Pivot Table & Charts Component Documentation

by Flexmonster

1. Getting started

Welcome to JavaScript Pivot Table & Charts component — web client side component designed to view, analyze and manage multidimensional data online.

It is an equivalent of Microsoft Excel Pivot Table and can be integrated into any web applications or websites.

Flexmonster is a cross-platform web Component that supports Microsoft Analysis Services OLAP cubes, Mondrian, icCube, JSON, SQL (MS SQL, MySQL and others) databases or static CSV files. This solution allows you to work extremely quickly with really large data volumes.

The data is represented in compact yet interactive visual reports – multidimensional tables and charts that are fully customizable for client's needs.

The Component has 2 main advantages from competing types of applications:

- Usability and speed of the desktop application,
- Mobility and scalability of the web-based application.

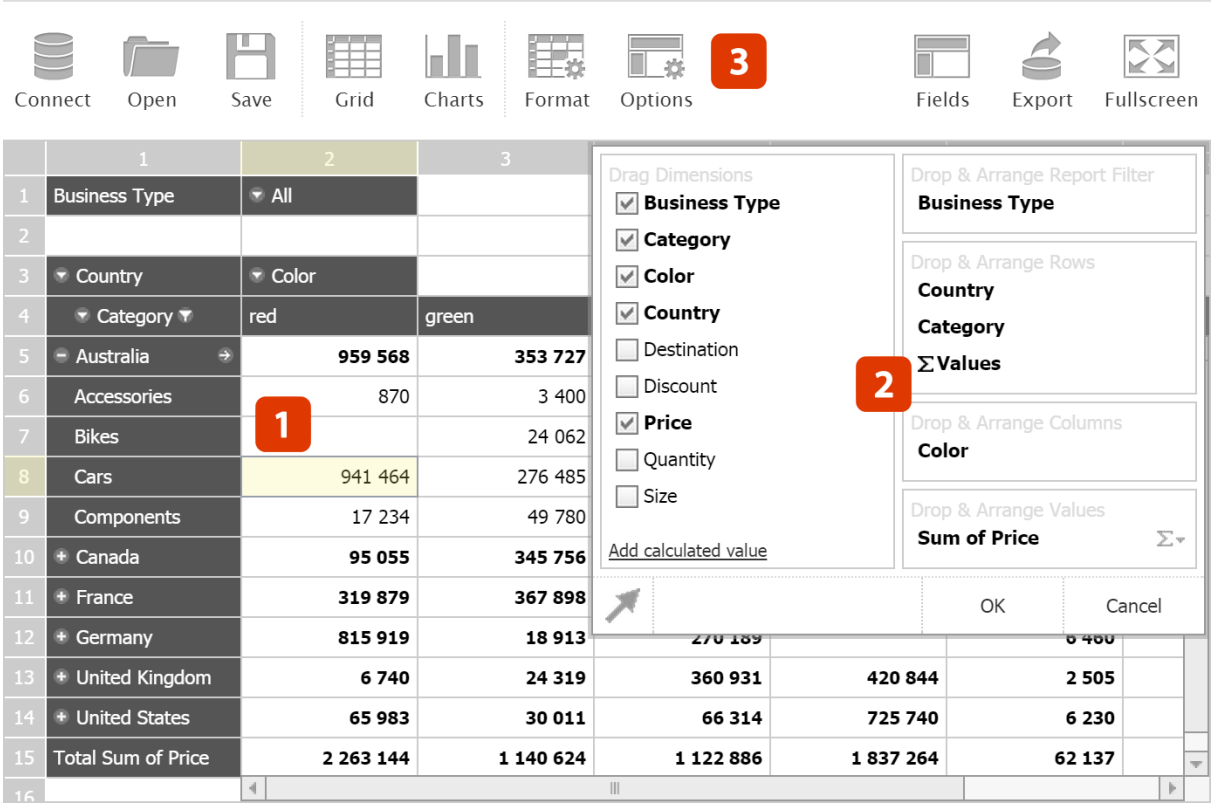
The tool is ideal for business intelligence data analysis. The world's most respected brands use Flexmonster Pivot Table to create and analyze reports online.

Flexmonster Pivot Table & Charts can use different data sources and can be embedded in various ways:

Database OLAP cube	?	JSON CSV XMLA	?	**Pivot Table & Charts**	HTML5 Page Flex/AIR Application

This guide will provide you with quick introduction to Pivot Table & Charts Component, information about its features, and its ways of usage.
Pivot Table enables you to analyze numerical data. With Pivot Table, you can look at the same information in different ways with just a few mouse clicks.

Pivot Table and Charts



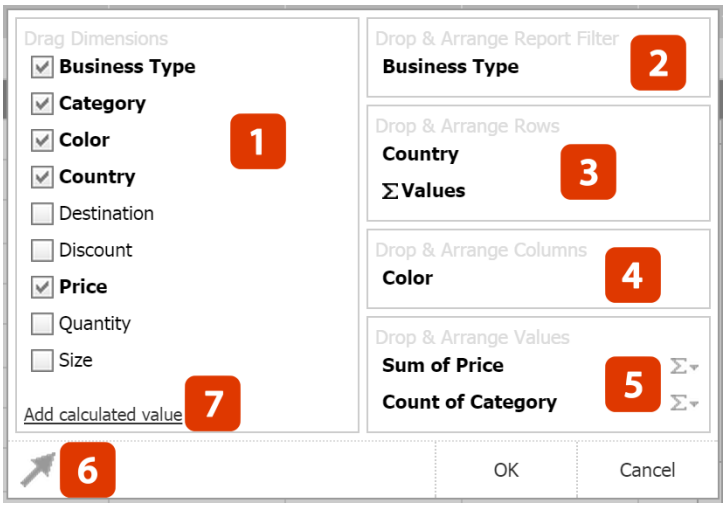
1. Pivot Grid or Pivot Chart
2. Fields List
3. Toolbar

Fields List

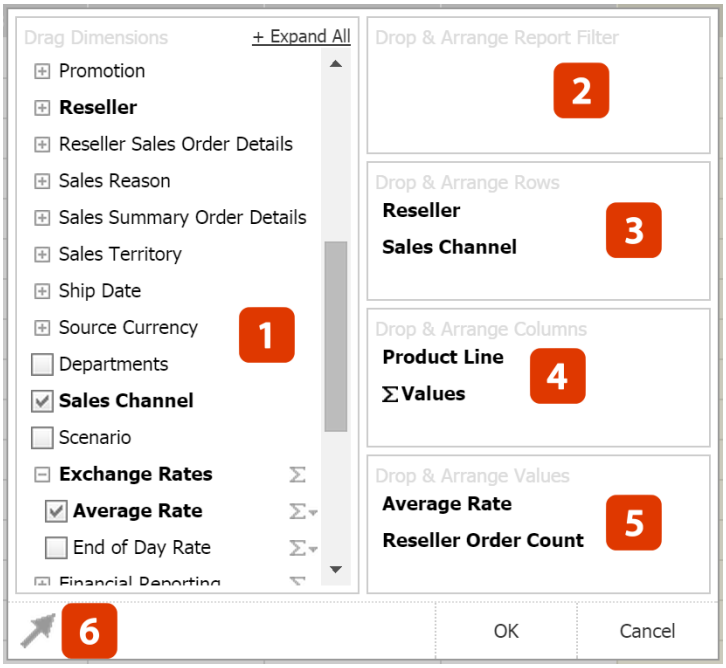
You can configure any view using Fields List with drag-and-drop.

1. Fields and Values to choose from
    - Dimensions/Folder tree in OLAP
    - Dimensions in CSV
2. Report Filter
3. Rows Fields
4. Columns Fields
5. Values
6. Show/Hide Fields List view button
7. Add calculated value in CSV (there is no such an option for reports based on OLAP data source)

CSV version



OLAP version



1.1. System requirements

HTML5 version

- Web Browser. Minimum browser requirements are listed below. Whichever browser you prefer, it is recommended that you use the most up-to-date version available for the best experience.
    - Chrome 12+
    - Firefox 15+
    - Internet Explorer 10+
    - Opera 15+
    - Safari 6.1+
    - iOS Safari 5.1.1+
- JavaScript must be enabled.
- Minimal recommended size for Pivot Component is 400×300px.

Flash/JS version

- Web Browser. Minimum browser requirements are listed below.
  - Chrome 4+
  - Firefox 3.6+
  - Internet Explorer 8+
  - Opera 15+
  - Safari 5+
- Adobe Flash Player 10.3+
  Adobe Flash Player is a cross-platform browser plug-in that delivers breakthrough Web experiences to over 99% of Internet users.
  You can download at http://www.adobe.com/go/getflash.
- JavaScript must be enabled.
- To use Full-screen mode, flash player should be allowed to use it. More details in Flash player documentation.
- Minimal recommended size for Pivot Component is 400×300px.

Flex version

- Flex SDK 4.1+
- Minimal recommended size for Pivot Component is 400×300px.

1.2. Installation

Embed into the web application

Here you can find a guide through the process of embedding Pivot Table and Charts Component into your HTML using flexmonster.js library.

To get your Pivot Table component embedded you should accomplish the following steps:

1. Download a free trial of the component and extract the files from the downloaded package. There are separate packages for HTML5 and Flash versions of the component. Please choose what version you would like to use in your web application.
2. Copy flexmonster/ forder into your web project root to your server.
3. Include flexmonster.js into your HTML page:

```
<script src="flexmonster/flexmonster.js"></script>
```

4. Create container for the component:

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

5. Add simple script to embed the component:

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

<script>
    flexmonster.embedPivotComponent(
        "flexmonster/", "pivotContainer", "100%", "500", {
```

```
            licenseKey: "XXXX-XXXX-XXXX-XXXX-XXXX"
        }, true
    );
</script>
```

6. Now launch the page from browser — here you go! The component is embedded into your project.

If you need a troubleshooting with this process, please open browser console and check if you see any errors there.

embedPivotComponent() API call

This is the first API call you need to know.

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

It embeds the component into HTML page and allows you to provide it with all necessary information for the initialization by setting the following arguments:

- 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.
- 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.

1.3. Your first pivot table

This article explains how to create your first pivot table based on a simple report with JSON as a data source. JSON data source is good to display data which already is on your page. This is the most simple case of the configuration.

You already know how to install the component. Now it is time to see data in it and configure what you want to see in rows and in columns.

1. You already have in your HTML page the following empty pivot table component:

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

```
<script>
    flexmonster.embedPivotComponent(
        "flexmonster/", "pivotContainer", "100%", "500", {
            licenseKey: "XXXX-XXXX-XXXX-XXXX-XXXX"
        }, true
    );
</script>
```

2. Copy the following JSON and paste it into your HTML page:

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

<script>
    var jsonData = [
        {
            "Color" : "green",
            "country" : "Canada",
            "state" : "Ontario",
            "city" : "Toronto",
            "Price" : 174,
            "Quantity" : 22
        },
        {
            "Color" : "red",
            "country" : "USA",
            "state" : "California",
            "city" : "Los Angeles",
            "Price" : 166,
            "Quantity" : 19
        }
    ];

    flexmonster.embedPivotComponent(
        "flexmonster/", "pivotContainer", "100%", "500", {
            licenseKey: "XXXX-XXXX-XXXX-XXXX-XXXX"
        }, true
    );
</script>
```

3. Add dataSourceType and data properties to report object:

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

<script>
    var jsonData = [
        {
            "Color" : "green",
            "country" : "Canada",
            "state" : "Ontario",
            "city" : "Toronto",
            "Price" : 174,
            "Quantity" : 22
        },
```

```
        {
            "Color" : "red",
            "country" : "USA",
            "state" : "California",
            "city" : "Los Angeles",
            "Price" : 166,
            "Quantity" : 19
        }
    ];

    flexmonster.embedPivotComponent(
        "flexmonster/", "pivotContainer", "100%", "500", {
            dataSourceType: "json",
            data: jsonData,
            licenseKey: "XXXX-XXXX-XXXX-XXXX-XXXX"
        }, true
    );
</script>
```

4. Define a slice – fields that go to rows, go to columns and go to measures:

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

<script>
    var jsonData = [
        {
            "Color" : "green",
            "country" : "Canada",
            "state" : "Ontario",
            "city" : "Toronto",
            "Price" : 174,
            "Quantity" : 22
        },
        {
            "Color" : "red",
            "country" : "USA",
            "state" : "California",
            "city" : "Los Angeles",
            "Price" : 166,
            "Quantity" : 19
        }
    ];

    flexmonster.embedPivotComponent(
        "flexmonster/", "pivotContainer", "100%", "500", {
            dataSourceType: "json",
            data: jsonData,
            rows: [
                { uniqueName: "Color" },
                { uniqueName: "[Measures]" }
            ],
            columns: [
                { uniqueName: "country" }
            ],
```

```
            measures: [
                { uniqueName: "Price", aggregation: "sum" }
            ],
            licenseKey: "XXXX-XXXX-XXXX-XXXX-XXXX"
        }, true
    );
</script>
```

5. Now launch the page from browser — here you go! Your first pivot table based on JSON data is ready.

1.4. Managing license keys

All Flexmonster Pivot Table & Charts Component editions need a license key.

There are 2 ways to set a license key for the component:

- via licenseKey parameter in JavaScript or Flex code
- via TXT file – flexmonsterpivottablekey.txt

When you download a trial package, you receive a trial key within the package.

It is in component/ folder or in client/ folder (depending on the trial package you downloaded):

- HTML5 version – component/index.html or client/index.html – set inline via licenseKey
- Flash version – component/flexmonster/flexmonsterpivottablekey.txt
- Flex version – component/flex/src/flexmonster/flexmonsterpivottablekey.txt

When you purchase the license, the trial key can be replaced by the license key.

Types of keys

There are the following types of keys:

- **Trial key** is provided within the trial package to try the component. The component runs with a trial key **without any restrictions**, with all capabilities according to the downloaded edition. This key **is temporary** and has an expiration date. Usually, trial period lasts for 2 weeks.
- **Development license key** is provided for development purposes. It is applicable to your local host environment. Using development key, the component can be run locally on your computer **(localhost)** or on the server using **IP address**. We do not license on per-developer basis, thus, you need only one license for the whole development team that works for the same project. This key is issued right after the purchase. Development license key can be limited in time or perpetual – depending on the purchased license.
- **Production license key** is provided for the production environment (domain/URL) where the component runs (e.g. yourcompany.com). It is not obligatory to know the production domain at the moment of purchase. It is needed when you publish your application on the web, so you can ask for the production key later. This key is **tied up to your domain name**. Production license key can be limited in time or perpetual – depending on the purchased license.
- **Staging key** can be issued by request after Flexmonster license purchase for testing on the real domain, but not in production environment.

You will receive development key right after Flexmonster license purchase. Also, as soon as you know the target domain/URL, the production key will be issued.

Component's editions

Flexmonster Pivot Table & Charts Component is available in 4 editions, you can choose from: Professional HTML5, Enterprise Site HTML5, Enterprise Advanced HTML5 and Enterprise Premium HTML5.

There are some differences between them. Each edition has its own key which unlocks it.

Professional HTML5 edition is the simplest one but it already has a good set of instruments and features for analysis. Features available in Professional HTML5 edition are available in all other editions as well – in Site, Advanced and Premium. Features available in Enterprise Site HTML5 edition are available in two higher editions – in Advanced and in Premium. And so on. Please see the details below:

- **Professional HTML5 edition** works with limited size CSV data sources (up to 2Mb) and limited size JSON data (up to 2000 records). This edition has professional features to work.
- **Enterprise Site HTML5 edition** supports Mondrian, icCube, CSV and JSON data sources. CSV file has a limit up to 4Mb. JSON data has a limit up to 4000 records. This edition has professional and enterprise features to work.
- **Enterprise Advanced HTML5 edition** supports Microsoft Analysis Services, Mondrian, icCube, CSV and JSON data sources. It has no restrictions for CSV files size and JSON data size. All professional and enterprise features are fully supported.
- **Enterprise Premium HTML5 edition** supports all data sources and has all features. It includes premium 24/7 support.

For prices and detailed comparison of all features available for each component's edition please visit Pricing & Download page.

1.5. Updating to the latest version

Our customers have an opportunity to update Flexmonster Pivot Table & Charts component to the latest version. This is a very simple process requiring just a couple of steps:

- Go to our Pricing & Download page and download a trial package of corresponding version.
- Unzip the files from the package.
- **For HTML5**: Replace your html5-assets/, toolbar/ and flexmonster.js files with the new ones (find them in component/flexmonster/ directory or in client/flexmonster/ directory – depending on the trial package you downloaded).
- **For Flash/JS**: Replace your PivotTable.swf, toolbar/ and flexmonster.js files with the new ones (find them in component/flexmonster/ directory).
- **For Flex**: Replace your FlexPivotComponent.swc file with the new one (find it in component/flex/libs/ directory).
- To make sure you updated the component you can check the **component's version** by clicking on the grid and pressing Ctrl+Alt+i. The information about the version of the component and its edition will be shown in the pop-up.
- If you use our **Accelerator**, please find instructions in the separate section below.
- If you use our **Compressor**, please find instructions in the separate section below.

Now it's done. You are welcome to use the most recent version of Flexmonster Pivot Table & Charts with its newest features.

NOTE! Being our active customer (during 1 year after purchase or renewal payment) you are able to **update for free**. If being our client you would like to renew the maintenance, please contact us for more details.

Updating Flexmonster Accelerator

Microsoft Analysis Services / EXE version

- Stop Accelerator by closing the program.
- Replace flexmonster-proxy-ssas.exe with updated version from the package (find it in server/ directory).
- Start Accelerator by launching flexmonster-proxy-ssas.exe.

Microsoft Analysis Services / Windows Service version

- Reinstall Accelerator using updated MSI installer from the package (find it in server/installer/ directory).

Pentaho Mondrian

- Stop Accelerator by closing the program.
- Replace flexmonster-proxy-mondrian.jar with updated version from the package (find it in server/ directory).
- Start Accelerator by executing java -jar flexmonster-proxy-mondrian.jar in terminal.

Updating Flexmonster Compressor

.NET

- Replace Flexmonster.Compressor.dll with updated version from the package (find it in server/.net/ directory).

Java

- Replace flexmonster-compressor.jar with updated version from the package (find it in server/java/ directory).

PHP

- Replace flexmonster-compressor.php with updated version from the package (find it in server/php/ directory).

2. JSON data source

This article illustrates how to build a report based on JSON data source. JSON data source is good to display data which already is on your page. This is the most simple case of the configuration. Also, you can load JSON data from a file (please see Connect > To local JSON in the toolbar) or refer JSON file as data source.

The component supports a certain format of JSON – array of objects, where each object is an unordered set of name/value pairs. For example:

```
var jsonData = [
    {
        "Color" : "green",
        "country" : "Canada",
        "state" : "Ontario",
        "city" : "Toronto",
        "Price" : 174,
        "Quantity" : 22
    },
    {
        "Color" : "red",
```

```
        "country" : "USA",
        "state" : "California",
        "city" : "Los Angeles",
        "Price" : 166,
        "Quantity" : 19
    }
];
```

You need to accomplish the following steps to use JSON data as a data source in pivot table embedded into your project.

1. You already have in your HTML page the following empty pivot table component:

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

<script>
    flexmonster.embedPivotComponent(
        "flexmonster/", "pivotContainer", "100%", "500", {
            licenseKey: "XXXX-XXXX-XXXX-XXXX-XXXX"
        }, true
    );
</script>
```

2. Copy the following JSON and paste it into your HTML page:

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

<script>
    var jsonData = [
        {
            "Color" : "green",
            "country" : "Canada",
            "state" : "Ontario",
            "city" : "Toronto",
            "Price" : 174,
            "Quantity" : 22
        },
        {
            "Color" : "red",
            "country" : "USA",
            "state" : "California",
            "city" : "Los Angeles",
            "Price" : 166,
            "Quantity" : 19
        }
    ];

    flexmonster.embedPivotComponent(
        "flexmonster/", "pivotContainer", "100%", "500", {
            licenseKey: "XXXX-XXXX-XXXX-XXXX-XXXX"
        }, true
```

```
    );
</script>
```

3. Add dataSourceType and data properties to report object:

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

<script>
    var jsonData = [
        {
            "Color" : "green",
            "country" : "Canada",
            "state" : "Ontario",
            "city" : "Toronto",
            "Price" : 174,
            "Quantity" : 22
        },
        {
            "Color" : "red",
            "country" : "USA",
            "state" : "California",
            "city" : "Los Angeles",
            "Price" : 166,
            "Quantity" : 19
        }
    ];

    flexmonster.embedPivotComponent(
        "flexmonster/", "pivotContainer", "100%", "500", {
            dataSourceType: "json",
            data: jsonData,
            licenseKey: "XXXX-XXXX-XXXX-XXXX-XXXX"
        }, true
    );
</script>
```

4. Define a slice – fields that go to rows, go to columns and go to measures:

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

<script>
    var jsonData = [
        {
            "Color" : "green",
            "country" : "Canada",
            "state" : "Ontario",
            "city" : "Toronto",
            "Price" : 174,
            "Quantity" : 22
        },
        {
            "Color" : "red",
            "country" : "USA",
```

```
            "state" : "California",
            "city" : "Los Angeles",
            "Price" : 166,
            "Quantity" : 19
        }
    ];

    flexmonster.embedPivotComponent(
        "flexmonster/", "pivotContainer", "100%", "500", {
            dataSourceType: "json",
            data: jsonData,
            rows: [
                { uniqueName: "Color" },
                { uniqueName: "[Measures]" }
            ],
            columns: [
                { uniqueName: "country" }
            ],
            measures: [
                { uniqueName: "Price", aggregation: "sum" }
            ],
            licenseKey: "XXXX-XXXX-XXXX-XXXX-XXXX"
        }, true
    );
</script>
```

5. Now launch the page from browser — here you go! Pivot table based on JSON data is embedded into your project

Now you know how to build a report based on our sample JSON. The next step for you is to visualize your JSON data.

2.1. Data types in JSON

In addition, the first object in JSON array can be used to define data types, captions, group fields under one dimension, define hierarchy with 3 levels, 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.

For example, you can add the following first object in JSON array and see how it changes the report:

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

<script>
    var jsonData = [
        {
            "Color": {type: "string"},
            "country":{type: "level", hierarchy: "Geography", level: "Country"},
            "state": {type: "level", hierarchy: "Geography", level: "State", parent:
 "Country"},
            "city": {type: "level", hierarchy: "Geography", parent: "State"},
            "Price": {type: "number"},
            "Quantity": {type: "number"}
        },
        {
            "Color" : "green",
            "country" : "Canada",
            "state" : "Ontario",
            "city" : "Toronto",
            "Price" : 174,
            "Quantity" : 22
        },
        {
            "Color" : "red",
            "country" : "USA",
            "state" : "California",
            "city" : "Los Angeles",
            "Price" : 166,
            "Quantity" : 19
        }
    ];

    flexmonster.embedPivotComponent(
        "flexmonster/", "pivotContainer", "100%", "500", {
            dataSourceType: "json",
            data: jsonData,
            rows: [
```

```
                { uniqueName: "Color" },
                { uniqueName: "[Measures]" }
            ],
            columns: [
                { uniqueName: "country" }
            ],
            measures: [
                { uniqueName: "Price", aggregation: "sum" }
            ],
            licenseKey: "XXXX-XXXX-XXXX-XXXX-XXXX"
        }, true
    );
</script>
```

Supported date formats

To make date fields be interpreted as date, you should define data type, for example, "type":"date", "type":"date string", "type":"year/month/day" or "type":"year/quarter/month/day". Besides, data from these fields should have special date format to be understood properly.

Pivot table component supports ISO 8601 date (other formats may be used, but results can be unexpected). For example, "2016-03-20" (just date) or "2016-03-20T14:48:00" (date and time).

3. CSV data source

This article illustrates how to build a report based on CSV data source. CSV data source is good to display data which is exported from other sources (for instance, from Excel). Also, you can load CSV data from a file (please see Connect > To local CSV in the toolbar) or refer CSV file as data source.

You need to accomplish the following steps to use CSV data as a data source in pivot table embedded into your project.

1. You already have in your HTML page the following empty pivot table component:

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

   <script>
       flexmonster.embedPivotComponent(
           "flexmonster/", "pivotContainer", "100%", "500", {
               licenseKey: "XXXX-XXXX-XXXX-XXXX-XXXX"
           }, true
       );
   </script>
   ```

2. Let’s say you have the following CSV file called data.csv. To make it simple, copy this file to the flexmonster/ folder.

   ```
   Category,Size,Color,Destination,Business Type,Country,Price,Quantity,Discount
   Accessories,262 oz,red,Australia,Specialty Bike Shop,Australia,174,225,23
   Accessories,214 oz,yellow,Canada,Specialty Bike Shop,Canada,502,90,17
   Accessories,147 oz,white,France,Specialty Bike Shop,France,242,855,37
   ```

```
Accessories,112 oz,yellow,Germany,Specialty Bike Shop,Germany,102,897,42
```

3. Add dataSourceType and filename properties to report object:

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

<script>
    flexmonster.embedPivotComponent(
        "flexmonster/", "pivotContainer", "100%", "500", {
            dataSourceType: "csv",
            filename: "data.csv",
            licenseKey: "XXXX-XXXX-XXXX-XXXX-XXXX"
        }, true
    );
</script>
```

4. Define a slice – fields that go to rows, go to columns and go to measures:

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

<script>
    flexmonster.embedPivotComponent(
        "flexmonster/", "pivotContainer", "100%", "500", {
            dataSourceType: "csv",
            filename: "data.csv",
            rows: [
                { uniqueName: "Color" },
                { uniqueName: "[Measures]" }
            ],
            columns: [
                { uniqueName: "?ountry" }
            ],
            measures: [
                { uniqueName: "Price", aggregation: "sum" }
            ],
            licenseKey: "XXXX-XXXX-XXXX-XXXX-XXXX"
        }, true
    );
</script>
```

5. Now launch the page from browser — here you go! Pivot table based on CSV data is embedded into your project.

Now you know how to build a report based on our sample CSV. The next step for you is to visualize your CSV data.

3.1. Data types in CSV

Using CSV data source you can indicate how data will be interpreted by Pivot table component. For this, you can use special prefixes for columns names. Pivot table has the following prefixes for CSV files:

- + means that field is a dimension.

- - field is a value.
- m+ this prefix will indicate that the field is a month.
- w+ to indicate that the field is a day of the week.
- d+ field is a date. Such field will be split into 3 different fields: Year, Month, Day. Date formats that are supported by Pivot table is described below.
- D+ field is a date. You will see such date as a hierarchy: Year > Month > Day.
- D4+ field is a date. You will see such date as a hierarchy: Year > Quarter > Month > Day.
- ds+ field is a date. Such field will be formatted using date pattern (default is dd/MM/yyyy)
- t+ field is a time (measure). Such field will be formatted using HH:mm pattern
- dt+ field is a date (measure). Such field will be formatted using dd/MM/yyyy HH:mm:ss pattern
- id+ field is an id of the fact.

Here is the minimal CSV file which will treat Year as dimension, rather than numeric measure:

```
Country, +Year, Sales
US, 2010, 200
UK, 2010, 100
```

Supported date formats

To make date column be interpreted as date, you should use prefixes d+, D+ and D4+ for CSV columns. Besides, data from these columns should have special date format to be understood properly.

HTML5 component

HTML5 version of the component supports ISO 8601 date (other formats may be used, but results can be unexpected). For example, "2016-03-20" (just date) or "2016-03-20T14:48:00" (date and time).

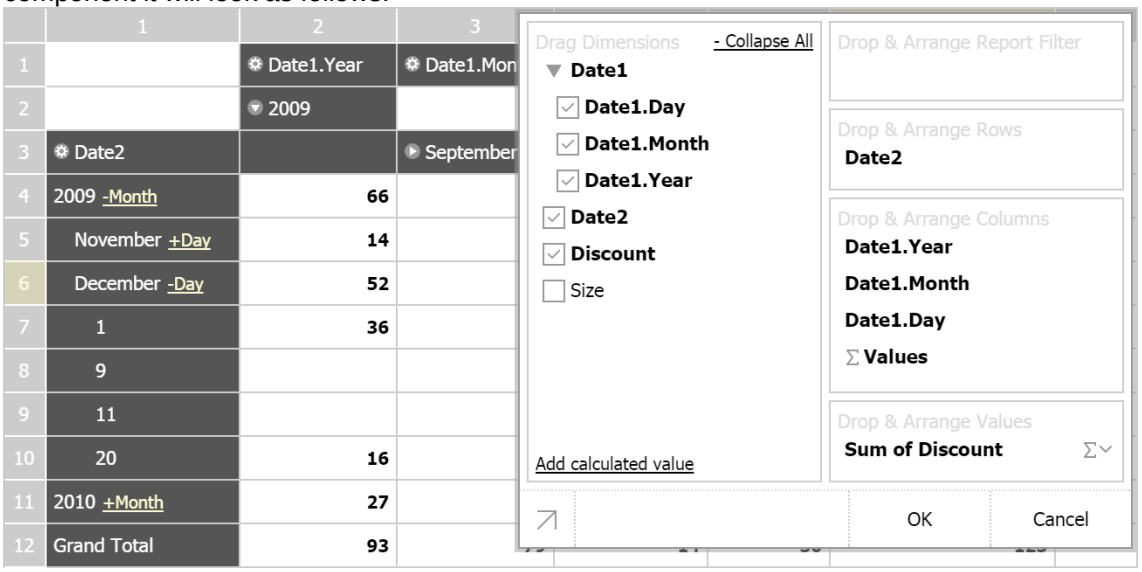
Here is an example of the CSV file with date columns – Date1 and Date2:

```
Size, Discount, d+Date1, D+Date2
214 oz, 14, 2009-11-01, 2009-11-09
214 oz, 12, 2010-12-09, 2009-12-09
212 oz, 36, 2009-09-01, 2009-12-01
212 oz, 27, 2009-09-01, 2010-12-02
212 oz, 18, 2010-11-09, 2009-12-11
212 oz, 16, 2009-09-01, 2009-12-20
```

The pivot table based on this CSV will look as follows:

	1	2	3	4	5	6
1		Date1.Year	Date1.Month	Date1.Day		
2		2009			2010	Totals
3	Date2		September	November		Total Sum of Discount
4	2009 -Month	66	52	14	30	96
5	November +Day	14		14		14
6	December -Day	52	52		30	82
7	1	36	36			36
8	9				12	12
9	11				18	18
10	20	16	16			16
11	2010 +Month	27	27			27
12	Grand Total	93	79	14	30	123
13						
14						

As you can see, Date1 column with prefix d+ is split into three separate fields — Year, Month, Day. In Pivot component it will look as follows:



Date2 column with D+ prefix is interpreted as a hierarchy that can be drilled down to months and days.

Flash/Flex component

Flash/Flex version of the component supports the following date formats:

- MMM/dd/yyyy – Dec/21/2012
- MMM dd yyyy – Dec 21 2012
- MM/dd/yyyy – 12/21/2012
- MM-dd-yyyy – 12-21-2012
- MM.dd.yyyy – 12.21.2012
- MM dd yyyy – 12 21 2012

Here is an example of the CSV file with date columns – Date1 and Date2:

```
Size, Discount, d+Date1, D+Date2
214 oz, 14, 11/1/2009, 11/9/2009
214 oz, 12, 12/9/2010, 12/9/2009
212 oz, 36, 9/1/2009, 12/1/2009
212 oz, 27, 9/1/2009, 12/2/2010
212 oz, 18, 9/11/2010, 12/11/2009
212 oz, 16, 9/1/2009, 12/20/2009
```

4. Connecting to SQL database

Everyone who has ever made reports or pivot tables with relational databases has most probably faced the problem with finding a suitable component that not only performs all required functions but also executes them fast. Flexmonster solved this problem and created **Flexmonster Data Compressor – a special server-side compression tool that helps you to increase data loading speed** from server to customer's browser.

Advantages of Flexmonster Compressor:

- Easy convert response from database with few lines of code
- Stream large datasets with minimum delay to the user
- Lightweight and simple format which does not require significant amount of RAM and CPU power
- Available for various platforms and databases

Supported databases

- Oracle
- MySQL
- Microsoft SQL Server
- PostgreSQL
- Other ADO.NET / ODBC / JDBC databases

Guides

Flexmonster Compressor is available for .NET, Java and PHP. Follow detailed tutorials for each technology:

- Connecting to relational database with .NET
- Connecting to relational database with Java
- Connecting to relational database with PHP

4.1. Connecting to relational database with .NET

Requirements

- Flexmonster Pivot Component v2.213 PROFESSIONAL edition or higher
- Microsoft .NET Framework 4 or higher
- Driver for the database

Supported databases

- Oracle Database – driver
- MySQL – driver
- Microsoft SQL Server – driver is built-in
- PostgreSQL – driver
- Other ADO.NET / ODBC databases

Step 1: Embed the component into your web page

Embedding Flexmonster Pivot is a simple copy-paste procedure that takes just minutes!

1. Copy contents of client/ folder into the web project root to your server.
2. Copy the code below and paste it into your HTML page.

```
<div id="pivotContainer">The component will appear here</div>
<script src="flexmonster/flexmonster.js"></script>
<script>
    flexmonster.embedPivotComponent(
        "flexmonster/", "pivotContainer", "100%", "500", {
            licenseKey: "XXXX-XXXX-XXXX-XXXX-XXXX"
        }, true
    );
</script>
```

Run your web page and see the empty table. The next step is to see your own data on the grid.

Step 2: Setup Data Compressor on the server

First of all, you need to add the following dependencies to your project:

- Appropriate database driver. Please note that Microsoft SQL Server driver is built-in.
- Flexmonster.Compressor.dll – it is located in the Download Package

Below is the connection and query sample for MS SQL Server:

```
string connectionString = "Server=localhost;Database=XXX;";
SqlConnection sqlConnection = new SqlConnection(connectionString);
sqlConnection.Open();
string query = "SELECT * FROM YYY";
DbCommand command = new SqlCommand(query, sqlConnection);
DbDataReader reader = command.ExecuteReader();
```

The following line of code will convert DbDataReader to compressed Stream:

```
Stream inputStream = Flexmonster.Compressor.CompressDb(reader);
```

Now, you can create a response from the Stream. For simple cases, it is enough to read all content:

```
string output = new StreamReader(inputStream).ReadToEnd();
```

Also, it is possible to create streaming response. It means that the end user will get the response almost without delay and server will use less amount of memory. It is recommended way for large datasets:

```
HttpResponseMessage response = Request.CreateResponse();
response.Content =
    new PushStreamContent((Stream outputStream, HttpContent content,
                           TransportContext context) =>
    {
        int length = 0;
        byte[] buffer = new byte[10240];
        while ((count = inputStream.Read(buffer, 0, buffer.Length)) > 0)
        {
            outputStream.Write(buffer, 0, length);
            outputStream.Flush();
        }
        outputStream.Close();
    }, new MediaTypeHeaderValue("text/plain")
);
```

Full project is available at the server/.net/ inside Download Package.

Step 3: Enable cross-origin resource sharing (CORS).

By default, browser prevents JavaScript from making requests across domain boundaries. CORS allows web applications to make cross-domain requests. Here are some useful links explain how to setup CORS on your server:

- IIS – CORS on IIS

Step 4: Configure report with your own data

Now it's time to configure Flexmonster Pivot Component on the web page. Let's create a minimal report for this (replace filename and other parameters with your specific values):

```
flexmonster.embedPivotComponent(
    "flexmonster/", "pivotContainer", "100%", "500",
    {
        dataSourceType: "ocsv",
        /* URL to the Data Compressor .NET */
        filename: "http://localhost:55772/api/flexmonster/get",
        licenseKey: "XXXX-XXXX-XXXX-XXXX-XXXX"
    }, true
);
```

Launch the web page from browser — here you go! Pivot table is embedded into your project.

Examples

Saving to file

```
string connectionString = "Server=localhost;Database=XXX;";
SqlConnection sqlConnection = new SqlConnection(connectionString);
sqlConnection.Open();
string query = "SELECT * FROM YYY";
DbCommand command = new SqlCommand(query, sqlConnection);
DbDataReader reader = command.ExecuteReader();

Stream inputStream = Flexmonster.Compressor.CompressDb(reader);
FileStream fileStream = File.Create("data.ocsv");
inputStream.CopyTo(fileStream);
fileStream.Close();
```

4.2. Connecting to relational database with Java

Requirements

- Flexmonster Pivot Component v2.213 PROFESSIONAL edition or higher
- Java 7 or higher
- Driver for the database

Supported databases

- Oracle Database – driver
- MySQL – driver
- Microsoft SQL Server – driver
- PostgreSQL – driver
- Other JDBC databases

Step 1: Embed the component into your web page

Embedding Flexmonster Pivot is a simple copy-paste procedure that takes just minutes!

1. Copy contents of client/ folder into the web project root to your server.
2. Copy the code below and paste it into your HTML page.

```
<div id="pivotContainer">The component will appear here</div>
<script src="flexmonster/flexmonster.js"></script>
<script>
    flexmonster.embedPivotComponent(
        "flexmonster/", "pivotContainer", "100%", "500", {
            licenseKey: "XXXX-XXXX-XXXX-XXXX-XXXX"
        }, true
    );
</script>
```

Run your web page and see the empty table. The next step is to see your own data on the grid.

Step 2: Setup Data Compressor on the server

First of all, you need to add the following dependencies to your project:

- Appropriate database driver
- flexmonster-compressor.jar – it is located in the Download Package

Below is the connection and query sample for MySQL database:

```
Class.forName("com.mysql.jdbc.Driver").newInstance();
String connectionString = "jdbc:mysql://localhost:3306/foodmart";
Connection connection = DriverManager.getConnection(connectionString, "user", "pass"
);
String query = "SELECT * FROM customer";
Statement statement = connection.createStatement();
ResultSet resultSet = statement.executeQuery(query);
```

The following line of code will convert ResultSet to compressed InputStream:

```
InputStream inputStream = Compressor.compressDb(resultSet);
```

Now, you can create a response from the InputStream. For simple cases, it is enough to read all content:

```
Scanner s = new Scanner(inputStream).useDelimiter("\\A");
String output = s.hasNext() ? s.next() : "";
```

Also, it is possible to create streaming response. It means that the end user will get the response almost without delay and server will use less amount of memory. It is recommended way for large datasets:

```
response.setContentType("text/plain");
OutputStream outputStream = response.getOutputStream();
int length = 0;
byte[] buffer = new byte[10240];
while ((length = inputStream.read(buffer)) > 0) {
    outputStream.write(buffer, 0, length);
    outputStream.flush();
}
```

Full project is available at the server/java/ inside Download Package.

Step 3: Enable cross-origin resource sharing (CORS).

By default, browser prevents JavaScript from making requests across domain boundaries. CORS allows web applications to make cross-domain requests. Here are some useful links explain how to setup CORS on your server:

- **Tomcat** – Configuration Reference (CORS Filter)

- **Jetty** – Jetty/Feature/Cross Origin Filter
- **JBOSS** – CORS Filter Installation

Step 4: Configure report with your own data

Now it's time to configure Flexmonster Pivot Component on the web page. Let's create a minimal report for this (replace filename and other parameters with your specific values):

```
flexmonster.embedPivotComponent(
    "flexmonster/", "pivotContainer", "100%", "500",
    {
        dataSourceType: "ocsv",
        /* URL to the Data Compressor Java */
        filename: "http://localhost:8400/FlexmonsterCompressor/get",
        licenseKey: "XXXX-XXXX-XXXX-XXXX-XXXX"
    }, true
);
```

Launch the web page from browser — here you go! Pivot table is embedded into your project.

Examples

Saving to file

```
Class.forName("com.mysql.jdbc.Driver").newInstance();
String connectionString = "jdbc:mysql://localhost:3306/foodmart";
Connection connection = DriverManager.getConnection(connectionString, "user", "pass"
);
String query = "SELECT * FROM customer";
Statement statement = connection.createStatement();
ResultSet resultSet = statement.executeQuery(query);

InputStream inputStream = Compressor.compressDb(resultSet);
OutputStream outputStream = new FileOutputStream("data.ocsv");
int length = 0;
byte[] buffer = new byte[10240];
while ((length = inputStream.read(buffer)) > 0) {
    outputStream.write(buffer, 0, length);
}
inputStream.close();
outputStream.close();
```

4.3. Connecting to relational database with PHP

Requirements

- Flexmonster Pivot Component v2.213 PROFESSIONAL edition or higher
- PHP 5.3+ (earlier versions may work, but not tested)

Supported databases

- MySQL
- Oracle Database
- Microsoft SQL Server
- PostgreSQL
- PDO databases
- Other databases supported by PHP

Step 1: Embed the component into your web page

Embedding Flexmonster Pivot is a simple copy-paste procedure that takes just minutes!

1. Copy contents of client/ folder into the web project root to your server.
2. Copy the code below and paste it into your HTML page.

```
<div id="pivotContainer">The component will appear here</div>
<script src="flexmonster/flexmonster.js"></script>
<script>
    flexmonster.embedPivotComponent(
        "flexmonster/", "pivotContainer", "100%", "500", {
            licenseKey: "XXXX-XXXX-XXXX-XXXX-XXXX"
        }, true
    );
</script>
```

Run your web page and see the empty table. The next step is to see your own data on the grid.

Step 2: Setup Data Compressor on the server

First of all, you need to add the following dependency to your project, flexmonster-compressor.php is located in the Download Package:

```
require_once("flexmonster-compressor.php");
```

Below is the connection and query sample for MySQL database:

```
mysql_connect($server, $username, $password);
mysql_select_db($dbname);
$result = mysql_query("SELECT * FROM customer");
```

The following line of code will start streaming compressed $result:

```
header('Content-Type: text/plain');
Compressor::compressMySql($result);
```

Full project is available at the server/php/ inside Download Package.

Step 3: Enable cross-origin resource sharing (CORS).

By default, browser prevents JavaScript from making requests across domain boundaries. CORS allows web applications to make cross-domain requests. Here are some useful links explain how to setup CORS on your server:

- IIS – CORS on IIS
- Apache – CORS on Apache

Also, you can add the following headers at the beginning of PHP file:

```
header('Access-Control-Allow-Origin: *');
header('Access-Control-Allow-Methods', 'OPTIONS,GET,PUT,POST,DELETE');
header('Access-Control-Allow-Headers', 'Content-Type');
```

Step 4: Configure report with your own data

Now it's time to configure Flexmonster Pivot Component on the web page. Let's create a minimal report for this (replace filename and other parameters with your specific values):

```
flexmonster.embedPivotComponent(
    "flexmonster/", "pivotContainer", "100%", "500",
    {
        dataSourceType: "ocsv",
        /* URL to the Data Compressor PHP */
        filename: "http://localhost/demo-compress-mysql.php",
        licenseKey: "XXXX-XXXX-XXXX-XXXX-XXXX"
    }, true
);
```

Launch the web page from browser — here you go! Pivot table is embedded into your project.

Examples

Here you can find few PHP examples of compressing different databases.

MySQL

```
header("Content-Type: text/plain");
require_once("flexmonster-compressor.php");

mysql_connect("localhost", "username", "password");
mysql_select_db("foodmart");
$result = mysql_query("SELECT * FROM customer");
Compressor::compressMySql($result);
```

MySQLi

```
header("Content-Type: text/plain");
require_once("flexmonster-compressor.php");

$mysqli = new mysqli("localhost", "username", "password", "foodmart");
$result = $mysqli->query("SELECT * FROM customer");
Compressor::compressMySqli($result);
```

Microsoft SQL Server

```
header("Content-Type: text/plain");
require_once("flexmonster-compressor.php");

$conn = sqlsrv_connect("localhost", array(
    "Database"=>"foodmart",
    "UID"=>"username",
    "PWD"=>"password"
  )
);
$result = sqlsrv_query($conn, "SELECT * FROM customer");
Compressor::compressSQLSRV($result);
```

PostgreSQL

```
header("Content-Type: text/plain");
require_once("flexmonster-compressor.php");

$conn = pg_connect("host=localhost dbname=foodmart user=username password=password")
;
$result = pg_query($conn, "SELECT * FROM customer");
Compressor::compressPostgreSQL($result);
```

Oracle Database

```
header("Content-Type: text/plain");
require_once("flexmonster-compressor.php");

$conn = oci_connect("username", "password", "localhost/XE");
$stid = oci_parse($conn, 'SELECT * FROM customer');
oci_execute($stid);
Compressor::compressOCI($stid);
```

PDO / MySQL example

```
header("Content-Type: text/plain");
require_once("flexmonster-compressor.php");

$conn = new PDO("mysql:host=localhost;dbname=foodmart", "username", "password");
```

```
$result = $conn->query("SELECT * from customer");
Compressor::compressPDO($result);
```

Other Databases / MySQL example

```
header("Content-Type: text/plain");
require_once("flexmonster-compressor.php");

mysql_connect("localhost", "username", "password");
mysql_select_db("foodmart");
$result = mysql_query("SELECT * FROM customer");

$header = array();
for ($i=0; $i < mysql_num_fields($result); $i++) {
  $header[$i] = array(
    'name' => mysql_field_name($result, $i),
    'type' => mysql_field_type($result, $i)
  );
}
$array = array();
$array[] = $header;
while ($row = mysql_fetch_row($result)) {
  $array[] = $row;
}
Compressor::compressArray($array);
```

Saving to file / MySQL example

```
header("Content-Type: text/plain");
require_once("flexmonster-compressor.php");

mysql_connect("localhost", "username", "password");
mysql_select_db("foodmart");
mysql_set_charset("utf8");
$result = mysql_query("SELECT * FROM customer");
$outFile = "data.ocsv";
Compressor::compressMySql($result, $outFile);
```

Troubleshooting

Data is not streaming

It is likely that output_buffering is enabled in the PHP configuration on your server and therefore data is not streaming. Please try to disable it in the php.ini for better performance:
output_buffering = Off (see http://php.net/manual/en/outcontrol.configuration.php).

5. Connecting to Microsoft Analysis Services

There are two ways to connect to Microsoft Analysis Services using Flexmonster Pivot Table:

1. via XMLA – an industry standard for data access in analytical systems
2. via Flexmonster Accelerator – special server-side utility developed by Flexmonster

If you already have configured XMLA (msmdpump.dll) it will be preferable to start with option #1. In case you do not have XMLA or you need some advanced features (increase loading speed, use credentials, etc.) – option #2 is a better choice.

Connecting to MS Analysis Services via XMLA

XMLA (XML for Analysis) – an industry standard for data access in analytical systems, such as OLAP and Data Mining. Please follow the steps below to configure a connection to Microsoft Analysis Services via XMLA.

Step 1: Embed the component into your web page

Embedding Flexmonster Pivot is simple copy-paste procedure that takes just minutes!

1. Copy contents of component/ folder into the web project root to your server.
2. Copy the code below and paste it into your HTML page.

```
<div id="pivotContainer">The component will appear here</div>
<script src="flexmonster/flexmonster.js"></script>
<script>
    flexmonster.embedPivotComponent(
        "flexmonster/", "pivotContainer", "100%", "500",
        {   /* empty report */
            licenseKey: "XXXX-XXXX-XXXX-XXXX-XXXX"
        }, true
    );
</script>
```

Run your web page and see the empty table. The next step is to see your own data on the grid.

Step 2: Configure XMLA access to the cube

If you have XMLA already configured please skip this step. Otherwise, please refer to the article that explains how to set up an HTTP endpoint for accessing an Analysis Services instance.

Step 3: Enable cross-origin resource sharing (CORS)

By default, browser prevents JavaScript from making requests across domain boundaries. CORS allows web applications to make cross-domain requests. Please read this step-by-step instruction to enable CORS for IIS to read data from Microsoft Analysis Services.

Step 4: Configure report with your own data

Now it's time to configure Flexmonster Pivot Component on the web page. Let's create a minimal report for this (replace proxyUrl, catalog and cube parameters with your specific values):

```
flexmonster.embedPivotComponent(
    "flexmonster/", "pivotContainer", "100%", "500",
    {
```

```
        dataSourceType: "microsoft analysis services",

        /* URL to msmdpump.dll */
        proxyUrl: "http://olap.flexmonster.com/olap/msmdpump.dll",

        /* Catalog name */
        catalog: "Adventure Works DW Standard Edition",

        /* Cube name */
        cube: "Adventure Works",

        licenseKey: "XXXX-XXXX-XXXX-XXXX-XXXX"
    }, true
);
```

Launch the web page from browser — here you go! Pivot table is embedded into your project.

Connecting to MS Analysis Services via Flexmonster Accelerator

Everyone who has ever worked with multidimensional databases analysis has most probably faced the problem with finding a suitable component that not only performs all required functions but also executes them fast. Flexmonster solved this problem and created **Flexmonster Accelerator for Microsoft Analysis Services** cubes – a special server-side proxy that helps you to increase data loading speed from server to customer's browser.

Working with OLAP cubes, a browser component is communicating with the server via XMLA protocol. It's no secret that the XMLA protocol is heavy and exchanges a lot of excessive information. Thus, it takes too much time and memory to load and process the data.

We replaced XMLA protocol and use direct requests from the Component to a server.

In this way, we have come up with solutions to two major problems that many of those who work with big data had faced to a different extent:

- We made big data transfer from server to browser enormously fast. Our tool allows you to transfer large multidimensional data in super easy and fast way. Reporting becomes more enjoyable and prompt for your end users.
- We greatly reduced the load on a browser memory.

Please refer to Getting started with Accelerator guide to find step-by-step instructions.

5.1. MS Analysis Services / Getting started with Accelerator

Everyone who has ever worked with multidimensional databases analysis has most probably faced the problem with finding a suitable component that not only performs all required functions but also executes them fast. Flexmonster solved this problem and created **Flexmonster Accelerator for Microsoft Analysis Services** cubes – a special server-side proxy that helps you to increase data loading speed from server to customer's browser.

Working with OLAP cubes, a browser component is communicating with the server via XMLA protocol. It's no secret that the XMLA protocol is heavy and exchanges a lot of excessive information. Thus, it takes too much time

and memory to load and process the data.

We replaced XMLA protocol and use direct requests from the Component to a server.

In this way, we have come up with solutions to two major problems that many of those who work with big data had faced to a different extent:

- We made big data transfer from server to browser enormously fast. Our tool allows you to transfer large multidimensional data in super easy and fast way. Reporting becomes more enjoyable and prompt for your end users.
- We greatly reduced the load on a browser memory.

Open Live Demo

Requirements

- Flexmonster Pivot Component v2.2 ADVANCED edition or higher
- Microsoft Analysis Services installed and configured
- Microsoft .NET Framework 4 or higher

Step 1: Configure Data Speed Accelerator on the server

The package contains following files:

- flexmonster-proxy-ssas.exe – server-side utility that handles connectivity between Microsoft Analysis Services and Flexmonster Pivot Component
- flexmonster.config – file that contains configuration parameters for the utility (connection string, port, etc.)

First of all, let's review flexmonster.config file. It contains following parameters:

- CONNECTION_STRING – connection string for Microsoft Analysis Services. Example: *Data Source=localhost;*. Required.
- PORT – port number for the proxy service endpoint. Optional. Default is 50005.
- CACHE_MEMORY_LIMIT – size of maximum RAM memory available for cache (in MB). Optional. Default is 0 (unlimited).
- CACHE_ENABLED – indicates whether the cache is enabled. Optional. Default is true. Available since version 2.211.

After configuring all the necessary options, Data Speed Accelerator is ready to be launched. Just run the flexmonster-proxy-ssas.exe with Administrator privileges.

You can check the Accelerator is up and running by navigating to its URL in the browser (i.e. http://localhost:50005).

Step 2: Open a port for Data Speed Accelerator in the firewall

If it's planned to allow connection to Data Speed Accelerator from outside of the server, you should open an appropriate port in the firewall. Default port number is 50005, but it may vary depending on PORT parameter in flexmonster.config file.

Step 3: Configure Flexmonster Pivot Component

Now it's time to configure the client – Flexmonster Pivot Component. Let's create a minimal configuration using JavaScript API (replace proxyUrl, catalog and cube parameters with your specific values):

```
var config = {
  dataSourceType: "microsoft analysis services",
  // URL to the Data Speed Accelerator
  proxyUrl: "http://localhost:50005",
  // Catalog name
  catalog: "Adventure Works DW Standard Edition",
  // Cube name
  cube: "Adventure Works",
  // Flag to use Data Speed Accelerator instead of XMLA protocol
  binary: true
};
flexmonster.embedPivotComponent("", "pivot-container", "100%", "500", config);
```

Also, it's possible to create XML report with the same configuration:

```
<config>
  <dataSource type="microsoft analysis services">
    <!-- URL to the Data Speed Accelerator -->
    <proxyUrl>http://localhost:50005</proxyUrl>
    <!-- Catalog name -->
    <catalog>Adventure Works DW Standard Edition</catalog>
    <!-- Cube name -->
    <cube>Adventure Works</cube>
    <!-- Flag to use Data Speed Accelerator instead of XMLA protocol -->
    <binary>true</binary>
  </dataSource>
</config>
```

Cache control

Usually, the cache is a great help, but the cache becomes out of date if the underlying database is changing. By default, cache is enabled and controlled by the Accelerator.

Also, it is possible to disable cache by the following parameter in flexmonster.config (available since version 2.211):

```
CACHE_ENABLED = false
```

5.2. Installing Accelerator as a Windows Service

Overview

There is an option to install Flexmonster Accelerator as a Windows Service instead of using Console application. Main benefits of running Accelerator as a Windows Service are:

- it's running in the background and out of sight

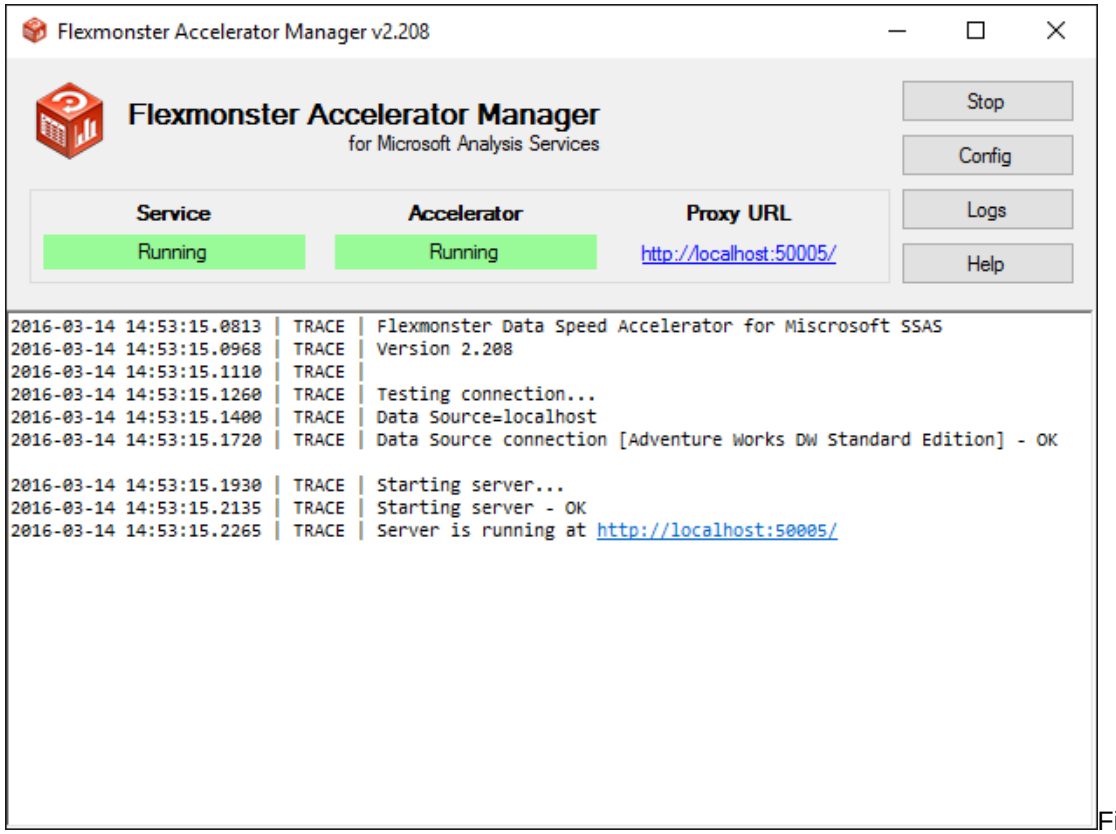
- it's started automatically on Windows startup
- it's harder for a user to inadvertently quit the application

Requirements

- Flexmonster Pivot Component v2.2 ADVANCED edition or higher
- Microsoft Analysis Services installed and configured
- Microsoft .NET Framework 4 or higher

Step 1: Configure Data Speed Accelerator on the server

Start installation using server/installer/Flexmonster Accelerator.msi setup file and follow the wizard. After successful installation you can run **Flexmonster Accelerator Manager**:



First of all, let's review Config file. It contains following parameters:

- CONNECTION_STRING – connection string for Microsoft Analysis Services. Example: Data Source=localhost;. Required.
- PORT – port number for the proxy service endpoint. Optional. Default is 50005.
- CACHE_MEMORY_LIMIT – size of maximum RAM memory available for cache (in MB). Optional. Default is 0 (unlimited).
- CACHE_ENABLED – indicates whether the cache is enabled. Optional. Default is true. Available since version 2.211.

You can check the Accelerator is up and running by navigating to its URL in the browser (i.e. http://localhost:50005).

After configuration you can close **Flexmonster Accelerator Manager** – service will continue working in the background.

Step 2: Open a port for Data Speed Accelerator in the firewall

If it's planned to allow connection to Data Speed Accelerator from outside of the server, you should open an appropriate port in the firewall. Default port number is 50005, but it may vary depending on PORT parameter in flexmonster.config file.

Step 3: Configure Flexmonster Pivot Component

Now it's time to configure the client – Flexmonster Pivot Component. Let's create a minimal configuration using JavaScript API (replace proxyUrl, catalog and cube parameters with your specific values):

```
var config = {
  dataSourceType: "microsoft analysis services",
  // URL to the Data Speed Accelerator
  proxyUrl: "http://localhost:50005",
  // Catalog name
  catalog: "Adventure Works DW Standard Edition",
  // Cube name
  cube: "Adventure Works",
  // Flag to use Data Speed Accelerator instead of XMLA protocol
  binary: true
};
flexmonster.embedPivotComponent("", "pivot-container", "100%", "500", config);
```

Also, it's possible to create XML report with the same configuration:

```
<config>
  <dataSource type="microsoft analysis services">
    <!-- URL to the Data Speed Accelerator -->
    <proxyUrl>http://localhost:50005</proxyUrl>
    <!-- Catalog name -->
    <catalog>Adventure Works DW Standard Edition</catalog>
    <!-- Cube name -->
    <cube>Adventure Works</cube>
    <!-- Flag to use Data Speed Accelerator instead of XMLA protocol -->
    <binary>true</binary>
  </dataSource>
</config>
```

5.3. MS Analysis Services / Configuring username/password protection

Overview

Flexmonster Pivot Component supports credentials to access the data source. It is commonly used for the following purposes:

- provide credentials for data source connectivity
- provide access for users with different roles and access levels

Here are the step-by-step instructions to configure secure connection with username/password protection and define different usernames on the client side.

Requirements

- Flexmonster Pivot Component v2.205 ADVANCED edition or higher
- Microsoft Analysis Services installed and configured
- Microsoft .NET Framework 4 or higher

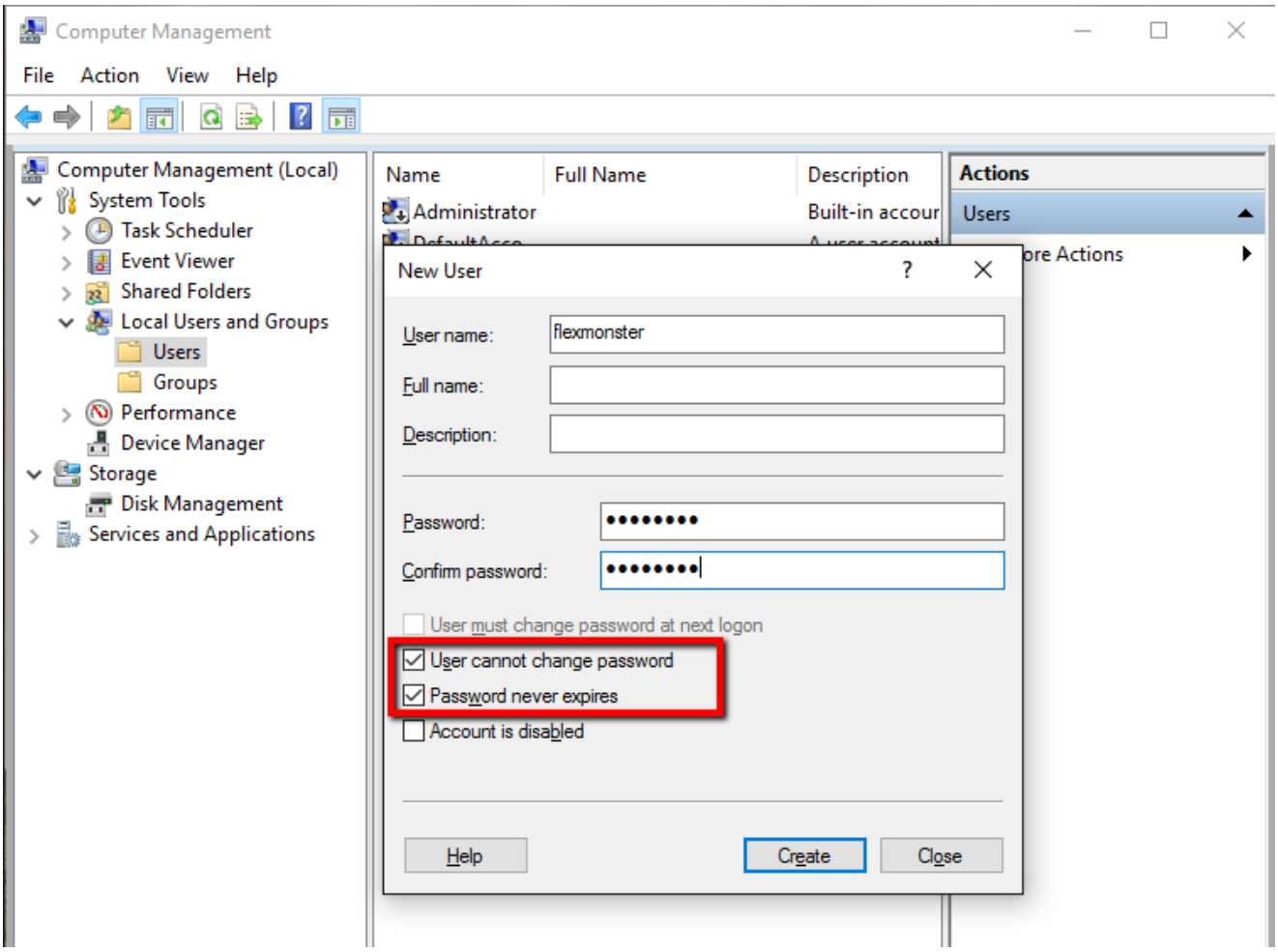
Step 1: Configure secure HTTP endpoint for accessing an Analysis Services

First of all, you need to set up HTTP endpoint for Analysis Services on IIS server. If it is already done, skip this step. Regarding IIS configuration, please refer to the MSDN documentation. In this step, you also configure authentication types.

Step 2: Create a default user for Accelerator

One username/password is used in the flexmonster.config file for the Accelerator on the server side. In general, Accelerator will use these credentials if client side report does not contain own credentials. Also, these credentials are used to start the Accelerator, check connection to the data source and connect anonymous users. So, it is recommended to create a user for Accelerator with minimum privileges and DO NOT use an admin account for this purpose.

Navigate to **Control Panel > Administrative Tools > Computer Management** and choose **System Tools > Local Users and Groups > Users**. Add a new user (i.e. flexmonster) by right-click as shown in the screenshot.



Step 3: Configure Accelerator

Open flexmonster.config and specify following CONNECTION_STRING parameters:

```
CONNECTION_STRING=Data Source=<endpoint_url>;UID=<username>;Password=<password>;
```

Where:

- <endpoint_url> – URL to HTTP endpoint (i.e. http://localhost/ssas/msmdpump.dll)
- <username> – username of the default user (i.e. flexmonster)
- <password> – password of the default user

Accelerator is ready to be launched and should successfully connect to the data source. Just run the flexmonster-proxy-ssas.exe with Administrator privileges.

You can check the Accelerator is up and running by navigating to its URL in the browser (i.e. http://localhost:50005).

Step 4: Configure Flexmonster Pivot Component

Note!
All data that is sent by HTTP is not encrypted and username/password can be intercepted. We recommend using credentials only with HTTPS. HTTPS encrypts all data that is sent from the client to Accelerator.
Read more at How to configure secure HTTPS connection*.*

Now, let's specify credentials for Analysis Services user in the configuration. Here is a minimal sample created with JavaScript API (replace proxyUrl, catalog, cube and credentials parameters with your specific values):

```
var config = {
    dataSourceType: "microsoft analysis services",
    proxyUrl: "http://localhost:50005",
    catalog: "Adventure Works DW Standard Edition",
    cube: "Adventure Works",
    binary: true,
    // Credentials of Analysis Services user
    username: "webuser",
    password: "1234"
}
flexmonster.embedPivotComponent("", "pivot-container","100%", "500", config);
```

Also, it's possible to create XML report with the same configuration:

```
<config>
  <dataSource type="microsoft analysis services">
    <proxyUrl>http://localhost:50005</proxyUrl>
    <catalog>Adventure Works DW Standard Edition</catalog>
    <cube>Adventure Works</cube>
    <binary>true</binary>
    <!-- Credentials of Analysis Services user -->
    <credentials>
      <username>webuser</username>
      <password>1234</password>
    </credentials>
  </dataSource>
</config>
```

5.4. MS Analysis Services / Configuring secure HTTPS connection

Overview

All data that is sent by HTTP is not encrypted and can be intercepted. That's why we have added an option to enable HTTPS for Accelerator. HTTPS encrypts all data that is sent from the client to Accelerator and vice versa. Please follow the steps below to configure secure HTTPS connection for your setup.

Requirements

- Flexmonster Pivot Component v2.206 ADVANCED edition or higher

- Microsoft Analysis Services installed and configured
- Microsoft .NET Framework 4 or higher
- **Valid and trusted SSL certificate**

Step 1: Register the server certificate

On an elevated console (“Run as administrator”), register the server certificate by running the following command:

```
netsh http add sslcert ipport=0.0.0.0:<port> certhash=<hash> appid='<app-guid>'
```

Where:

- <port> – the listening port (i.e. 50005); the special IP address 0.0.0.0 matches any IP address for the local machine
- <hash> – the certificate’s SHA-1 hash, represented in hexadecimal, without spaces
- <app-guid> – any GUID (i.e. {00000000-0000-0000-0000-000000000000}), used to identify the owning application

In case of any errors, please verify that the SSL certificate is properly installed in the Personal store and contains private key assigned.
Aslo, please note that appid parameter contains single quotes and curly braces ('{}').

Step 2: Enable HTTPS in Accelerator

After you have the certificate registered let’s enable the HTTPS in the Accelerator’s config. Open flexmonster.config file and modify/add the HTTPS parameter as follows:

```
HTTPS=true
```

Available values for HTTPS parameter is true or false. By default HTTPS is disabled (false).

Data Speed Accelerator is ready to be launched. Just run the flexmonster-proxy-ssas.exe with Administrator privileges.

You can check the Accelerator is up and running by navigating to its URL in the browser (i.e. https://localhost:50005).

Step 3: Configure Flexmonster Pivot Component

Now it’s time to configure the client – Flexmonster Pivot Component. Let’s create a minimal configuration using JavaScript API (replace proxyUrl, catalog and cube parameters with your specific values):

```
var config = {
    dataSourceType: "microsoft analysis services",
    proxyUrl: "https://localhost:50005",
    catalog: "Adventure Works DW Standard Edition",
    cube: "Adventure Works",
```

```
    binary: true
}
flexmonster.embedPivotComponent("", "pivot-container","100%", "500", config);
```

Please note that proxyUrl now contains https://. That means that the data is protected and encrypted with your SSL certificate.

5.5. MS Analysis Services / Troubleshooting

Erorr: Your current version of Flexmonster Pivot Component is not compatible with Data Speed Accelerator. Please update the component to the minimum required version: X.XXX

It means that versions of Flexmonster Accelerator and Pivot Table Component are different and not compatible. To fix this error it is recommended to update both items to the latest available version. Please refer to the "How to update" guide for more details.

Error: Your current version of Data Speed Accelerator is not compatible with Flexmonster Pivot Component. Please update the Accelerator to the minimum required version: X.XXX

It means that versions of Flexmonster Accelerator and Pivot Table Component are different and not compatible. To fix this error it is recommended to update both items to the latest available version. Please refer to the "How to update" guide for more details.

6. Connecting to Pentaho Mondrian

There are two ways to connect to Pentaho Mondrian using Flexmonster Pivot Table:

1. via XMLA – an industry standard for data access in analytical systems
2. via Flexmonster Accelerator – special server-side utility developed by Flexmonster

If you already have configured XMLA provider it will be preferable to start with option #1. In case you do not have XMLA or you need some advanced features (increase loading speed, use credentials, etc.) – option #2 is a better choice.

Connecting to Pentaho Mondrian via XMLA

XMLA (XML for Analysis) – an industry standard for data access in analytical systems, such as OLAP and Data Mining. Please follow the steps below to configure a connection to Pentaho Mondrian via XMLA.

Step 1: Embed the component into your web page

Embedding Flexmonster Pivot is simple copy-paste procedure that takes just minutes!

1. Copy contents of component/ folder into the web project root to your server.
2. Copy the code below and paste it into your HTML page.

```
<div id="pivotContainer">The component will appear here</div>
<script src="flexmonster/flexmonster.js"></script>
<script>
    flexmonster.embedPivotComponent(
        "flexmonster/", "pivotContainer", "100%", "500",
        {   /* empty report */
            licenseKey: "XXXX-XXXX-XXXX-XXXX-XXXX"
        }, true
    );
</script>
```

Run your web page and see the empty table. The next step is to see your own data on the grid.

Step 2: Configure XMLA access to the cube

If you have XMLA already configured please skip this step. Otherwise, please refer to the article that explains how to configure Mondrian as an XMLA provider.

Step 3: Enable cross-origin resource sharing (CORS)

By default, browser prevents JavaScript from making requests across domain boundaries. CORS allows web applications to make cross-domain requests. Please find instructions for common Java servers below:

- Tomcat – Configuration Reference (CORS Filter)
- Jetty – Jetty/Feature/Cross Origin Filter
- JBOSS – CORS Filter Installation

Step 4: Configure report with your own data

Now it's time to configure Flexmonster Pivot Component on the web page. Let's create a minimal report for this (replace proxyUrl, dataSourceInfo, catalog and cube parameters with your specific values):

```
flexmonster.embedPivotComponent(
    "flexmonster/", "pivotContainer", "100%", "500",
    {
        dataSourceType: "mondrian",

        /* URL to XMLA provider */
        proxyUrl: "http://olap.flexmonster.com:8080/mondrian/xmla",

        /* Data source info */
        dataSourceInfo: "MondrianFoodMart",

        /* Catalog name */
        catalog: "FoodMart",

        /* Cube name */
        cube: "Sales",

        licenseKey: "XXXX-XXXX-XXXX-XXXX-XXXX"
    }, true
);
```

Launch the web page from browser — here you go! Pivot table is embedded into your project.

Connecting to Pentaho Mondrian via Flexmonster Accelerator

Everyone who has ever worked with multidimensional databases analysis has most probably faced the problem with finding a suitable component that not only performs all required functions but also executes them fast. Flexmonster solved this problem and created **Flexmonster Accelerator for Pentaho Mondrian** cubes – a special server-side proxy that helps you to increase data loading speed from server to customer's browser.

Working with OLAP cubes, a browser component is communicating with the server via XMLA protocol. It's no secret that the XMLA protocol is heavy and exchanges a lot of excessive information. Thus, it takes too much time and memory to load and process the data.

We replaced XMLA protocol and use direct requests from the Component to a server.

In this way, we have come up with solutions to two major problems that many of those who work with big data had faced to a different extent:

- We made big data transfer from server to browser enormously fast. Our tool allows you to transfer large multidimensional data in super easy and fast way. Reporting becomes more enjoyable and prompt for your end users.
- We greatly reduced the load on a browser memory.

Please refer to Getting started with Accelerator guide to find step-by-step instructions.

6.1. Mondrian / Getting started with Accelerator

Everyone who has ever worked with multidimensional databases analysis has most probably faced the problem with finding a suitable component that not only performs all required functions but also executes them fast. Flexmonster solved this problem and created **Flexmonster Accelerator for Pentaho Mondrian** cubes – a special server-side proxy that helps you to increase data loading speed from server to customer's browser.

Working with OLAP cubes, a browser component is communicating with the server via XMLA protocol. It's no secret that the XMLA protocol is heavy and exchanges a lot of excessive information. Thus, it takes too much time and memory to load and process the data.

We replaced XMLA protocol and use direct requests from the Component to a server.

In this way, we have come up with solutions to two major problems that many of those who work with big data had faced to a different extent:

- We made big data transfer from server to browser enormously fast. Our tool allows you to transfer large multidimensional data in super easy and fast way. Reporting becomes more enjoyable and prompt for your end users.
- We greatly reduced the load on a browser memory.

Open Live Demo

Requirements

- Flexmonster Pivot Component v2.2 SITE edition or higher

- Relational database and Mondrian schema for it
- Java JRE 1.7+

Step 1: Configure Data Speed Accelerator on the server

The package contains following files (server/ folder):

- flexmonster-proxy-mondrian.jar – server-side utility that handles connectivity between Pentaho Mondrian and Flexmonster Pivot Component
- flexmonster-proxy-mondrian-4.jar – same as flexmonster-proxy-mondrian.jar, but for newer version of Pentaho Mondrian 4.4
- flexmonster.config – file that contains configuration parameters for the utility (connection string, port, etc.)

Also, there are additional sample files:

- FoodMart.xml – sample Mondrian schema for FoodMart database
- FoodMart-4.xml – same as FoodMart.xml, but for newer version of Pentaho Mondrian 4.4
- mysql-connector-java-*.jar – Java connector for MySQL database

First of all, let's review flexmonster.config file. It contains following parameters:

- CONNECTION_STRING – connection string for Pentaho Mondrian. Required.
  Example:
  Jdbc=jdbc:mysql://localhost:3306/foodmart?user=root&password=password;JdbcDrivers=com.mysql.jdbc.Driver;
- JDBC_DRIVER_JAR – path to the Java connector (.jar file) for database. Required.
  Example: *./mysql-connector-java-5.1.37.jar*
- JDBC_DRIVER_CLASS – class name of the Java connector. Required.
  Example: *com.mysql.jdbc.Driver*
- CATALOGS_PATH – a path to the folder that contains Mondrian schemas. Optional. Default is ./
- PORT – port number for the proxy service endpoint. Optional. Default is 50006
- CACHE_MEMORY_LIMIT – size of maximum RAM memory available for cache (in MB). Optional. Default is 0 (unlimited).
- CACHE_ENABLED – indicates whether the cache is enabled. Optional. Default is true. Available since version 2.211.

After configuring all the necessary options, Data Speed Accelerator is ready to be launched. Just execute the following commands in terminal:

```
# navigate to folder that contains Accelerator
cd {path_to_package}/server
# start Accelerator
java -jar flexmonster-proxy-mondrian.jar
```

If your schema is built for Mondrian 4, just use flexmonster-proxy-mondrian-4.jar instead.

Step 2: Open a port for Data Speed Accelerator in the firewall

If it's planned to allow connection to Data Speed Accelerator from outside of the server, you should open an appropriate port in the firewall. Default port number is 50006, but it may vary depending on PORT parameter in flexmonster.config file.

Step 3: Configure Flexmonster Pivot Component

Now it's time to configure the client – Flexmonster Pivot Component. Let's create a minimal configuration using JavaScript API (replace proxyUrl, catalog and cube parameters with your specific values):

```
var config = {
  dataSourceType: "mondrian",
  // URL to the Data Speed Accelerator
  proxyUrl: "http://localhost:50006",
  // Catalog name / Mondrian schema
  catalog: "FoodMart",
  // Cube name
  cube: "Sales",
  // Flag to use Data Speed Accelerator instead of XMLA protocol
  binary: true
};
flexmonster.embedPivotComponent("", "pivot-container", "100%", "500", config);
```

Also, it's possible to create XML report with the same configuration:

```
<config>
  <dataSource type="mondrian">
    <!-- URL to the Data Speed Accelerator -->
    <proxyUrl>http://localhost:50006</proxyUrl>
    <!-- Catalog name / Mondrian schema -->
    <catalog>FoodMart</catalog>
    <!-- Cube name -->
    <cube>Sales</cube>
    <!-- Flag to use Data Speed Accelerator instead of XMLA protocol -->
    <binary>true</binary>
  </dataSource>
</config>
```

Cache control

Usually, the cache is a great help, but the cache becomes out of date if the underlying database is changing. Mondrian cannot deduce when the database is being modified, so we introduce a method to force clearing the cache.

It works by triggering the "ClearCache" as follows:

http://localhost:50006/FlexmonsterProxy/ClearCache

Also, it is possible to disable cache by the following parameter in flexmonster.config (available since version 2.211):

```
CACHE_ENABLED=false
```

Mondrian configuration

Mondrian has a properties file to allow you to configure how it executes. It is possible to use any of these properties with Accelerator.

You just need to create mondrian.properties file in the same location as flexmonster-proxy-mondrian.jar.

For example:

```
mondrian.rolap.aggregates.ChooseByVolume = false
mondrian.rolap.aggregates.generateSql = false
```

Please refer to the full list of Mondrian properties to find out more.

6.2. Configuring Mondrian roles

Overview

Sometimes it's necessary to limit access to some parts of the data. To solve this you can define an access-control profile, called a Role, as part of the Mondrian schema and set this role when establishing a connection with Flexmonster Pivot Component.

Requirements

- Flexmonster Pivot Component v2.210 SITE edition or higher
- Relational database and Mondrian schema for it
- Java JRE 1.7+

Step 1: Configure roles in Mondrian schema file

First of all, you need to configure roles in the Mondrian schema file. Please refer to the Mondrian documentation for more details.

Step 2: Launch Flexmonster Accelerator

Start (or restart) Flexmonster Accelerator for Mondrian. Please refer to the Accelerator "Getting Started" guide for more details.

Step 3: Configure Flexmonster Pivot Component

Now, let's specify Mondrian roles in the configuration. Here is a minimal sample created with JavaScript API (replaceproxyUrl, catalog, cube and roles parameters with your specific values):

```
var config = {
  dataSourceType: "mondrian",
  proxyUrl: "http://localhost:50006",
  catalog: "FoodMart",
```

```
  cube: "Sales",
  binary: true,
  // Mondrian roles
  roles: "California manager"
}
flexmonster.embedPivotComponent("", "pivot-container","100%", "500", config);
```

Also, it's possible to create XML report with the same configuration:

```
<config>
  <dataSource type="mondrian">
    <proxyUrl>http://localhost:50006</proxyUrl>
    <catalog>FoodMart</catalog>
    <cube>Sales</cube>
    <binary>true</binary>
    <!-- Mondrian roles -->
    <roles>California manager</roles>
  </dataSource>
</config>
```

6.3. Mondrian / ?onfiguring username/password protection

Overview

Flexmonster Pivot Component supports credentials to access the data source. It is commonly used for the following purposes:

- provide credentials for data source connectivity
- provide access for users with different roles and access levels

Here are the step-by-step instructions to configure secure connection with username/password protection.

Requirements

- Flexmonster Pivot Component v2.205 SITE edition or higher
- Relational database and Mondrian schema for it
- Java JRE 1.7+

Step 1: Create a default user for Accelerator

We recommend creating a default user for Accelerator with minimum privileges. It is used to start the Accelerator, check connection to the data source and connect anonymous users. Refer to the documentation of your database to create a new user (i.e. MySQL, Oracle).

Step 2: Configure Accelerator

Open flexmonster.config and specify user and password in the CONNECTION_STRING parameters. After

editing, flexmonster.config should look like the following:

```
CONNECTION_STRING=Jdbc=jdbc:mysql://localhost:3306/foodmart?user=flexmonster&passwor
d=password;JdbcDrivers=com.mysql.jdbc.Driver;
```

Accelerator is ready to be launched. Just execute the following command in terminal:

```
java -jar flexmonster-proxy-mondrian.jar
```

You can check the Accelerator is up and running by navigating to its URL in the browser (i.e. http://localhost:50006).

Step 3: Configure Flexmonster Pivot Component

Note!
All data that is sent by HTTP is not encrypted and username/password can be intercepted. We recommend using credentials only with HTTPS. HTTPS encrypts all data that is sent from the client to Accelerator.
Read more at How to configure secure HTTPS connection*.*

Now, let's specify credentials for database user in the configuration. Here is a minimal sample created with JavaScript API (replace proxyUrl, catalog, cube and credentials parameters with your specific values):

```
var config = {
  dataSourceType: "mondrian",
  proxyUrl: "http://localhost:50006",
  catalog: "FoodMart",
  cube: "Sales",
  binary: true,
  // Credentials of database user
  username: "webuser",
  password: "1234"
}
flexmonster.embedPivotComponent("", "pivot-container","100%", "500", config);
```

Also, it's possible to create XML report with the same configuration:

```
<config>
  <dataSource type="mondrian">
    <proxyUrl>http://localhost:50006</proxyUrl>
    <catalog>FoodMart</catalog>
```

```
    <cube>Sales</cube>
    <binary>true</binary>
    <!-- Credentials of database user -->
    <credentials>
      <username>webuser</username>
      <password>1234</password>
    </credentials>
  </dataSource>
</config>
```

6.4. Mondrian / ?onfiguring secure HTTPS connection

Overview

All data that is sent by HTTP is not encrypted and can be intercepted. That's why we have added an option to enable HTTPS for Accelerator. HTTPS encrypts all data that is sent from the client to Accelerator and vice versa. Please follow the steps below to configure secure HTTPS connection for your setup.

Requirements

- Flexmonster Pivot Component v2.205 SITE edition or higher
- Relational database and Mondrian schema for it
- Java JRE 1.7+
- **Valid and trusted SSL certificate**

Step 1: Import certificate to Java KeyStore

A Java KeyStore (JKS) is a repository of security certificates. JDKs provide a keytool utility to manipulate the keystore.

If you already have the private key in PKCS 12, just navigate to JRE/bin/ folder and execute the following:

```
keytool -importkeyst
ore -destkeystore keystore.jks -srckeystore <private_key> -srcstoretype PKCS12
```

Where:

- <private_key> – path to .p12 file with the private key (i.e. *flexmonster.p12*)

It will ask to enter a new password for keystore and a password for the private key. In a result, it will generate a keystore.jks file with imported private key inside. Copy this file to some location, it will be necessary for the next step.

Step 2: Enable HTTPS in Accelerator

After you have the certificate imported let's enable the HTTPS in the Accelerator's config. Open flexmonster.config file and modify/add the necessary parameters as follows:

```
HTTPS=true
KEY_STORE_PATH=<keystore_path>
KEY_STORE_PASSWORD=<keystore_password>
KEY_MANAGER_PASSWORD=<private_key_password>
```

Where:

- <keystore_path> – path to JKS file (i.e. *keystore.jks*)
- <keystore_password> – password for the keystore
- <private_key_password> – password for the imported key

Available values for HTTPS parameter is true or false. By default HTTPS is disabled (false).

Accelerator is ready to be launched. Just execute the following command in terminal:

```
java -jar flexmonster-proxy-mondrian.jar
```

You can check the Accelerator is up and running by navigating to its URL in the browser (i.e. https://localhost:50006).

Step 3: Configure Flexmonster Pivot Component

Now it's time to configure the client – Flexmonster Pivot Component. Let's create a minimal configuration using JavaScript API (replace proxyUrl, catalog and cube parameters with your specific values):

```
var config = {
  dataSourceType: "mondrian",
  proxyUrl: "https://localhost:50006",
  catalog: "FoodMart",
  cube: "Sales",
  binary: true
};
flexmonster.embedPivotComponent("", "pivot-container", "100%", "500", config);
```

Please note that proxyUrl now contains https://. That means that the data is protected and encrypted with your SSL certificate.

6.5. Mondrian / Troubleshooting

Erorr: Your current version of Flexmonster Pivot Component is not compatible with Data Speed Accelerator. Please update the component to the minimum required

version: X.XXX

It means that versions of Flexmonster Accelerator and Pivot Table Component are different and not compatible. To fix this error it is recommended to update both items to the latest available version. Please refer to the "How to update" guide for more details.

Error: Your current version of Data Speed Accelerator is not compatible with Flexmonster Pivot Component. Please update the Accelerator to the minimum required version: X.XXX

It means that versions of Flexmonster Accelerator and Pivot Table Component are different and not compatible. To fix this error it is recommended to update both items to the latest available version. Please refer to the "How to update" guide for more details.

Error: java.lang.OutOfMemoryError: GC overhead limit exceeded

Please try to start the Accelerator with the following: java -Xmx1024m -jar flexmonster-proxy-mondrian.jar where -Xmx<size> is the maximum Java heap size.

Error: java.net.UnknownHostException: <hostname>: <hostname>: Name or service not known

Some enterprise environments do not allocate DNS names to their devices – that results in an UnknownHostException, often after a lengthy DNS lookup attempt. Here are few possible actions:

- the first suggestion is to ignore the message. Given that error the application is working fine and the message is only seen in the logs. Please try to open your browser and navigate to the following URL: http://127.0.0.1:50006/ or http://localhost:50006/. There should be a message "Flexmonster Data Speed Accelerator for Pentaho Mondrian".
- or you can try to add the following record to the hosts file: 127.0.0.1 <hostname>

7. Connecting to icCube

Please follow the steps below to configure a connection to icCube via XMLA service.

Step 1: Embed the component into your web page

Embedding Flexmonster Pivot is simple copy-paste procedure that takes just minutes!

1. Copy contents of component/ folder into the web project root to your server.
2. Copy the code below and paste it into your HTML page.

```
<div id="pivotContainer">The component will appear here</div>
<script src="flexmonster/flexmonster.js"></script>
<script>
    flexmonster.embedPivotComponent(
        "flexmonster/", "pivotContainer", "100%", "500",
        {   /* empty report */
            licenseKey: "XXXX-XXXX-XXXX-XXXX-XXXX"
        }, true
    );
</script>
```

Run your web page and see the empty table. The next step is to see your own data on the grid.

Step 2: Configure XMLA access to the cube

Assuming the server with the default configuration is running on the local machine, the XMLA service is available at the following address:

```
XMLA/HTTP : http://localhost:8282/icCube/xmla
```

Please refer to the icCube documentation for more details.

Step 3: Enable cross-origin resource sharing (CORS)

By default, browser prevents JavaScript from making requests across domain boundaries. CORS allows web applications to make cross-domain requests. Please open icCube configuration file (located at $install/bin/icCube.xml) and edit the following:

1. Cross Origin filter:

```
<icCubeConfiguration>
    ...
    <xmlaComponentConfiguration>
        ...
        <filter>Cross Origin</filter>
        ...
    </xmlaComponentConfiguration>
    ...
</icCubeConfiguration>
```

2. Cross Origin filter configuration:

```
<icCubeConfiguration>
    ...
    <filterConfiguration>
        ...
        <filter>
            <filter-name>Cross Origin</filter-name>
            <filter-class>crazydev.iccube.server.crossorigin.IcCubeCrossOrigin
ServletFilter</filter-class>
            <init-param>
                <param-name>allowedOrigins</param-name>
                <param-value>*</param-value>
            </init-param>
            <init-param>
                <param-name>allowedMethods</param-name>
                <param-value>GET,POST,HEAD,OPTIONS</param-value>
            </init-param>
            <init-param>
                <param-name>allowedHeaders</param-name>
                <param-value>X-Requested-With,Content-Type,Accept,Origin,X-
DataSource-Auth</param-value>
            </init-param>
```

```
            </filter>
            ...
        </filterConfiguration>
        ...
    </icCubeConfiguration>
```

Step 4: Configure report with your own data

Now it's time to configure Flexmonster Pivot Component on the web page. Let's create a minimal report for this (replace proxyUrl, catalog and cube parameters with your specific values):

```
flexmonster.embedPivotComponent(
    "flexmonster/", "pivotContainer", "100%", "500",
    {
        dataSourceType: "iccube",

        /* URL to XMLA service */
        proxyUrl: "http://olap.flexmonster.com:8282/icCube/xmla",

        /* Catalog name */
        catalog: "Sales",

        /* Cube name */
        cube: "Sales",

        licenseKey: "XXXX-XXXX-XXXX-XXXX-XXXX"
    }, true
);
```

Launch the web page from browser — here you go! Pivot table is embedded into your project.

Please visit our live icCube demo to see how this sample works.

8. Configuring report

What is a report?

A report is a definition of what data should be shown in the component and how it will be visualized.

Reports can be either in JSON or in XML format.

You are already familiar with the report object thanks to Your first pivot table article where the report object was used to visualize pivot table based on JSON data. The properties data, dataSourceType, rows, columns and measures of the report object were defined there. Also, report objects were used to connect to JSON, CSV, SQL databases, Microsoft Analysis Services, Pentaho Mondrian and icCube in the previous articles of the documentation.

Now it is time to understand what else except the data source configuration can be defined in reports.

Report object has many properties. Actually, all the possible aspects of pivot tables and pivot charts configuration can be set via report object. The component supports saving reports and loading of previously saved ones.

Report parts

Report properties can be divided into the following logical parts:

- data source*
- default slice
- options
- number formatting
- conditional formatting

*Only the part that describes data source is the required one in reports. Others are optional.

The simplest report

The simplest report consists only of the data source definition.

Let's see how the same simplest report that defines connection to the static CSV file looks like in JSON format and in XML format (please note that report.xml file should be in UTF-8 encoding):

JSON

```
{
  dataSourceType: "csv",
  filename: "data.csv"
}
```

XML

```
<config>
  <dataSource type="csv">
    <filename>data.csv</filename>
  </dataSource>
</config>
```

This is how the above report is set in embedPivotComponent():

JSON

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

<script>
    flexmonster.embedPivotComponent(
        "flexmonster/", "pivotContainer", "100%", "500", {
            dataSourceType: "csv",
            filename: "data.csv",
            licenseKey: "XXXX-XXXX-XXXX-XXXX-XXXX"
        }, true
    );
</script>
```

XML

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

<script>
    flexmonster.embedPivotComponent(
        "flexmonster/", "pivotContainer", "100%", "500", {
            configUrl: "report.xml",
            licenseKey: "XXXX-XXXX-XXXX-XXXX-XXXX"
        }, true
    );
</script>
```

8.1. Number formatting

The way how numeric values are formatted in the component can be defined in a report.

A default format is for all measures. The specific number formats can be defined for some measures in addition to the default format. More details are below in the following sections:

- Number format properties
- Default number format
- Number format for a specific measure
- Change number formatting using Toolbar

If the component is connected to OLAP cube and you already have formatted numbers there (Microsoft Analysis Services or Mondrian), you can display these formatted values without applying number formatting in the component. See more about this option below:

- Number formatting from OLAP cube

Number format properties

With number formatting you can define thousands and decimal separators, the number of decimals to show in the fractional part of a number, how to display null and infinity values, you can format number as currencies specifying the currency symbol and its position – right or left.

Here is a list of all available properties:

- 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".

Default number format

Flexmonster component has a built-in default number format which is applied to all measures by default. It is composed of the default values of the number format properties. The default format can be overridden in a report.

If you want to override the default number format for a report, please define a number format with an empty string name property in a report, as follows:

JSON

```
{
  dataSourceType: "csv",
  filename: "data.csv",
  rows: [
    { uniqueName: "?ountry" }
  ],
  columns: [
    { uniqueName: "[Measures]" }
  ],
  measures: [
    { uniqueName: "Price", aggregation: "sum", active: true },
    { uniqueName: "Quantity", aggregation: "sum", active: true }
  ],
  formats: [
    {
```

```
      name: "",
      thousandsSeparator: " ",
      decimalSeparator: ".",
      decimalPlaces: -1,
      maxDecimalPlaces: -1,
      maxSymbols: 20,
      currencySymbol: "",
      currencySymbolAlign: "left",
      nullValue: "",
      infinityValue: "Infinity",
      divideByZeroValue: "Infinity",
      textAlign: "right"
    }
  ]
}
```

XML

```
<config>
  <dataSource type="csv">
    <filename>data.csv</filename>
  </dataSource>
  <defaultSlice>
    <axes>
      <axis name="rows">
        <hierarchy sort="asc">
          <dimensionName>Country</dimensionName>
          <hierarchyName>Country</hierarchyName>
        </hierarchy>
      </axis>
      <axis name="columns">
        <hierarchy>
          <dimensionName>[Measures]</dimensionName>
          <hierarchyName>[Measures]</hierarchyName>
        </hierarchy>
      </axis>
    </axes>
    <measures>
      <measure aggregation="sum" active="true">Price</measure>
      <measure aggregation="sum" active="true">Quantity</measure>
    </measures>
  </defaultSlice>
  <format name="">
    <param name="decimalPlaces">-1</param>
    <param name="maxDecimalPlaces">-1</param>
    <param name="maxSymbols">20</param>
    <param name="currencySymbolAlign">left</param>
    <param name="thousandsSeparator"><![CDATA[ ]]></param>
    <param name="decimalSeparator"><![CDATA[.]]></param>
    <param name="currencySymbol"><![CDATA[]]></param>
    <param name="nullValue"><![CDATA[]]></param>
    <param name="infinityValue"><![CDATA[Infinity]]></param>
```

```
    <param name="divideByZeroValue"><![CDATA[Infinity]]></param>
    <param name="textAlign">right</param>
  </format>
</config>
```

Number format for a specific measure

A number format can be applied to a specific measure or measures. Each measure has only one format but a format can be applied to more than one measure.

For example, if you are visualizing financial data, you may want to apply currency formatting to some of the measures in addition to the default format. To apply the format to the specific measure, two things should be done:

- a format should be named,
- the format name should be defined for the measure(-s) in a default slice.

Please see an example below:

JSON

```
{
  dataSourceType: "csv",
  filename: "data.csv",
  rows: [
    { uniqueName: "?ountry" }
  ],
  columns: [
    { uniqueName: "[Measures]" }
  ],
  measures: [
    { uniqueName: "Price", aggregation: "sum", active: true, format: "currency" },
    { uniqueName: "
Discount", aggregation: "sum", active: false, format: "currency" },
    { uniqueName: "Quantity", aggregation: "sum", active: true }
  ],
  formats: [
    {
      name: "",
      thousandsSeparator: " ",
      decimalSeparator: ".",
      decimalPlaces: -1,
      maxDecimalPlaces: -1,
      maxSymbols: 20,
      currencySymbol: "",
      currencySymbolAlign: "left",
      nullValue: "",
```

```
      infinityValue: "Infinity",
      divideByZeroValue: "Infinity",
      textAlign: "right"
    },
    {
      name: "currency",
      thousandsSeparator: " ",
      decimalSeparator: ".",
      decimalPlaces: 2,
      maxDecimalPlaces: -1,
      maxSymbols: 20,
      currencySymbol: "$",
      currencySymbolAlign: "left",
      nullValue: "",
      infinityValue: "Infinity",
      divideByZeroValue: "Infinity",
      textAlign: "right"
    }
  ]
}
```

XML

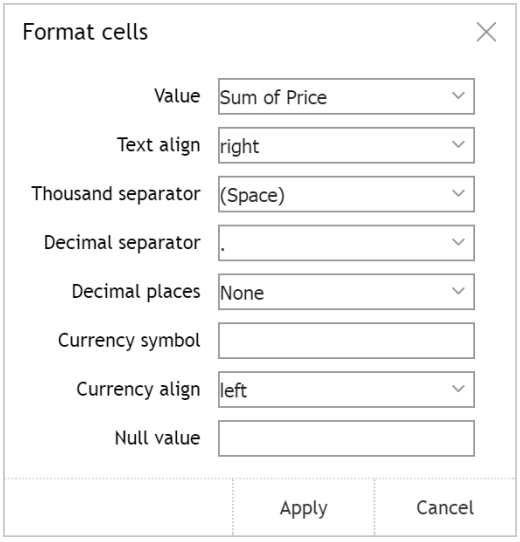
```
<config>
  <dataSource type="csv">
    <filename>data.csv</filename>
  </dataSource>
  <defaultSlice>
    <axes>
      <axis name="rows">
        <hierarchy sort="asc">
          <dimensionName>Country</dimensionName>
          <hierarchyName>Country</hierarchyName>
        </hierarchy>
      </axis>
      <axis name="columns">
        <hierarchy>
          <dimensionName>[Measures]</dimensionName>
          <hierarchyName>[Measures]</hierarchyName>
        </hierarchy>
      </axis>
    </axes>
    <measures>
      <measure aggregation="sum" active="true" format="currency">Price</measure>
      <measure aggregation="sum" active="false" format="currency">Discount</measure>
      <measure aggregation="sum" active="true">Quantity</measure>
    </measures>
  </defaultSlice>
  <format name="">
    <param name="decimalPlaces">-1</param>
    <param name="maxDecimalPlaces">-1</param>
    <param name="maxSymbols">20</param>
```

```
    <param name="currencySymbolAlign">left</param>
    <param name="thousandsSeparator"><![CDATA[ ]]></param>
    <param name="decimalSeparator"><![CDATA[.]]></param>
    <param name="currencySymbol"><![CDATA[]]></param>
    <param name="nullValue"><![CDATA[]]></param>
    <param name="infinityValue"><![CDATA[Infinity]]></param>
    <param name="divideByZeroValue"><![CDATA[Infinity]]></param>
    <param name="textAlign">right</param>
  </format>
  <format name="currency">
    <param name="decimalPlaces">2</param>
    <param name="maxDecimalPlaces">-1</param>
    <param name="maxSymbols">20</param>
    <param name="currencySymbolAlign">left</param>
    <param name="thousandsSeparator"><![CDATA[ ]]></param>
    <param name="decimalSeparator"><![CDATA[.]]></param>
    <param name="currencySymbol"><![CDATA[$]]></param>
    <param name="nullValue"><![CDATA[]]></param>
    <param name="infinityValue"><![CDATA[Infinity]]></param>
    <param name="divideByZeroValue"><![CDATA[Infinity]]></param>
    <param name="textAlign">right</param>
  </format>
</config>
```

Please note that a format can be defined for the measure(-s) even if they are not active (active property is false) in a default slice.

Change number formatting using Toolbar

Please use Format > Format cells in Toolbar to change/define number formatting for measures in run time.



The number format will be applied to the measures and will be saved within the report.

API calls setFormat() and getFormat() are used in Toolbar to manipulate number formatting in run time.

Number formatting from OLAP cube

If you already have formats defined for measures in OLAP cube and you want to use the formatted values from the cube, please set useOLAPFormatting report property to true to enable this (it is turned off by default), as follows:

JSON

```
{
  dataSourceType: "microsoft analysis services",
  proxyUrl: "http://olap.flexmonster.com/olap/msmdpump.dll",
  cube: "Adventure Works",
  catalog: "Adventure Works DW Standard Edition",
  rows: [{uniqueName: "[Product].[Category]"}, {uniqueName: "[Reseller].[Business Ty
pe]"}],
  columns: [{uniqueName: "[Measures]"}],
  measures: [{uniqueName: "[Measures].[Reseller Order Count]"}],
  useOLAPFormatting: true
}
```

XML

```
<config>
  <dataSource type="microsoft analysis services">
    <catalog>Adventure Works DW Standard Edition</catalog>
    <cube>Adventure Works</cube>
    <proxyUrl>http://olap.flexmonster.com/olap/msmdpump.dll</proxyUrl>
  </dataSource>
  <defaultSlice>
    <axes>
      <axis name="rows">
        <hierarchy sort="asc">
          <dimensionName>[Product]</dimensionName>
          <hierarchyName>[Product].[Category]</hierarchyName>
        </hierarchy>
        <hierarchy sort="asc">
          <dimensionName>[Reseller]</dimensionName>
          <hierarchyName>[Reseller].[Business Type]</hierarchyName>
        </hierarchy>
      </axis>
      <axis name="columns">
        <hierarchy>
          <dimensionName>[Measures]</dimensionName>
          <hierarchyName>[Measures]</hierarchyName>
        </hierarchy>
      </axis>
    </axes>
    <measures>
      <measure aggregation="none" active="true">[Measures].[Reseller Order Count]</m
```

```
easure>
    </measures>
  </defaultSlice>
  <params>
    <param name="useOlapFormatting">true</param>
  </params>
</config>
```

Please note that useOLAPFormatting is supported for Microsoft Analysis Services via both Accelerator and XMLA, and for Mondrian via Accelerator. It is not available for Mondrian via XMLA and for icCube.

8.2. Set report to the component

There are several ways to set report in JSON or XML format to the component:

- Using the 5th parameter in embedPivotComponent() API call

JSON

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

<script>
    flexmonster.embedPivotComponent(
        "flexmonster/", "pivotContainer", "100%", "500", {
            dataSourceType: "json",
            data: jsonData,
            rows: [
                { uniqueName: "Color" },
                { uniqueName: "[Measures]" }
            ],
            columns: [
                { uniqueName: "country" }
            ],
            measures: [
                { uniqueName: "Price", aggregation: "sum" }
            ],
            licenseKey: "XXXX-XXXX-XXXX-XXXX-XXXX"
        }, true
    );
</script>
```

XML

Note: using configUrl property in the 5th parameter in embedPivotComponent()

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

<script>
    flexmonster.embedPivotComponent(
        "flexmonster/", "pivotContainer", "100%", "500", {
            configUrl : "report.xml",
            licenseKey: "XXXX-XXXX-XXXX-XXXX-XXXX"
        }, true
    );
</script>
```

- Using setReport() API call

JSON

```
<button onclick="setReport()">Set report</button>
<div id="pivotContainer">The component will appear here</div>
<script src="flexmonster/flexmonster.js"></script>

<script>
    flexmonster.embedPivotComponent(
        "flexmonster/", "pivotContainer", "100%", "500", {
            licenseKey: "XXXX-XXXX-XXXX-XXXX-XXXX"
        }, true
    );
    function setReport() {
        var report = flexmonster.getReport();
        // parse, change or save for later use
        flexmonster.setReport(report);
    }
</script>
```

XML

```
<button onclick="setReportXML()">Set report XML</button>
<div id="pivotContainer">The component will appear here</div>
<script src="flexmonster/flexmonster.js"></script>
```

```
<script>
    flexmonster.embedPivotComponent(
        "flexmonster/", "pivotContainer", "100%", "500", {
            licenseKey: "XXXX-XXXX-XXXX-XXXX-XXXX"
        }, true
    );
    function setReportXML() {
        var reportXML = flexmonster.getReport("xmlstring");
        // parse, change or save for later use
        flexmonster.setReport(reportXML);
    }
</script>
```

- Using open() API call to select XML file from the local file system

XML

```
<button onclick="openReport()">Open report XML</button>
<div id="pivotContainer">The component will appear here</div>
<script src="flexmonster/flexmonster.js"></script>

<script>
    flexmonster.embedPivotComponent(
        "flexmonster/", "pivotContainer", "100%", "500", {
            licenseKey: "XXXX-XXXX-XXXX-XXXX-XXXX"
        }, true
    );
    function openReport() {
        flexmonster.open();
    }
</script>
```

- Using load() API call to load XML file from URL

XML

```
<button onclick="loadReport()">Load report XML</button>
<div id="pivotContainer">The component will appear here</div>
<script src="flexmonster/flexmonster.js"></script>
```

```
<script>
    flexmonster.embedPivotComponent(
        "flexmonster/", "pivotContainer", "100%", "500", {
            licenseKey: "XXXX-XXXX-XXXX-XXXX-XXXX"
        }, true
    );
    function loadReport() {
        flexmonster.load("report.xml");
    }
</script>
```

8.3. Get report from the component

There are several ways to get report in JSON or XML format from the component:

- You can get report from the component using getReport() API call

JSON

```
<button onclick="getReport()">Get report</button>
<div id="pivotContainer">The component will appear here</div>
<script src="flexmonster/flexmonster.js"></script>

<script>
    flexmonster.embedPivotComponent(
        "flexmonster/", "pivotContainer", "100%", "500", {
            licenseKey: "XXXX-XXXX-XXXX-XXXX-XXXX"
        }, true
    );
    function getReport() {
        var report = flexmonster.getReport();
        // parse, change or save for later use
        flexmonster.setReport(report);
    }
</script>
```

XML

```
<button onclick="getReportXML()">Get report XML</button>
<div id="pivotContainer">The component will appear here</div>
```

```
<script src="flexmonster/flexmonster.js"></script>

<script>
    flexmonster.embedPivotComponent(
        "flexmonster/", "pivotContainer", "100%", "500", {
            licenseKey: "XXXX-XXXX-XXXX-XXXX-XXXX"
        }, true
    );
    function getReportXML() {
        var reportXML = flexmonster.getReport("xmlstring");
        // parse, change or save for later use
        flexmonster.setReport(reportXML);
    }
</script>
```

- Using save() API call to save XML report to the server or to the local file system

XML

```
<button onclick="saveReport()">Save report XML</button>
<div id="pivotContainer">The component will appear here</div>
<script src="flexmonster/flexmonster.js"></script>

<script>
    flexmonster.embedPivotComponent(
        "flexmonster/", "pivotContainer", "100%", "500", {
            licenseKey: "XXXX-XXXX-XXXX-XXXX-XXXX"
        }, true
    );
    function saveReport() {
        flexmonster.save("report.xml", "file");
    }
</script>
```

The easiest way to create report XML file is to open All-in-One demo that is already connected to the sample CSV data and click Save button on the Toolbar.

Here is a sample of such a report file:

XML

```
<config>
  <dataSource type="csv">
    <filename>/flexmonster/data/data.csv</filename>
  </dataSource>
  <defaultSlice>
    <axes>
      <axis name="rows">
        <hierarchy sort="asc">
          <dimensionName>Country</dimensionName>
          <hierarchyName>Country</hierarchyName>
        </hierarchy>
        <hierarchy>
          <dimensionName>[Measures]</dimensionName>
          <hierarchyName>[Measures]</hierarchyName>
        </hierarchy>
      </axis>
      <axis name="columns">
        <hierarchy sort="asc">
          <dimensionName>Color</dimensionName>
          <hierarchyName>Color</hierarchyName>
        </hierarchy>
      </axis>
      <axis name="pages"/>
    </axes>
    <measures>
      <measure aggregation="sum" active="true" format="22qgtu8h">Price</measure>
      <measure aggregation="sum" active="true">Discount</measure>
    </measures>
  </defaultSlice>
  <params>
    <param name="configuratorActive">true</param>
    <param name="configuratorButton">true</param>
    <param name="configuratorMatchHeight">false</param>
    <param name="viewType">grid</param>
    <param name="zoom">1</param>
    <param name="gridTitle"></param>
    <param name="chartTitle"></param>
    <param name="showFilter">true</param>
    <param name="showFilterInCharts">true</param>
    <param name="showAggregations">true</param>
    <param name="showHierarchyCaptions">true</param>
    <param name="showMemberProperties">false</param>
    <param name="memberProperties"></param>
    <param name="chartOneLevel">false</param>
    <param name="chartAutoRange">false</param>
    <param name="chartReversedAxes">false</param>
    <param name="chartMultipleMeasures">false</param>
    <param name="showHeaders">true</param>
    <param name="showReportFiltersArea">true</param>
    <param name="editing">false</param>
    <param name="drillThrough">true</param>
    <param name="sorting">on</param>
    <param name="fitGridlines">false</param>
    <param name="expandAll">false</param>
```

```
    <param name="drillAll">false</param>
    <param name="showTotals">true</param>
    <param name="showFiltersExcel">false</param>
    <param name="showHierarchies">true</param>
    <param name="showGrandTotals">on</param>
    <param name="datePattern">dd/MM/yyyy</param>
    <param name="dateTimePattern">dd/MM/yyyy HH:mm:ss</param>
    <param name="showChartsWarning">true</param>
    <param name="showChartMeasures">true</param>
    <param name="showChartLegendButton">false</param>
    <param name="saveAllFormats">false</param>
    <param name="defaultHierarchySortName">asc</param>
    <param name="ignoreQuotedLineBreaks">true</param>
    <param name="classicView">false</param>
    <param name="flatView">false</param>
    <param name="useOlapFormatting">false</param>
    <param name="showDefaultSlice">true</param>
    <param name="showAllChartLabels">false</param>
    <param name="showCalculatedValuesButton">true</param>
    <param name="showOutdatedDataAlert">false</param>
    <param name="chartType" pieDataIndex="0" labelsHierarchy="" position="bottom" ac
tiveMeasure="">bar</param>
    <param name="localSettingsUrl"></param>
  </params>
  <conditions>
    <condition measure="[Measures].[Discount]"><![CDATA[if(AND(#value > 2000, #value
 < 4000), 'trueStyle')]]>
      <trueStyle><![CDATA[{"backgroundColor":"#ff9966","color":"#ffffff","fontFamily
":"Tahoma","fontSize":13}]]></trueStyle>
      <falseStyle><![CDATA[{}]]></falseStyle>
    </condition>
  </conditions>
  <format name="22qgtu8h">
    <param name="decimalPlaces">2</param>
    <param name="maxDecimalPlaces">-1</param>
    <param name="maxSymbols">20</param>
    <param name="currencySymbolAlign">left</param>
    <param name="thousandsSeparator"><![CDATA[ ]]></param>
    <param name="decimalSeparator"><![CDATA[.]]></param>
    <param name="currencySymbol"><![CDATA[]]></param>
    <param name="nullValue"><![CDATA[]]></param>
    <param name="infinityValue"><![CDATA[Infinity]]></param>
    <param name="divideByZeroValue"><![CDATA[Infinity]]></param>
    <param name="textAlign">right</param>
  </format>
</config>
```

8.4. Date and time formatting

This article explains how to define a format for date and time strings presentation inside the component.

Input date format

As an input date format, pivot table component supports ISO 8601 date (other formats may be used, but results can be unexpected). For example, "2016-03-20" (just date) or "2016-03-20T14:48:00" (date and time).

Presentation format

The format for presentation of dates and time inside the component differs from the input format. The component supports the pattern strings to format date and time data. There are two properties of a report to format date and time strings:

1. datePattern. 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.
2. dateTimePattern. 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.

In order to change the default date and time format for “date string” date fields, you need to specify datePattern in the report explicitly. The same is for “datetime” date fields format, so you need to specify dateTimePattern in the report.

Here is how to define datePattern in the report (JSON or XML):

JSON

```
{
  dataSourceType: "json",
  data: [
    {
      "date":{"type":"date string"},
      "n":{"type":"number"}
    },
    {
      "date":"2016-04-06 23:59:30",
      "n":1
    },
    {
      "date":"2016-04-06 23:59:30",
      "n":1
    },
    {
      "date":"2016-02-07 20:33",
      "n":1
    }
  ],
  rows: [{ uniqueName: "date" }],
  columns: [{ uniqueName: "[Measures]" }],
  measures: [{ uniqueName: "n" }],
  datePattern: "yyyy-MM-dd'T'HH:mm:ss"
}
```

XML

```
data.csv:
ds+date,n
2016-04-06 23:59:30,1
2016-04-06 23:59:30,1
2016-02-07 20:33,1

report.xml:
<config>
  <dataSource type="csv">
    <filename>data.csv</filename>
  </dataSource>
  <defaultSlice>
    <axes>
      <axis name="rows">
        <hierarchy sort="asc">
          <dimensionName>date</dimensionName>
          <hierarchyName>date</hierarchyName>
        </hierarchy>
      </axis>
      <axis name="columns">
        <hierarchy>
          <dimensionName>[Measures]</dimensionName>
          <hierarchyName>[Measures]</hierarchyName>
        </hierarchy>
      </axis>
    </axes>
    <measures>
      <measure aggregation="sum" active="true">n</measure>
    </measures>
  </defaultSlice>
  <params>
    <param name="datePattern">yyyy-MM-dd'T'HH:mm:ss</param>
  </params>
</config>
```

The same can be done for dateTimePattern in the report. Please note that both patterns datePattern and dateTimePattern can contain time part.

Also, user's local time zone is used to represent the date by default. If it is required to use UTC time zone, please include UTC: part at the beginning of the pattern (i.e. **UTC:**dd/MM/yyyy HH:mm:ss).

Patterns syntax

The pattern contains sequences of letters that are replaced with date and time values in the formatted string. For example, in the pattern "yyyy/MM" the characters "yyyy" are replaced with a four-digit year, followed by a "/" character, and the characters "MM" are replaced with a two-digit month.

The following list describes the valid pattern letters and their meaning:

- d – Day of the month. It is interpreted as numeric in one or two digits. For example: 2 or 18

- dd – Day of the month. It is interpreted as numeric in two digits. For example: 02 or 18
- ddd – Day of the month (HTML5 version only). It is interpreted as a three letters abbreviation. For example: Wed
- dddd – Day of the month (HTML5 version only). It is interpreted as the full name. For example: Wednesday
- M – Month. It is interpreted as numeric in one or two digits. For example: 3 or 11
- MM – Month. It is interpreted as numeric in two digits. For example: 03
- MMM – Month. It is interpreted as a three letters abbreviation. For example: Mar
- MMMM – Month. It is interpreted as the full name. For example: March
- yy – Year. It is interpreted as numeric in two digits. For example: 16
- yyyy – Year. It is interpreted as numeric in four digits. For example: 2016
- h – Hour of the day in a 12-hour format [1 – 12]. It is interpreted as numeric in one or two digits. For example: 1 or 12
- hh – Hour of the day in a 12-hour format [1 – 12]. It is interpreted as numeric in two digits. For example: 01 or 12
- H – Hour of the day in a 24-hour format [0 – 23]. It is interpreted as numeric in one or two digits. For example: 0 or 23
- HH – Hour of the day in a 24-hour format [0 – 23]. It is interpreted as numeric in two digits. For example: 00 or 23
- m – Minute of the hour [0 – 59]. It is interpreted as numeric in one or two digits. For example: 0 or 59
- mm – Minute of the hour [0 – 59]. It is interpreted as numeric in two digits. For example: 00 or 59
- s – Seconds in the minute [0 – 59]. It is interpreted as numeric in one or two digits. For example: 0 or 59
- ss – Seconds in the minute [0 – 59]. It is interpreted as numeric in two digits. For example: 00 or 59
- l – Milliseconds (HTML5 version only). It is interpreted as numeric in three digits. For example: 100
- L – Milliseconds (HTML5 version only). It is interpreted as numeric in two digits (rounded, if needed). For example: 10
- t – am/pm one letter indicator (HTML5 version only). For example: a or p
- tt – am/pm two letters indicator (HTML5 version only). For example: am or pm
- T – AM/PM one letter indicator (HTML5 version only). For example: A or P
- TT – AM/PM two letters indicator (HTML5 version only). For example: AM or PM
- UTC: – indicates that UTC time zone should be used. For example: UTC:dd/MM/yyyy HH:mm:ss

Here are several common date and time formats:

- ISO date – "yyyy-MM-dd"
- ISO time – "HH:mm:ss"
- ISO date and time – "yyyy-MM-dd'T'HH:mm:ss"
- long date – "MMMM d, yyyy"
- short time – "h:mm TT"
- long time – "h:mm:ss TT"

“date string” and “datetime” data types

In addition, we would like to explain the difference between “date string” and “datetime” data types.

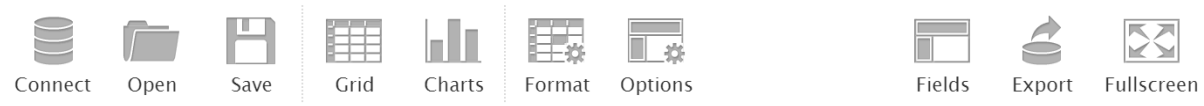
“date string” is a date field that can be used for rows, columns or pages in the pivot table when you want to present date as a string. The members of such a field are formatted using datePattern and sorted inside the component as dates.

“datetime” is a date field that can be used for values in the pivot table. Min, Max, Count and Distinct Count aggregations can be applied to it. The aggregated date/time strings in values cells in the pivot table are formatted using dateTimePattern.

9. Customizing toolbar

About Toolbar

Toolbar is an extra HTML/JS addition to Flexmonster Pivot Component. It uses standard API and provides easy access to the most used features. Toolbar is free and provided “as is”.



Toolbar is available since version 2.0 for all HTML5 and Flash/JS editions. Enabling toolbar is very easy, just set the appropriate parameter in flexmonster.embedPivotComponent() function to true.

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

Please verify that your componentFolder includes toolbar/ folder and it’s not empty.

Note: Flex edition of the component is an exception, where toolbar is provided as ActionScript 3.0 code within All-In-One demo. This article is not applicable to Flex edition of the component.

Customizing

Toolbar can be customized. Tabs and buttons can be removed from it and new ones can be added to it easily.

To get started, let’s discover the toolbar/ folder contents (you can find it in the **[package]/**demo all-in-one/html5/flexmonster/):

- toolbar/
  - img/ – icons and other graphic assets
  - flexmonster.toolbar.js – JavaScript code
  - flexmonster.toolbar.css – CSS styles

Removing tab from Toolbar

Open flexmonster.toolbar.js file. Find the Tab section (~120th line). Every tab describes with an Object, and Toolbar simply represents an Array of such Objects. For example, **Connect Tab** looks in the code like this:

```
dataProvider.push({ title: Labels.CONNECT, id: "fm-tab-connect",
  menu: [
    { title: Labels.CONNECT_LOCAL_CSV, id: "fm-tab-connect-local-
csv", handler: "connectLocalCSVHandler", mobile: false },
    { title: Labels.CONNECT_LOCAL_OCSV, id: "fm-tab-connect-local-
ocsv", handler: "connectLocalOCSVHandler", mobile: false },
    { title: isMobile ? Labels.CONNECT_REMOTE_CSV_MOBILE : Labels.CONNECT_REMOTE_CSV
, id: "fm-tab-connect-remote-csv", handler: "connectRemoteCSV" },
```

```
    { title: isMobile ? Labels.CONNECT_OLAP_MOBILE : Labels.CONNECT_OLAP, id: "fm-
tab-connect-olap", handler: "connectOLAP", flat: false }
  ]
});
```

To remove a tab you should just comment these lines.

Adding new tab to Toolbar

To add a new tab let's add the following code:

```
dataProvider.push({
  title: "New Tab",
  id: "fm-tab-newtab",
  handler: "newtabHandler"
});
```

where:

- title – [String] label of the tab
- id – [String] id used in CSS styles
- handler – [String] name of the function that handles click on this tab

Also there are another optional parameters:

- args – [Any] argument to pass to handler
- menu – [Array] dropdown menu items
- mobile – [Boolean] if false – doesn't show on mobile devices
- ios – [Boolean] if false – doesn't show on iOS devices
- android – [Boolean] if false – doesn't show on Android devices
- flash – [Boolean] if false – doesn't show with Flash/JS version
- html5 – [Boolean] if false – doesn't show with HTML5 version

Look at the existing code to understand how to use these parameters properly.

CSS styles

Best practice to apply your custom styles is to look into flexmonster.toolbar.css file and create a separate CSS file on its basis. And then include it in the HTML page with Flexmonster Pivot Component. In such a way you will override styles without changing the original files.

Localization

Please see Step 3 in Localizing component article to find out how to localize toolbar.

10. Localizing component

On our website you can see localized version of Pivot Table in Demos – Localize Pivot demo, for example to Spanish, Portuguese, Chinese, etc. The following article describes how Pivot Table can be localized.

Default language for all text elements in Pivot Table is English. The full process of component localization involves three steps:

- Step 1. Create localization XML file.
- Step 2. Load localization XML file into Pivot Table.
- Step 3. Toolbar localization (Flash/HTML5 only).

Step 1. Create your localization XML file

Localization XML file is the XML file that has the list of all text messages and labels that are used in Pivot Table and can be localized.

If you plan to use one of localizations, that is used in our demos, you are free to download respective localization file from our site. Just click the file with language below or right click and save locally.

- English
- Español
- Français
- Português
- Chinese
- ??????????

If you have downloaded one of the above files, you can jump to "Step 2. Load localization XML file into Pivot Table".

If you use different language, you can create your own localization file. We recommend using English localization file as a template. Download it, make a copy and replace all English texts with your own.

In the following example, we localize Calculated Values window in Pivot to French.

The original section of localization file looks as follows.

```
<calculatedView>
  <measureBox>Drag Values To Formula</measureBox>
  <measureName>Value name</measureName>
  <formula>Formula</formula>
</calculatedView>
```

Now we replace the respective terms with French.

```
<calculatedView>
  <measureBox>Deplacez valeur a la formule</measureBox>
  <measureName>Nom de la valeur</measureName>
  <formula>Formule</formula>
</calculatedView>
```

Once the Localization XML file is loaded (see next step), your Pivot Table component should show the following changes in Calculated Values view

NOTE: If translated terms contain not only alphabetical symbols, write them in [CDATA[ ]] blocks to avoid XML parsing problem.

Your localization can be partial. For example, if you don't need to localize Pivot Table completely, you can replace only those XML nodes that are necessary to translate. If certain label translations are not mentioned in the Localization XML file, they will be set to default English values.

Step 2. Load localization XML file into Pivot Table

To enable localization you should add path to the localization file in the Report XML file. E.g. as below

```
<params>
  ....
  <param name="localSettingsUrl"><![CDATA[local.english.xml]]></param>
</params>
```

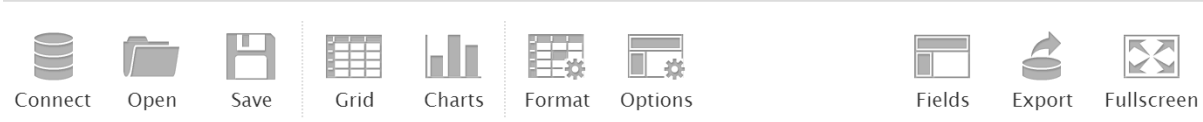
Note: We recommend writing file path in [CDATA[ ]] block to avoid XML parsing problems.

After the Report XML file is saved, you can reload Pivot Table to see localization changes effective.

Additional implementation notes:

- Every saved Report XML file can have its own localization file, referenced in localSettingsUrl node.
- If you load two reports and only first one contains localSettingsUrl node, the localization will not be changed on loading of the second report.
- Loading new report with localization changes substitutes localization that was set before.

Step 3. Toolbar localization (Flash/HTML5 only).



Also, it's possible to localize a toolbar, which easily integrates with version 2.0 and higher. Follow the steps below:

1. Create a JavaScript Object with translations. Here are the default labels:

```
var Labels = {
  // Menu
  CONNECT: "Connect",
  CONNECT_LOCAL_CSV: "To local CSV",
  CONNECT_LOCAL_OCSV: "To local OCSV",
  CONNECT_REMOTE_CSV: "To remote CSV",
  CONNECT_REMOTE_CSV_MOBILE: "CSV",
  CONNECT_OLAP: "To OLAP (XMLA)",
  CONNECT_OLAP_MOBILE: "OLAP",
  OPEN: "Open",
  LOCAL_REPORT: "Local report",
  REMOTE_REPORT: "Remote report",
  REMOTE_REPORT_MOBILE: "Report",
  SAVE: "Save",
  GRID: "Grid",
  CHARTS: "Charts",
  CHARTS_BAR: "Bar",
  CHARTS_LINE: "Line",
  CHARTS_SCATTER: "Scatter",
  CHARTS_PIE: "Pie",
  CHARTS_BAR_STACK: "Bar stack",
  CHARTS_BAR_LINE: "Bar line",
  CHARTS_MULTIPLE: "Multiple values",
  FORMAT: "Format",
  FORMAT_CELLS: "Format cells",
  FORMAT_CELLS_MOBILE: "Format",
  CONDITIONAL_FORMATTING: "Conditional formatting",
  CONDITIONAL_FORMATTING_MOBILE: "Conditional",
  OPTIONS: "Options",
  STYLES: "Styles",
  STYLES_BLUEBERRY: "Blueberry",
  STYLES_ASPHALT: "Asphalt",
  STYLES_NAVY: "Navy",
  STYLES_EMERALD: "Emerald",
  STYLES_COCOA: "Cocoa",
  STYLES_RASPBERRY: "Raspberry",
  FULLSCREEN: "Fullscreen",
  ZOOM: "Zoom",
  EXPORT: "Export",
  EXPORT_PRINT: "Print",
  EXPORT_HTML: "To HTML",
  EXPORT_CSV: "To CSV",
  EXPORT_EXCEL: "To Excel",
  EXPORT_IMAGE: "To Image",
  EXPORT_PDF: "To PDF",
  FIELDS: "Fields",
```

```
// General
OK: "OK",
APPLY: "Apply",
CANCEL: "Cancel",
VALUE: "Value",
// Connect
OPEN_REMOTE_CSV: "Open remote CSV",
OLAP_CONNECTION_TOOL: "OLAP connection tool",
SELECT_DATA_SOURCE: "Select data source",
SELECT_CATALOG: "Select catalog",
SELECT_CUBE: "Select cube",
PROXY_URL: "Proxy URL",
DATA_SOURCE_INFO: "Data Source Info",
CATALOG: "Catalog",
CUBE: "Cube",
CREDENTIALS: "credentials",
USERNAME: "Username",
PASSWORD: "Password",
// Report
OPEN_REMOTE_REPORT: "Open remote report",
// Format
TEXT_ALIGN: "Text align",
ALIGN_LEFT: "left",
ALIGN_RIGHT: "right",
NONE: "None",
SPACE: "(Space)",
THOUSAND_SEPARATOR: "Thousand separator",
DECIMAL_SEPARATOR: "Decimal separator",
DECIMAL_PLACES: "Decimal places",
CURRENCY_SYMBOL: "Currency symbol",
CURRENCY_ALIGN: "Currency align",
NULL_VALUE: "Null value",
// Conditional
ADD_CONDITION: "+ Add condition",
LESS_THAN: "Less than",
LESS_THAN_OR_EQUAL: "Less than or equal to",
GREATER_THAN: "Greater than",
GREATER_THAN_OR_EQUAL: "Greater than or equal to",
EQUAL_TO: "Equal to",
NOT_EQUAL_TO: "Not equal to",
BETWEEN: "Between",
ALL_VALUES: "All values",
// Options
LAYOUT_OPTIONS: "Layout options",
GRAND_TOTALS: "Grand totals",
SUBTOTALS: "subTotals",
OFF_FOR_ROWS_AND_COLUMNS: "Off for rows and columns",
ON_FOR_ROWS_AND_COLUMNS: "On for rows and columns",
ON_FOR_ROWS: "On for rows only",
ON_FOR_COLUMNS: "On for columns only",
DO_NOT_SHOW_SUBTOTALS: "Do not show subtotals",
SHOW_ALL_SUBTOTALS: "Show all subtotals",
// Export PDF
CHOOSE_PAGE_ORIENTATION: "Choose page orientation",
LANDSCAPE: "Landscape",
```

```
  PORTRAIT: "Portrait"
};
```

2. Pass translations to flexmonster.embedPivotComponent():

```
flexmonster.embedPivotComponent("flexmonster/", "pivotContainer","100%", "500", {
  configUrl : "config.xml"
}, true, Labels);
```

11. API reference - JavaScript(http://www.flexmonster.com/api/)

12. API reference - Flex(http://www.flexmonster.com/flex-api/)