Maps

Overview

The map is an interactive widget that displays numeric data with geographic locations. The data can be supplied in a few different ways, depending on your needs. The map can use statically defined data, or the data can load dynamically from an RPG program or database.

Maps are a special type of chart widget, and you may notice in Visual Designer that the map properties are labelled as "chart" properties.

Maps can be used both the Genie environment as well as with Rich Display Files. However, many map options are unsupported in a Genie-only environment.

This page outlines the different attributes you can use with maps, and it provides limited examples. You can find a step-by-step example in the post, Creating an RPG driven Map.

Field Binding Dialog (Rich UI Only)

If you are using the Rich UI, then you can bind any of the map properties to variables in an RPG program. For more information about binding, please visit the Field Binding page.

Properties that are specific to map widgets are described below. In Visual Designer you'll find these properties by clicking on a map widget and looking under "Chart Properties" on the lower right side of the designer page.

Identification

chart response - Specifies a response field to be populated with the name of the data-point selected by the user.

Chart Settings

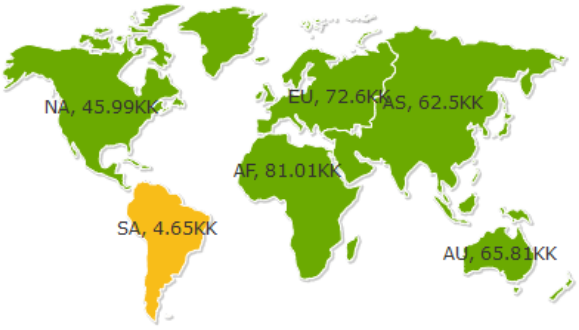
chart type - This property must remain empty for maps.

map type - A required property that identifies the type of map to display. Some common examples are: "world", "usa", and "europe". All map types are listed on the FusionMaps page under the *javaScript Alias* column. The map type is valid with or without the "maps/" string; that is, "maps/world" and "world" are both acceptable.

chart options - Specifies map options as a set of XML attributes that will be placed inside the FusionCharts **<chart>** tag. All attributes are found on the FusionCharts Attribute References. See also the Map Attributes section below.

> **Note:** If you are using RPG to generate the map's XML or JSON, then this field is ignored.
>
> Example: include the data value with the region label:

Chart Settings	
chart type	
map type	world
chart options	includeValuesInLabels="1"



> Example: hide region labels, but show them in tool tips.

Chart Settings	
chart type	
map type	usa
chart options	showlabels="0" showToolTip="1"

chart overlay - If the property is set to true, then the map background color will depend on the chart type. Otherwise, the chart background color will be transparent. Read further in the FusionCharts setTransparent method.

onchartclick - Specify JavaScript to handle the chart-click event. When the user clicks the map, then an event is generated with a parameter named, "name", whose value will be the name of the region clicked on the map. There are a few ways to handle the event.

1. You may specify inline JavaScript code:

onchartclick	console.log(name);

With inline code, a variable named "name" is automatically defined and set to the name of the data column/bar/point that was clicked.

2. Create an anonymous function:

```
(function(par1){
  console.log(par1);
});
```

When using a function to handle the click, the first parameter is assigned with the name of the data column/bar/point that was clicked.

3. Specify a user-defined function:

onchartclick	myfunction

When using a function to handle the click, the first parameter is assigned with the name of the data column/bar/point that was clicked. User defined functions can be defined in Custom External Javascript.

chart response - Bind this property to an RPG variable to allow the map to be clicked. When the user clicks on a location in the map, control is passed back to the RPG program, and the value of the bound field is filled with the clicked location.

Chart Data

One way to define the data presented on the map is by setting the names and values from within Visual Designer. Defining data this way, the *names* and *val ues* fields are comma-separated lists, and the lists must be the same length.

names - Specify the names of the geographic regions used by the map. The names corresponding to map regions are the Id's defined on the FusionCharts Specification Sheets. Use the exact Id from the specs; otherwise, the FusionCharts (FC) software doesn't recognize the data. For example, in the "europe" map "005" is recognized as Belgium, but "5" is not (as of FC version 3.8.0).

values - Specify the value associated with a region. The values can be displayed alongside the region's label, or the value can appear as a tool tip (when the mouse hovers over a region), depending on the map attributes.

Limitations: when using this method you cannot specify different colors for different regions; all regions will have the same default color.

Hint: it is possible to create a dynamic chart by binding the *names* and *values* properties to your RPG program. However, other methods may better suit your needs.

Database-Driven Chart

In Profound UI Version 5, Fix Pack 8.0 and later, maps can be populated by setting the Database-Driven properties:

Database-Driven Chart	
database file	TESTMAP1
name field	ID
value field	VALUE

Dynamic Chart

There are several ways to provide chart data dynamically, rather than setting constant name-value data in Designer or using Database-Driven maps: use a web service to provide JSON or XML, or let your RPG program generate the JSON or XML.

When using example XML or JSON from FusionCharts' website, please be aware that not all XML or JSON will work with the "onchartclick" or "chart response" properties. Please see the examples below for the form of XML and JSON that the map widget expects.

FusionCharts provides a simple guide for generating maps with XML or JSON. You may also find their Attribute Reference helpful.

Coloring the regions in a map is only possible with JSON or XML generated map. The a region's color is determined by the *colorRange* element in the XML or JSON.

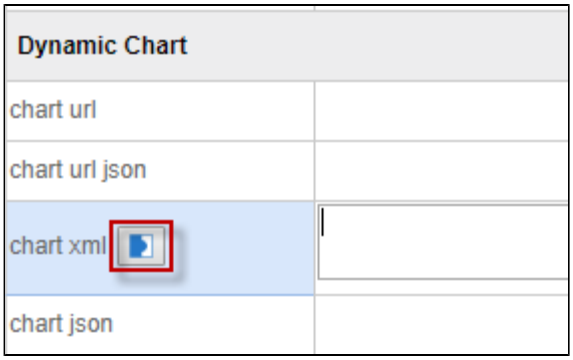
Binding "chart json" or "chart xml" Properties

See Field Binding for details about binding any property. To generate dynamic data, your RPG program can generate the XML or JSON text and put the result in a bound variable. Use either XML or JSON, but not both.

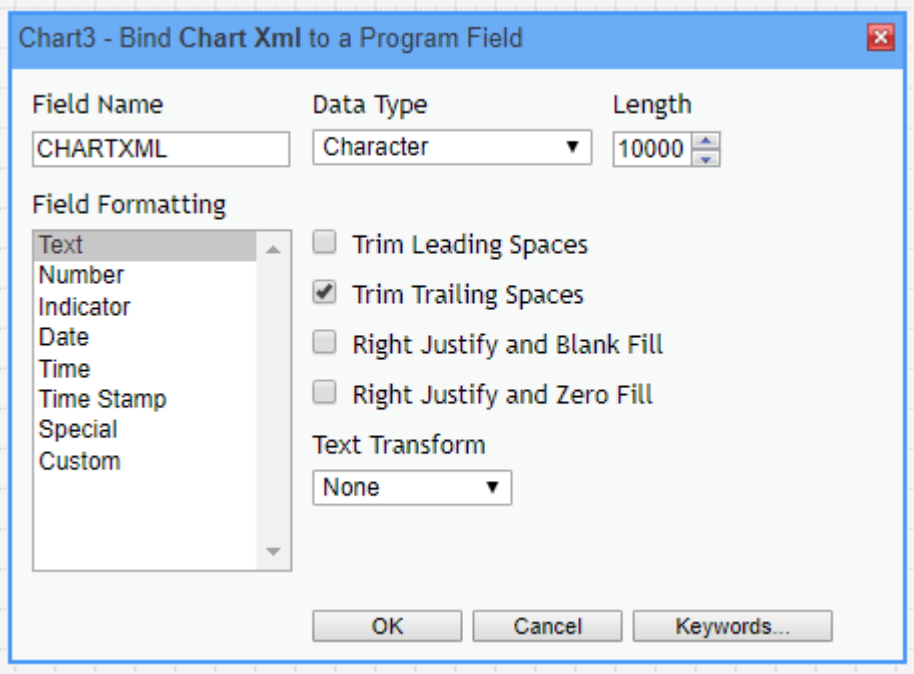
For a detailed example on creating a Rich display file and RPG program to generate map data, see Creating an RPG driven Map.

chart xml - Specify the raw XML used for generating the map. Below is a simple example of binding map XML so you can generate a dynamic map.

Click on the text "chart xml" to display the Binding button:



Click the Bind button to display a binding dialog:



Put in a field name that the RPG program will write to; e.g. CHARTXML. Use Character data type and use a length large enough to hold the XML you will generate; e.g. 10000.

Press OK, and you should see the chart xml property is bound to CHARTXML. (The green text indicates that the property is bound.)

chart xml	CHARTXML

With that property bound, your RPG program can write to the CHARTXML field, and that data will be used to generate the chart widget when the page loads.

chart json - Specify the JSON data used to generate the map. This is conceptually the same as *chart xml*. Use the example above as a guide, and bind to *chart json* instead. Perhaps use a different field name, such as CHARTJSON.

chart url - Instead of generating the map data via an RPG program, an alternative is to specify the URL of a web-service. The web-service must return the chart definition as XML, and the chart XML must correspond to the ***map type*** property; that is, if your ***map type*** property is set to *world*, then the XML response must have data for the 6 continents–NA, SA, EU, AF, AS, AU.

chart url json - As with "chart url", specify a URL of a web-service that generates JSON text.

For any of the three properties under the Dynamic Chart category, you must only use one. Furthermore, these methods should not be used in conjunction with the Chart Data properties (i.e. names/values), nor should Database-Driven chart be used with these properties.

Map Attributes

When using XML, the attributes go inside of the root <chart> tag in the form, *attributeName="value"*. The tag below shows some XML attributes that are available.

```
<chart caption="Global Population" theme="fint" formatnumberscale="0" numbersuffix="M"
showlabels="1" includenameinlabels="1" usesnameinlabels="0">
```

With JSON, the attributes go inside the "chart" object definition in the form, *"attributeName": "value"*. You can see a few examples below:

```
{
    "chart": {
        "caption": "Global Population",
        "theme": "fint",
        "formatNumberScale": "0",
        "numberSuffix": "M",
        "includeNameInLabels": "1",
        "showLabels": "1"
    },
```

Some commonly used attributes are explained here. Boolean attributes require a value of 1 for true and 0 for false.

Attribute	Type	Description	Default
showToolTip	Boolean	When true, hovering the mouse pointer over a region causes the data value to appear.	True
showLabels	Boolean	When false, the map draws the regions without names.	True for XML, False for JSON
numberSuffix	String	This suffix is displayed after a data value for a region. For example, if your data values are scaled to thousands, then you might use "K" here.	Empty string
includeValueInLabels	Boolean	When true, the data value for a region is shown after the region's name.	False

See all attributes at http://www.fusioncharts.com/dev/maps/attribute-reference.html

Adding Links to XML and JSON Charts

FusionCharts allow users to add links to maps if certain JSON or XML values are set. If the "chart response" or "onchartclick" properties are setup, then Profound UI will automatically add links to areas on a Map widget. However, if the XML or JSON is structured incorrectly, the map may not be clickable, even though it may render correctly.

FusionCharts supports a few different structures of XML and JSON for maps. Any example from www.fusioncharts.com should render in a Profound UI Map widget. Note: regular charts use different property names in JSON and different attributes in XML tags.

For a map to be clickable:

XML

The XML for a map should use a <chart> tag (not a <map> tag). The data must be inside <set> tags using the attributes "id" and "value".

Example:

```
<chart caption="Global Population" theme="fint" formatnumberscale="0" numbersuffix="M">
   <set id="NA" value="515" />
   <set id="SA" value="373" />
   <set id="AS" value="3875" />
   <set id="EU" value="727" />
   <set id="AF" value="885" />
   <set id="AU" value="32" />
</chart>
```

JSON

The JSON for a map should use the property names "id" and "value".

Example:

```
{
    "chart": {
        "caption": "Global Population",
        "theme": "fint",
        "formatNumberScale": "0",
        "numberSuffix": "M"
    },
    "data": [
        { "id": "NA", "value": "515" },
        { "id": "SA", "value": "373" },
        { "id": "AS", "value": "3875" },
        { "id": "EU", "value": "727" },
        { "id": "AF", "value": "885" },
        { "id": "AU", "value": "32" }
    ]
}
```