for you line breaks in quotes. Please note that this might slow down CSV parsing a little bit.
- showAggregations (starting from v2.0) – Boolean. Indicates whether the aggregation selection control is visible (true) or not (false) for measures on Fields List. Default value is true.
- showCalculatedValuesButton (starting from v2.2) – Boolean. Controls the visibility of “Add calculated value” in Fields List. Default value is true.
- showChartLegendButton (starting from v2.2) – Boolean. Indicates whether the button to show/hide the legend on charts is visible. Default value is false which means that the legend is always visible, without the button that hides it.
- showChartMeasures (starting from v2.2) – Boolean. Hides all dropdowns on the top of charts if you want to show simple chart without controls or you want to save space. Default value is true– the dropdowns are visible by default, as it was in previous versions.
- showChartsWarning – Boolean. Indicates whether the warning are shown if data is too big for charts. Default value is true.
- showDefaultSlice (starting from v2.2) – Boolean. Defines whether the component selects a default slice for the report with empty slice (when nothing is set in rows, columns, pages and measures). If true, the first hierarchy from data goes to rows and the first measure goes to columns in the default slice. To avoid this default behavior, please set this property to false. Default value is true.
- showFilter – Boolean. Indicates whether the opening columns/rows filter controls and page filter controls are visible (true) or not (false) on the grid. Default value is true.
- showFilterInCharts (starting from v2.2) – Boolean. Indicates whether the opening columns and rows filter controls are visible (true) or not (false) on the charts. Default value is true.
- showFiltersExcel Deprecated Since 2.1: Please use showFilters in export options of API call exportTo().
- showGrandTotals – String. Specifies how to show grand totals: in rows ('rows'), in columns ('columns'), in rows and columns ('on') or hide them ('off'). Default value is 'on'.
- showHeaders – Boolean. Indicates whether the spreadsheet headers are visible (true) or not (false). Default value is true.
- showMemberProperties (starting from v2.1) – Boolean. Indicates whether the member properties for OLAP data source are visible (true) or not (false) on Fields List. Default value is false.
- showReportFiltersArea (starting from v2.2) – Boolean. Indicates whether the reports filtering cells on the grid should be visible (true) or not (false). Default value is true.
- showTotals – Boolean. Indicates whether the totals are visible (true) or not (false). Default value is true.
- sorting (starting from v2.0) – String. Indicates whether the sorting controls are visible in rows ('rows'), in columns ('columns'), in rows and columns ('on' or true) on the grid cells or not visible ('off' or false). Default value is 'on'.
- useOlapFormatting (starting from v2.1) – Boolean. Indicates whether the values from MSAS data source will be formatted according to the format defined in the cube (true) or not (false). Default value is false.

Examples

How to turn off totals:

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

See also

setOptions

1.32. getPages

getPages():Array

Version: 1.4

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

Returns

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

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

Examples

```
flexmonster.getPages();

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

See also

getAllHierarchies
getColumns
getRows
getAllMeasures
getMeasures

1.33. getReport

getReport(format:String):*

Version: 1.4

Returns an object or XML string which describes the current report. Use this object to save or edit report at runtime.

Parameters

- format (optional) – format in which report will be returned (starting from v1.901). Can be "xmlstring" or

"object". Default is "object".

Returns

Object or XML string which describes the report and contains all its properties. Report object has the following parameters:

- browseForFile – Boolean. Defines whether you want to load CSV file from the local file system (true) or not (false). Default value is false. Only for CSV and JSON data source type.
- catalog – String. The data source catalog name of the OLAP data source (only for "microsoft analysis services", "mondrian", "iccube" data source types).
- catalogURL – String.
- chartActiveMeasure – String. Selected measure on charts view.
- chartActiveTupleIndex – Number. Selected tuple index on the Pie chart.
- chartAutoRange – Boolean. Indicates whether the range of values on the charts is selected automatically or not.
- chartLabelsHierarchy (starting from v2.1) – String. Indicates what column should be used for x axis labels in Flat Table charts. User can choose it from UI and it will be saved in report's chartLabelsHierarchy property. If it is empty, row numbers will be used. Default value is empty string.
- chartMultipleMeasures – Boolean. Starting from v1.9. Indicates whether to show multiple measures on charts. Default value is false.
- chartOneLevel (starting from v2.1 in HTML5 only) – Boolean. In case of drillable chart, defines whether chart shows elements only for the lowest hierarchy (true). Default value is false. Bar, Line, Scatter, Bar stack and Bar line charts support this feature in HTML5 version.
- chartTitle – String. Title of the chart.
- chartType – String. Selected chart type. The following chart types are supported: "bar", "line", "scatter", "pie", "bar_stack" and "bar_line" (starting from v1.9).
- classicView (starting from v2.1) – Boolean. Defines whether pivot table should be shown in the classic view (true) or not (false). Default value is false, which means that classic view is turned off.
- columnSizes – Array of Numbers. It is used to save and restore columns width in report.
- columnSorting – Object. Describes sorting on columns. Object has 3 properties:
    - measure – String, measure unique name;
    - tuple – String, defines the tuple;
    - type – String, sorting type ("asc", "desc" or "unsorted").
- columns – Array of objects. A list of hierarchies selected in the report slice for columns. Each object has the following properties:
    - caption – String, hierarchy caption;
    - dimensionName – String, dimension name;
    - filter – object with the information on filtering:
        - members – Array of hierarchy's members to be reflected/shown according to the applied filter;
        - negation – Boolean. Represents the filter on hierarchy's members. It tells the component to show the members of hierarchy specified in items (true) or to show all the members of hierarchy except the items (false); (This property has been added in version 1.7)
        - measure – Represents the filter on values. The name of the measure on which Top X or Bottom X filter will be applied.
        - quantity – Represents the filter on values. Number of elements to choose for the Top X filter if type is 'top' or for the Bottom X filter if type is 'bottom'.
        - type – Represents the filter type applied to the hierarchy. It can be: 'none' – filter is not applied to the hierarchy, 'members' – the filter on hierarchy's members is applied, 'top' – the filter Top X is applied on values, 'bottom' – the filter Bottom X is applied on values.
    - sortName – String, sorting type for the members ("asc", "desc" or "unsorted");
    - uniqueName – String, hierarchy unique name.
- conditions – Array of objects. Each object can have 3 properties:
    - column – Number, default is -1;

    - row – Number, default is -1;
    - formula – String.
- configuratorActive – Boolean. Indicates whether the Fields List is opened (true) or closed (false).
- configuratorButton – Boolean. Indicates whether the Fields List toggle button is visible (true) or non (false).
- configuratorMatchHeight (starting from v2.1) – Boolean. Indicates whether the Fields List will be the same height as the component (true) or it's height will be defined by its content amount (false). Default value is false.
- cube – String. Given catalog's cube's name of the OLAP data source (only for "microsoft analysis services", "mondrian", "iccube" data source types).
- data (starting from v2.2) – Property to set JSON data if they are already on the page. See example in addJSON().
- dataSourceInfo – String. The service info of the OLAP data source (only for "microsoft analysis services", "mondrian", "iccube" data source types).
- dataSourceType – String. Type of data source. The component supports the following types: "microsoft analysis services", "mondrian", "iccube", "csv", "ocsv" and "json".
- datePattern – String. It is used to format "date string" date fields ("type":"date string" in JSON, "ds+" prefix in CSV). A default pattern string is dd/MM/yyyy.
- dateTimePattern – String. It is used to format "datetime" date fields ("type":"datetime" in JSON, "dt+" prefix in CSV). A default pattern string is dd/MM/yyyy HH:mm:ss.
- defaultHierarchySortName (starting from v2.0) – String. Sorting type for hierarchies' members ("asc", "desc" or "unsorted"). Default value is "asc".
- drillAll (starting from v2.1) – Boolean. Indicates whether all levels of all hierarchies in slice will be drilled down (true) or drilled up (false).
- drilledColumns – Array of objects. It is used to save and restore drilled down columns.
- drilledRows – Array of objects. It is used to save and restore drilled down rows.
- drillThrough (starting from v2.1) – Boolean. Indicates whether the drill through feature is enabled (true) or disabled (false). User can drill through by double-clicking the cell with value. This feature is available for CSV data sources only. Default value is true.
- editing (starting from v2.1) – Boolean. Indicates whether the editing feature is enabled (true) or disabled (false) on the Drill Through popup for CSV and JSON data sources. User will be able to double-click the cell and enter new value in it if the editing feature is enabled.
- expandAll – Boolean. Indicates whether all nodes in the data tree will be expanded (true) or collapsed (false) on the grid and on charts.
- expandedColumns – Array of objects. It is used to save and restore expanded columns.
- expandedRows – Array of objects. It is used to save and restore expanded rows.
- fieldSeparator – String. Defines specific fields separator to split CSV row (only for CSV data source type). There is no need to define it if CSV fields are separated by , or ;. This property is used if another char separates fields. For example, if you use TSV, where tab char is used to separate fields in row, fieldSeparator parameter should be defined the following way:

```
<dataSource type="csv">
  <filename>tab-data.csv</filename>
  <fieldSeparator><![CDATA[ ]]></fieldSeparator>
</dataSource>
```

- filename – String. The URL to CSV or JSON file or to server side script which generates CSV data or JSON data (only for CSV and JSON data source type).
- fitGridlines – Boolean. Indicates whether the gridlines are shown for all cells (false) or only non-empty (true).
- flatView (starting from v2.2) – Boolean. Defines whether data should be shown in the flat view (true) or not (false). Default value is false, which means that flat view is turned off. Report parameter tableType (deprecated in 2.2) was replaced with the new flatView property.
- formats – Array of objects. Each object represents the number formatting object and contains the following formatting parameters:

    - name – String. Default is "".
    - thousandsSeparator – String. Default is " " (space).
    - decimalSeparator – String. Default is ".".
    - decimalPlaces – Number. It defines how many decimals to show in the fractional part on a number, after the decimal separator. Default is -1, which means “no formatting”.
    - maxDecimalPlaces – Number. To show not more decimals than this property in the fractional part on a number. Default is -1, which means “no formatting”.
    - maxSymbols – Number. Max number of symbols in cell. Default is 20.
    - currencySymbol – String. The symbol which is shown near the value (currency symbol, hours, percent, etc.). Default is "".
    - currencySymbolAlign – String. The alignment of the currency symbol. It can be "left" or "right". Default is "left".
    - nullValue – String. It defines how to show null values in grid. Default is "".
    - infinityValue – String. It defines how to show infinity values in grid. Default is "Infinity".
    - divideByZeroValue – String. Default is "Infinity".
    - textAlign – String. Align of text: "right" or "left". Default is "right".
- gridTitle – String. Title of the grid.
- htmlFooter Deprecated Since 2.1: Please use footer in export options of API call exportTo().
- htmlHeader Deprecated Since 2.1: Please use header in export options of API call exportTo().
- ignoreQuotedLineBreaks (starting from v2.1) – Boolean. Indicates whether the line breaks in quotes will be ignored (true) in CSV files or not (false). Default value is true, which makes CSV parsing faster. Set it to false only if your data source has valuable for you line breaks in quotes. Please note that this might slow down CSV parsing a little bit.
- localSettingsUrl – String. Specifies the path to localization file.
- logoAlpha – Number. Specifies transparency of the logo.
- logoUrl – String. Specifies URL to the company’s logo.
- measures – Array of objects. A list of selected measures and those which have different properties from default ones. Each object has the following properties:
    - uniqueName – String, measure unique name.
    - active (optional) – Boolean value that defines whether the measure will be in the list of available values but not selected (false) or will be selected for the report (true).
    - aggregation (optional) — String, unique name of aggregation that will be applied to the measure ("sum", "count", "distinctcount", "average", "product", "min", "max", "percent", "percentofcolumn", "percentofrow", "index"). If it is calculated measure, it will be "none".
    - availableAggregations (optional) — Array of strings that represents the list of aggregation functions which can be applied to the current measure. If it is calculated measure, it will be [].
    - calculated (optional) — Boolean value that defines whether measure is calculated.
    - caption (optional) – String, measure caption.
    - formula (optional) – String that represents the formula that can contain the following operations: +, -, *, /; other measures can be addressed using measure unique name and aggregation function, for example sum("Price") or max("Order"). Pivot supports the following aggregation functions for CSV data source: "sum", "count", "distinctcount", "average", "product", "min", "max", "percent", "percentofcolumn", "percentofrow", "index".
    - format (optional) – String, name of number formatting.
    - grandTotalCaption (optional) – String, measure grand total caption.
- pages – Array of objects. A list of hierarchies selected in the report slice for pages (“Report Filter”). Each object has the following properties:
    - caption – String, hierarchy caption;
    - dimensionName – String, dimension name;
    - filter – object with the information on filtering:
        - members – Array of hierarchy’s members to be reflected/shown according to the applied filter;
        - negation – Boolean. It tells the component to show the members of hierarchy specified in items (true) or to show all the members of hierarchy except the items (false); (This property has been added in version 1.7)

    - sortName – String, sorting type for the members ("asc", "desc" or "unsorted");
    - uniqueName – String, hierarchy unique name.
- pdfFontUrl Deprecated Since 2.1: Please use fontUrl in export options of API call exportTo().
- pdfFooter Deprecated Since 2.1: Please use footer in export options of API call exportTo().
- pdfHeader Deprecated Since 2.1: Please use header in export options of API call exportTo().
- providerUrl – String.
- proxyUrl – String. The path to proxy URL to the OLAP data source, such as Microsoft Analysis Services, Mondrian, icCube (only for "microsoft analysis services", "mondrian", "iccube" data source types).
- recordsetDelimiter – String. Defines which char is used in CSV to denote the end of CSV row (only for CSV data source type). Default value is "?".
- rowSizes – Array of Numbers. It is used to save and restore rows height in report.
- rowSorting – Object. Describes sorting on rows. Object has 3 properties:
    - measure – String, measure unique name;
    - tuple – String, defines the tuple;
    - type – String, sorting type ("asc", "desc" or "unsorted").
- rows – Array of objects. A list of hierarchies selected in the report slice for rows. Each object has the following properties:
    - caption – String, hierarchy caption;
    - dimensionName – String, dimension name;
    - filter – object with the information on filtering:
        - members – Array of hierarchy's members to be reflected/shown according to the applied filter;
        - negation – Boolean. Represents the filter on hierarchy's members. It tells the component to show the members of hierarchy specified in items (true) or to show all the members of hierarchy except the items (false); (This property has been added in version 1.7)
        - measure – Represents the filter on values. The name of the measure on which Top X or Bottom X filter will be applied.
        - quantity – Represents the filter on values. Number of elements to choose for the Top X filter if type is 'top' or for the Bottom X filter if type is 'bottom'.
        - type – Represents the filter type applied to the hierarchy. It can be: 'none' – filter is not applied to the hierarchy, 'members' – the filter on hierarchy's members is applied, 'top' – the filter Top X is applied on values, 'bottom' – the filter Bottom X is applied on values.
    - sortName – String, sorting type for the members ("asc", "desc" or "unsorted");
    - uniqueName – String, hierarchy unique name.
- saveReportUrl – String. Specifies path to the server-side script which can save reports.
- showAggregations (starting from v2.0) – Boolean. Indicates whether the aggregation selection control is visible (true) or not (false) for measures on Fields List. Default value is true.
- showCalculatedValuesButton (starting from v2.2) – Boolean. Controls the visibility of "Add calculated value" in Fields List. Default value is true.
- showChartLegendButton (starting from v2.2) – Boolean. Indicates whether the button to show/hide the legend on charts is visible. Default value is false which means that the legend is always visible, without the button that hides it.
- showChartMeasures (starting from v2.2) – Boolean. Hides all dropdowns on the top of charts if you want to show simple chart without controls or you want to save space. Default value is true – the dropdowns are visible by default, as it was in previous versions.
- showChartsWarning – Boolean. Indicates whether the warning are shown if data is too big for charts.
- showChartZeroValues – Boolean. Indicates whether to show zero values on charts (true) or not (false). Default value is true.
- showDefaultSlice (starting from v2.2) – Boolean. Defines whether the component selects a default slice for the report with empty slice (when nothing is set in rows, columns, pages and measures). If true, the first hierarchy from data goes to rows and the first measure goes to columns in the default slice. To avoid this default behavior, please set this property to false. Default value is true.
- showFilter – Boolean. Indicates whether the opening columns/rows filter controls and page filter controls are visible (true) or not (false) on the grid. Default value is true.
- showFilterInCharts (starting from v2.2) – Boolean. Indicates whether the opening columns and rows filter

controls are visible (true) or not (false) on the charts. Default value is true.
- showFiltersExcel Deprecated Since 2.1: Please use showFilters in export options of API call exportTo().
- showGrandTotals – String. Specifies how to show grand totals: in rows ("rows"), in columns ("columns"), in rows and columns ("on") or hide them ("off"). Default value is "on".
- showHeaders – Boolean. Indicates whether the spreadsheet headers are visible (true) or not (false).
- showHierarchies – Boolean. Specifies how to show drillable hierarchy cells on the grid: with link on the right (true) or with icon on the lift (false). Default value is true.
- showHierarchyCaptions – Boolean. Indicates whether the hierarchy captions are visible (true) or not (false) on the grid. Default value is true.
- showMemberProperties (starting from v2.1) – Boolean. Indicates whether the member properties for OLAP data source are visible (true) or not (false) on Fields List. Default value is false.
- showReportFiltersArea (starting from v2.2) – Boolean. Indicates whether the reports filtering cells on the grid should be visible (true) or not (false). Default value is true.
- showTotals – Boolean. Indicates whether the totals are visible (true) or not (false).
- sorting (starting from v2.0) – String. Indicates whether the sorting controls are visible in rows ("rows"), in columns ("columns"), in rows and columns ("on" or true) on the grid cells or not visible ("off" or false). Default value is "on".
- styleSheetName – String. Specifies URL to CSS file.
- tableType **(deprecated in 2.2)** – String. Starting from v1.9. Type of table: "pivot" or "flat". Default value is "pivot". "flat" table option is available for CSV and JSON data sources only.
- useOlapFormatting (starting from v2.1) – Boolean. Indicates whether the values from MSAS data source will be formatted according to the format defined in the cube (true) or not (false). Default value is false.
- viewType – String. Type of view to show: "grid" or "charts" or "grid_charts" (starting from v1.9).
- zoom – Number. The value of zooming. The value can be from 0.5 to 4. Default value is 1.

Examples

Swap two reports:

```
<div id="firstPivotContainer">The component will appear within this DIV.</div>
<div id="secondPivotContainer">The component will appear within this DIV.</div>
<script type="text/javascript" src="flexmonster/flexmonster.js"></script>
<script type="text/javascript">
  var pivot1 = flexmonster.embedPivotComponent("flexmonster/", "firstPivotContainer"
,"100%", "500", {configUrl : "firstconfig.xml"});
  var pivot2 = flexmonster.embedPivotComponent("flexmonster/", "secondPivotContainer
","100%", "500", {configUrl : "secondconfig.xml"});
</script>

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

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

Using "xmlstring" option (available starting from version 1.901):

```
<div id="pivotContainer">The component will appear within this DIV.</div>
<script type="text/javascript" src="flexmonster/flexmonster.js"></script>
<script type="text/javascript">
    flexmonster.embedPivotComponent("flexmonster/", "firstPivotContainer","100%", "5
00", {configUrl : "config.xml"});
</script>
<button onclick="getReportXML()">Get Report XML</button>
<script type="text/javascript">
    function getReportXML() {
        var reportXML = flexmonster.getReport("xmlstring");
        // parse, change or save for later use
        flexmonster.setReport(reportXML);
    }
</script>
```

See also

setReport
open
load
save

1.34. getRowHeight

getRowHeight(rowIndex:Number):Number

Version: 1.4

Returns row's height with specified index.

Parameters

- rowIndex – index of the row.

Returns

Height of the specified row.

Examples

How to find max row height in the grid:

```
var maxHeight = 0;
var rowCount = pivot.gridRowCount();
for (var i = 0; i < rowCount; i++) {
   var height = pivot.getRowHeight(i);
   if (maxHeight < height) maxHeight = height;
}
alert("Max height is: " + maxHeight);
```

See also

setRowHeight
getColumnWidth
setColumnWidth

1.35. getRows

getRows():Array

Version: 1.4

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

Returns

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

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

Examples

```
flexmonster.getRows();

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

See also

getAllHierarchies
getColumns
getPages
getAllMeasures
getMeasures

1.36. getSelectedCell

getSelectedCell():Object

Version: 1.4

Returns information about selected cell.

Returns

Object which contains information about selected cell. Object has the following parameters:

HTML5 version:

- columnIndex
- columns
- isTotal
- label
- measure
- rowIndex
- rows
- type – it can be "header" or "value"
- value

Flash/Flex version:

- columnIndex
- columns
- height
- isTotal
- label
- measure
- rowIndex
- rows
- styleName
- type – it can be "header" or "value"
- value
- width
- x
- y

See also

getCell
removeSelection
setSelectedCell

1.37. getSort

getSort(hierarchyName:String):String

Version: 1.4

Returns the sort type which is applied to the hierarchy.

Parameters

- hierarchyName – The name of the hierarchy.

Returns

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

Examples

```
flexmonster.getSort("[Country].[Country]");
```

See also

setSort
sortValues

1.38. getValue

getValue(rowIdx:int, colIdx:int):Number

Version: 1.6

Returns the cell value by row and column indexes.

Parameters

- rowIdx – row index
- colIdx – column index

Returns

Cell value

See also

getCell

1.39. getXMLACatalogs

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

Version: 1.5

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

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

Parameters

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

- password (optional) – the password to user account at server. This parameter is necessary to complete authentication process.

See also

getXMLADataSources with Example
getXMLAProviderName
getXMLACubes

1.40. getXMLACubes

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

Version: 1.5

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

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

Parameters

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

See also

getXMLADataSources with Example
getXMLAProviderName
getXMLACatalogs

1.41. getXMLADataSources

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

Version: 1.5

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

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

Parameters

- proxyUrl – the path to proxy URL to the OLAP data source, such as Microsoft Analysis Services,

Mondrian, icCube.
- callbackHandler – JS function which will be called when the component obtains a list of all data sources.
- username (optional) – the name of user account at server. This parameter is necessary to complete authentication process.
- password (optional) – the password to user account at server. This parameter is necessary to complete authentication process.

Examples

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

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

function diagnostic() {
  getDataSources();
}

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

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

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

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

See also

getXMLAProviderName
getXMLACatalogs
getXMLACubes

1.42. getXMLAProviderName

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

Version: 1.5

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

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

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

Parameters

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

See also

getXMLADataSources with Example
getXMLACatalogs
getXMLACubes

1.43. gridColumnCount

gridColumnCount():Number

Version: 1.4

Returns a number of active columns in the grid.

Returns

The number of active columns.

See also

gridRowCount

getCell

1.44. gridRowCount

gridRowCount():Number

Version: 1.4

Returns a number of active rows in the grid.

Returns

The number of active rows.

See also

gridColumnCount
getCell

1.45. load

load(url:String)

Version: 1.4

Loads report from the specified URL.

Parameters

- url – the URL to report XML file.

Examples

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

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

See also

open
save
getReport
setReport

1.46. open

open()

Version: 1.6

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

Examples

```
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 XML file to be loaded.
*/
```

See also

load
save
getReport
setReport

1.47. openFieldsList

openFieldsList()

Version: 1.4

Opens Fields List.

Examples

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

See also

closeFieldsList

1.48. percentZoom

percentZoom(percent:Number)

Version: 1.4

Zooms grid in percentage.

Equal to zoomTo() method, but argument is in percent.

Parameters

- percent – the percent of zooming. The value can be from 25 to 1000.

Examples

```
flexmonster.percentZoom(120);
```

See also

zoomTo

1.49. print

print()

Version: 1.4

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

See also

exportTo

1.50. refresh

refresh()

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()
- addStyleToMember()
- addUrlToMember()
- removeAllConditions()
- removeCondition()
- setChartTitle()
- setFormat()
- setGridTitle()
- setLabels()
- setOptions()

Examples

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

```
flexmonster.setGridTitle('grid title');
flexmonster.setChartTitle('chart title');
flexmonster.refresh();
```

See also

addCondition
addStyleToMember
addUrlToMember
removeAllConditions
removeCondition
setChartTitle
setFormat
setGridTitle
setLabels
setOptions

1.51. removeAllConditions

removeAllConditions()

Version: 1.5

Removes all conditional formatting rules.

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

Examples

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

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

See also

addCondition
getCondition
getAllConditions
removeCondition
refresh

1.52. removeAllMeasures

removeAllMeasures()

Version: 1.6

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

Removes all calculated measures.

Examples

To remove all calculated measures use removeMeasure(), as follows:

```
flexmonster.removeAllMeasures();
```

See also

addMeasure
getAllMeasures
getMeasures
removeMeasure

1.53. removeCondition

removeCondition(id:String)

Version: 1.5

Removes the conditional formatting rule by id.

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

Parameters

- id – the id of the conditional formatting rule.

Examples

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

```
var id = 9;
flexmonster.removeCondition(id);
flexmonster.refresh();
```

See also

addCondition
getCondition
getAllConditions
removeAllConditions
refresh

1.54. removeMeasure

removeMeasure(measureName:String)

Version: 1.5

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

Removes the calculated measure by measure unique name.

Parameters

- measureName – the measure unique name.

Examples

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

```
var measureName = "[Measures].[Average Price]";
flexmonster.removeMeasure(measureName);
```

See also

addMeasure
getAllMeasures
getMeasures
removeAllMeasures

1.55. removeSelection

removeSelection()

Version: 1.4

Removes a selection from cells on the grid.

See also

getSelectedCell
getCell
setSelectedCell

1.56. runQuery

runQuery(slice:Object)

Version: 1.6

Runs a query with specified slice and displays the result data. Use this method to rearrange hierarchies on the axes or to compose a new report based on the current data source.

Parameters

- slice – object which contains slice (rows, cols, pages, measures with filters, sorting etc.)

Examples

```
var slice =
{
 rows:
```

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

See also

getReport
setReport

1.57. save

save(filename:String, destination:String, callbackHandler:String, url:String, embedData:Boolean):String

Version: 1.4

You can use this method to save your current report to a specified location. Later if open it you will see this saved previously report as it was seen while saving it with all fields, applied filters and sortings same like you placed it.

Returns XML configuration of a report as a String.

Parameters

- filename – a default name of the file.
- destination (optional) – parameter defines how to save a generated file. File can be saved to "server", "file" or "clipboard". The default value is "file".
- callbackHandler (optional) – JS function which will be called when the report is saved.
- url (optional) – an URL to the server-side script which saves the generated file. The file is sent as a POST parameter. Use this parameter only if destination parameter is "server".
- embedData (optional) – specifies whether to save CSV/OCSV data within the report or not. Default value is false. (Available since version 2.120)

Examples

How to save a report to the local file system:

```
flexmonster.save('myreport.xml', 'file');
```

How to save a report to the server:

```
flexmonster.save('myreport.xml',
  'server', '', 'http://yourserver.com/yourscript.php');
```

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

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

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

How to save a report to clipboard:

```
flexmonster.save('myreport.xml', 'clipboard');
```

See also

load
getReport
setReport

1.58. setBottomX

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

Version: 1.4

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

Parameters

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

Example

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

flexmonster.getFilterProperties("[Category].[Category]");

/*
```

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

flexmonster.getFilter("[Category].[Category]");

/*
method getFilter() returns the following Array:
[
  {caption: "Clothing", hierarchyName: "[Category].[Category]", uniqueName: "[Catego
ry].[Category].[Clothing]"},
  {caption: "Accessories", hierarchyName: "[Category].[Category]", uniqueName: "[Cat
egory].[Category].[Accessories]"},
  {caption: "Components", hierarchyName: "[Category].[Category]", uniqueName: "[Cate
gory].[Category].[Components]"}
]
*/
```

See also

setTopX
clearFilter
getFilter
getFilterProperties
setFilter

1.59. setChartTitle

setChartTitle(title:String)

Version: 1.6

Sets a title for charts in runtime.

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

Parameters

- title – the charts' title.

See also

setGridTitle
refresh

1.60. setColumnWidth

setColumnWidth(columnIndex:Number, width:Number)

Version: 1.4

Sets the width of the specified column.

Parameters

- columnIndex – index of the column
- width – new width of the column

Examples

Set width of the first column to 200 pixels:

```
felxmonster.setColumnWidth(0, 200);
```

See also

getColumnWidth
getRowHeight
setRowHeight

1.61. setFilter

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

Version: 1.4

Sets the filter for the specified hierarchy.

Version: 1.7

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

Parameters

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

Examples

If you want to see data on 'Canada' and 'France':

```
flexmonster.setFilter("[Country].[Country]",
  [
    "[Country].[Country].[Canada]",
    "[Country].[Country].[France]"
  ]);
```

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

```
flexmonster.setFilter("[Product].[Color]",
  [
    "[Product].[Color].[yellow]"
  ],
  true);
```

instead of

```
flexmonster.setFilter("[Product].[Color]",
  [
    "[Product].[Color].[blue]",
    "[Product].[Color].[green]",
    "[Product].[Color].[red]",
    "[Product].[Color].[silver]",
    "[Product].[Color].[white]"
  ]);
```

See also

clearFilter
getFilter
setBottomX
setTopX

1.62. setFormat

setFormat(format:Object, measureName:String)

Version: 1.4

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

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

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

Parameters

format – the object that contains the number format parameters:

- name – String. It identifies the format in the report, thus, it should be unique. The default is "", which means that this number format is a default one and it is applied to all the measures for which the specific number format is not set.
- thousandsSeparator – String. The default is " " (space).
- decimalSeparator – String. The default is ".".
- decimalPlaces – Number. The exact number of decimals to show in the fractional part of a number after the decimal separator. The default is -1, which means that the number will be shown as is.
- maxDecimalPlaces – Number. The maximum number of decimals to show in the fractional part of a number after the decimal separator. The default is -1, which means the number will be shown as is.
- maxSymbols – Number. The maximum number of symbols in a cell. The default is 20.
- currencySymbol – String. The symbol which is shown near the value (currency symbol, hours, percent, etc.). The default is "".
- currencySymbolAlign – String. The alignment of the currency symbol. It can be "left" or "right". The default is "left".
- nullValue – String. It defines how to show null values in the grid. The default is "".
- infinityValue – String. It defines how to show infinity values in the grid. The default is "Infinity".
- divideByZeroValue – String. It defines how to show divided by zero values in the grid. The default is "Infinity".
- textAlign – String. The alignment of formatted values in cells on the grid: "right" or "left". The default is "right".

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

Examples

How to override a default number format in run time:

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

See also

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();
```

See also

getFormat
refresh

1.63. setGridTitle

setGridTitle(title:String)

Version: 1.6

Sets a title for the pivot table in runtime.

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

Parameters

- title – the grid's title.

See also

setChartTitle
refresh

1.64. setHandler

setHandler(eventName:String, functionName:String)

Version: 1.4

Sets a JS function as a handler for the component's event. There is the following component's events: creation complete, report load, update, filter open, fields list open, fields list close, cell click and fullscreen.

Parameters

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

See also

jsPivotCreationCompleteHandler
jsReportLoadedHandler
jsPivotUpdateHandler
jsFullScreenHandler
jsFieldsListOpenHandler
jsFieldsListCloseHandler
jsFilterOpenHandler
jsCellClickHandler

1.65. setLabels

setLabels(labels:Object)

Version: 1.6

Applies new values to localized labels.

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

Parameters

- labels – the object which contains pairs of key and value for any amount of labels

Examples

Change labels on “OK” and “Cancel” buttons:

```
var labels = flexmonster.getLabels();
labels["BUTTON_OK"] = "Accept";
labels["BUTTON_CANCEL"] = "Reject";
flexmonster.setLabels(labels);
flexmonster.refresh()
```

See also

getLabels
refresh

1.66. setOptions

setOptions(options:Object)

Version: 1.6

Sets the component’s options.

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

Parameters

options – the object which contains the list of component’s options. The following options are available:

- chartAutoRange – Boolean. Indicates whether the range of values on the charts is selected automatically or not. Default value is false.
- chartMultipleMeasures – Boolean. Starting from v1.9. Indicates whether to show multiple measures on charts. Default value is false.
- chartOneLevel (starting from v2.1 in HTML5 only) – Boolean. In case of drillable chart, defines whether chart shows elements only for the lowest hierarchy (true). Default value is false. Bar, Line, Scatter, Bar stack and Bar line charts support this feature in HTML5 version.
- chartType – String. Selected chart type: "bar", "line", "scatter", "pie", "bar_stack" and "bar_line" (starting from v1.9). Default value is "bar".
- classicView (starting from v2.1) – Boolean. Defines whether pivot table should be shown in the classic view (true) or not (false). Default value is false, which means that classic view is turned off.
- configuratorActive – Boolean. Indicates whether the Fields List is opened (true) or closed (false). Default value is false.

- configuratorButton – Boolean. Indicates whether the Fields List toggle button is visible (true) or not (false). Default value is true.
- configuratorMatchHeight (starting from v2.1) – Boolean. Indicates whether the Fields List will be the same height as the component (true) or it's height will be defined by its content amount (false). Default value is false.
- drillThrough (starting from v2.1) – Boolean. Indicates whether the drill through feature is enabled (true) or disabled (false). User can drill through by double-clicking the cell with value. This feature is available for CSV data sources only. Default value is true.
- editing (starting from v2.1) – Boolean. Indicates whether the editing feature is enabled (true) or disabled (false) on the Drill Through popup for CSV data sources. User will be able to double-click the cell and enter new value in it if the editing feature is enabled.
- fitGridlines – Boolean. Indicates whether the gridlines are shown for all cells (false) or only non-empty (true). Default value is false.
- flatView (starting from v2.2) – Boolean. Defines whether data should be shown in the flat view (true) or not (false). Default value is false, which means that flat view is turned off. Report parameter tableType (deprecated in 2.2) was replaced with the new flatView property.
- ignoreQuotedLineBreaks (starting from v2.1) – Boolean. Indicates whether the line breaks in quotes will be ignored (true) in CSV files or not (false). Default value is true, which makes CSV parsing faster. Set it to false only if your data source has valuable for you line breaks in quotes. Please note that this might slow down CSV parsing a little bit.
- showAggregations (starting from v2.0) – Boolean. Indicates whether the aggregation selection control is visible (true) or not (false) for measures on Fields List. Default value is true.
- showCalculatedValuesButton (starting from v2.2) – Boolean. Controls the visibility of "Add calculated value" in Fields List. Default value is true.
- showChartLegendButton (starting from v2.2) – Boolean. Indicates whether the button to show/hide the legend on charts is visible. Default value is false which means that the legend is always visible, without the button that hides it.
- showChartMeasures (starting from v2.2) – Boolean. Hides all dropdowns on the top of charts if you want to show simple chart without controls or you want to save space. Default value is true– the dropdowns are visible by default, as it was in previous versions.
- showChartsWarning – Boolean. Indicates whether the warning are shown if data is too big for charts. Default value is true.
- showDefaultSlice (starting from v2.2) – Boolean. Defines whether the component selects a default slice for the report with empty slice (when nothing is set in rows, columns, pages and measures). If true, the first hierarchy from data goes to rows and the first measure goes to columns in the default slice. To avoid this default behavior, please set this property to false. Default value is true.
- showFilter – Boolean. Indicates whether the opening columns/rows filter controls and page filter controls are visible (true) or not (false) on the grid. Default value is true.
- showFilterInCharts (starting from v2.2) – Boolean. Indicates whether the opening columns and rows filter controls are visible (true) or not (false) on the charts. Default value is true.
- showFiltersExcel Deprecated Since 2.1: Please use showFilters in export options of API call exportTo().
- showGrandTotals – String. Specifies how to show grand totals: in rows ('rows'), in columns ('columns'), in rows and columns ('on') or hide them ('off'). Default value is 'on'.
- showHeaders – Boolean. Indicates whether the spreadsheet headers are visible (true) or not (false). Default value is true.
- showMemberProperties (starting from v2.1) – Boolean. Indicates whether the member properties for OLAP data source are visible (true) or not (false) on Fields List. Default value is false.
- showReportFiltersArea (starting from v2.2) – Boolean. Indicates whether the reports filtering cells on the grid should be visible (true) or not (false). Default value is true.
- showTotals – Boolean. Indicates whether the totals are visible (true) or not (false). Default value is true.
- sorting (starting from v2.0) – String. Indicates whether the sorting controls are visible in rows ('rows'), in columns ('columns'), in rows and columns ('on' or true) on the grid cells or not visible ('off' or false). Default value is 'on'.
- useOlapFormatting (starting from v2.1) – Boolean. Indicates whether the values from MSAS data source will be formatted according to the format defined in the cube (true) or not (false). Default value is false.

Examples

How to turn off totals:

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

See also

getOptions
setStyle
refresh

1.67. setReport

setReport(report:*)

Version: 1.4

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

Parameters

- report – the object or XML string (starting from v1.901) which describes the report and contains all its properties. See the report object description in getReport().

Examples

Swap two reports:

```
<div id="firstPivotContainer">The component will appear within this DIV.</div>
<div id="secondPivotContainer">The component will appear within this DIV.</div>
<script type="text/javascript" src="flexmonster/flexmonster.js"></script>
<script type="text/javascript">
  var pivot1 = flexmonster.embedPivotComponent("flexmonster/", "firstPivotContainer"
,"100%", "500", {configUrl : "firstconfig.xml"});
  var pivot2 = flexmonster.embedPivotComponent("flexmonster/", "secondPivotContainer
","100%", "500", {configUrl : "secondconfig.xml"});
</script>

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

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

Using "xmlstring" option (available starting from version 1.901):

```
<div id="pivotContainer">The component will appear within this DIV.</div>
<script type="text/javascript" src="flexmonster/flexmonster.js"></script>
<script type="text/javascript">
    flexmonster.embedPivotComponent("flexmonster/", "firstPivotContainer","100%", "5
00", {configUrl : "config.xml"});
</script>
<button onclick="getReportXML()">Get Report XML</button>
<script type="text/javascript">
    function getReportXML() {
        var reportXML = flexmonster.getReport("xmlstring");
        // parse, change or save for later use
        flexmonster.setReport(reportXML);
    }
</script>
```

See also

getReport
open
load
save

1.68. setRowHeight

setRowHeight(rowIndex:Number, height:Number)

Version: 1.4

Sets height of the specified row.

Parameters

- rowIndex – index of the row
- height – new height of the row

Examples

Set height of the first row to 50 pixels:

```
felxmonster.setRowHeight(0, 50);
```

See also

getRowHeight
getColumnWidth
setColumnWidth

1.69. setSelectedCell

setSelectedCell(rowIndex:Number, columnIndex:Number)

Version: 2.1

Sets the selection for the cell by row's and column's indexes.

Parameters

- rowIndex – index of the row
- columnIndex – index of the column

Examples

Select cell on row #3 and column #5:

```
felxmonster.setSelectedCell(3, 5);
```

See also

getSelectedCell
removeSelection

1.70. setSort

setSort(hierarchyName:String, sortType:String)

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".

Examples

```
flexmonster.setSort("[Country].[Country]", "desc");
```

See also

getSort
sortValues

1.71. setStyle

setStyle(url:String)

Version: 1.4

Only for Flash and Flex versions of the component.

Loads a style sheet from the specified URL and applies it to the component. This method is used to change the style of the component at runtime.

Parameters

- url – the URL to the style sheet (CSS file)

See also

getOptions
setOptions

1.72. setTopX

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

Version: 1.4

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

Parameters

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

Example

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

flexmonster.getFilterProperties("[Category].[Category]");

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

flexmonster.getFilter("[Category].[Category]");

/*
```

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

See also

setBottomX
clearFilter
getFilter
getFilterProperties
setFilter

1.73. showCharts

showCharts(type:String, multiple:Boolean)

Version: 1.4

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

Parameters

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

See also

showGrid
showGridAndCharts

1.74. showGrid

showGrid()

Version: 1.4

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

See also

showCharts

showGridAndCharts

1.75. showGridAndCharts

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

Version: 1.9

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

Parameters

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

See also

showCharts
showGrid

1.76. sortValues

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

Version: 1.4

Sorts values of the specified axis and measure.

Parameters

- axisName – String. The name of the axis to be sorted. It can be 'rows' or 'columns'.
- type – String. The type of sorting: 'asc' or 'desc'.
- tuple – Array. The tuple which defines the coordinate of row or column on the grid.
- measureName – String. Identifies the measure on which sorting will be applied.

Examples

```
flexmonster.sortValues(
  "rows",
  "desc",
  ["[Category].[Category].[Cars]"],
  "[Measures].[Price]"
  );
```

See also

getSort
setSort

1.77. zoomTo

zoomTo(num:Number)

Version: 1.4

Zooms grid to the specified value.

Parameters

- num – the value of zooming. The value can be from 0.25 to 10.

Examples

```
flexmonster.zoomTo(0.8);
```

See also

percentZoom

1.78. jsCellClickHandler

jsCellClickHandler(cell:Object)

Version: 1.4

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

Parameters

cell – the object which contains information about the clicked cell. Object contains the following parameters:

- columnIndex
- columns
- height
- hierarchy
- isTotal
- label
- measure
- member
- rowIndex
- rows
- styleName (Flash/Flex versions only)
- type – it can be "header" or "value"
- value
- width
- x
- y

1.79. jsFilterOpenHandler

jsFilterOpenHandler(params:Object)

Version: 1.4

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

Parameters

params – the object which contains the following filter window parameters:

HTML5 version:

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

Flash/Flex version:

- hierarchyName – the unique name of the hierarchy
- caption – the hierarchy's caption
- x – the x coordinate of the filter window
- y – the y coordinate of the filter window

See also

jsPivotCreationCompleteHandler

1.80. jsFieldsListCloseHandler

jsFieldsListCloseHandler()

Version: 1.6

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

See also

jsFieldsListOpenHandler

1.81. jsFieldsListOpenHandler

jsFieldsListOpenHandler()

Version: 1.6

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

See also

jsFieldsListCloseHandler

1.82. jsFullScreenHandler

jsFullScreenHandler(fullScreenMode:Boolean)

Version: 1.6

It is triggered when the component enters or leaves full-screen display mode.

Parameters

- fullScreenMode – Boolean. Indicates whether the component is in full-screen mode (true) or not (false).

1.83. jsPivotCreationCompleteHandler

jsPivotCreationCompleteHandler()

Version: 1.4

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 creation complete event is dispatched. In other words, when jsPivotCreationCompleteHandler handler is called you know that the component's creation completed and now you can use API calls.

Also, when jsPivotCreationCompleteHandler is called data structure is still loading if configUrl parameter was specified in embedPivotComponent(). To retrieve information about data source structure use jsPivotUpdateHandler. If you want to track changes in a report object, we recommend using jsReportChangeHandler.

Examples

Example 1: How to set JSON data using API call

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

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

```
      "level":"State",
      "parent": "Country"},
  "city":{"type":"level",
      "hierarchy": "Geography",
      "parent": "State"},
  "Price":0,
  "Quantity":{"type": "number"}
},
{
  "Color":"green",
  "M":"September",
  "W":"Wed",
  "country" : "Canada",
  "state" : "Ontario",
  "city" : "Toronto",
  "Price":174,
  "Quantity":22
},
{
  "Color":"red",
  "M":"March",
  "W":"Mon",
  "Time":"1000",
  "country" : "USA",
  "state" : "California",
  "city" : "Los Angeles",
  "Price":1664,
  "Quantity":19
},
{
  "Color":"red",
  "M":"January",
  "W":"Mon",
  "country" : "Canada",
  "state" : "Quebec",
  "city" : "Montreal",
  "Price":1190,
  "Quantity":292
},
{
  "Color":"green",
  "D":"04/08/2014",
  "M":"August",
  "W":"Fri",
  "Time":"1000",
  "country" : "USA",
  "state" : "California",
  "city" : "Los Angeles",
  "Price":1222,
  "Quantity":730
},
{
  "Color":"white",
  "M":"March",
  "W":"Wed",
```

```
  "country" : "USA",
  "state" : "California",
  "city" : "Los Angeles",
  "Price":7941,
  "Quantity":73
},
{
  "Color":"red",
  "M":"August",
  "W":"Wed",
  "country" : "Canada",
  "state" : "Ontario",
  "city" : "Toronto",
  "Price":6829,
  "Quantity":19
},
{
  "Color":"green",
  "M":"January",
  "W":"Wed",
  "country" : "Canada",
  "state" : "Quebec",
  "city" : "Montreal",
  "Price":2995,
  "Quantity":98
},
{
  "Color":"white",
  "M":"February",
  "W":"Mon",
  "country" : "USA",
  "state" : "California",
  "city" : "Los Angeles",
  "Price":2471,
  "Quantity":93
},
{
  "Color":"yellow",
  "M":"March",
  "W":"Fri",
  "country" : "Canada",
  "state" : "Ontario",
  "city" : "Toronto",
  "Price":6650,
  "Quantity":40
},
{
  "Color":"blue",
  "M":"February",
  "W":"Wed",
  "country" : "Canada",
  "state" : "Ontario",
  "city" : "Toronto",
  "Price":865,
  "Quantity":45
```

```
  },
  {
    "Color":"purple",
    "M":"August",
    "W":"Wed",
    "country" : "Canada",
    "state" : "Quebec",
    "city" : "Montreal",
    "Price":511,
    "Quantity":46
  },
  {
    "Color":"blue",
    "M":"September",
    "W":"Mon",
    "country" : "Canada",
    "state" : "Quebec",
    "city" : "Montreal",
    "Price":981,
    "Quantity":18
  },
  {
    "Color":"blue",
    "M":"September",
    "W":"Fri",
    "country" : "Canada",
    "state" : "Ontario",
    "city" : "Toronto",
    "Price":284,
    "Quantity":24
  }
];

/**
* The data can be set using addJSON() and setReport() API calls
* only after creationComplete event is dispatched by Flexmonster Component
*/
function pivotCreationCompleteHandler() {
  flexmonsterAddJSON();
}

flexmonster.embedPivotComponent("flexmonster/", "pivotContainer", "100%", "515", {
  jsPivotCreationCompleteHandler: "pivotCreationCompleteHandler"
});

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

```
  flexmonster.runQuery(slice);
}
```

Example 2: How to define report from JS

Defining a report from JS after the component is created:

```
var pivot = flexmonster.embedPivotComponent("", "pivotContainer", "100%", "600",
 {jsPivotCreationCompleteHandler: "onPivotCreationCompleteHandler"}, true);

function onPivotCreationCompleteHandler() {
 var report = {
  dataSourceType: "CSV",
  filename: "data.csv",
  rows:
  [
   {uniqueName: "[Country].[Country]"},
   {uniqueName: "[Measures]"}
  ],
  columns:
  [
   {uniqueName: "[Color].[Color]"}
  ],
  measures:
  [
   {uniqueName: "[Measures].[Price]"}
  ],
  gridTitle: "Report set via JS API"
 };
 pivot.setReport(report);
}
```

See also

embedPivotComponent
jsPivotUpdateHandler
jsReportChangeHandler

1.84. jsPivotUpdateHandler

jsPivotUpdateHandler()

Version: 1.6

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 jsReportChangeHandler instead.

Examples

```
flexmonster.embedPivotComponent("flexmonster/", "pivotContainer", "100%", "350",
  {jsPivotUpdateHandler: "pivotUpdateHandler"});

function pivotUpdateHandler() {
  alert("Updated!");
}
```

See also

jsPivotCreationCompleteHandler
jsReportChangeHandler

1.85. jsReportChangeHandler

jsReportChangeHandler()

Version: 1.9

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

Examples

```
flexmonster.embedPivotComponent("flexmonster/", "pivotContainer", "100%", "350",
  {jsReportChangeHandler: "pivotReportChangeHandler"});

function pivotReportChangeHandler() {
  alert("Report is changed!");
}
```

See also

jsPivotCreationCompleteHandler
jsPivotUpdateHandler
jsReportLoadedHandler

1.86. jsReportLoadedHandler

jsReportLoadedHandler()

Version: 1.8

It is triggered when the component loaded a report.

Examples

```
flexmonster.embedPivotComponent("flexmonster/", "pivotContainer", "100%", "350",
```

```
  {jsReportLoadedHandler: "pivotReportLoadedHandler"});

function pivotReportLoadedHandler() {
  alert("Report is ready!");
}
```

See also

jsPivotCreationCompleteHandler
jsPivotUpdateHandler
jsReportChangeHandler