European Commission

Map Data Visualizer

User Guide

Joint Research Centre

Release	Issue	Date
	6	June 2019

Disclaimer

On any of the MARS pages you may find reference to a certain software package, a particular contractor, or group of contractors, the use of one or another sensor product, etc. In all cases, unless specifically stated, this does not indicate any preference of the Commission for that particular product, party or parties. When relevant, we include links to pages that give you more information about the references.

Feel free to contact us, in case you need additional explanations or information.

Contents

Help contents

1

This Guide describes how to use Map Data Visualizer (MDV) to explore and analyze the results of model simulations.

The topics are organized as follows:

Topic	Contents
“Introduction” on page 5	What MDV can be used for, how to install it, and workspace overview.
“Using MDV to create graphs and maps” on page 11	How to use MDV to graphically view the data (either as a map or as a graph) and to customize the way data are displayed.
“Using the configuration files to customize MDV” on page 43	How expert users can customize the tool to suit their specific needs.

Tip:

Since MDV can be used as a BioMA Spatial plugin, for further information visit the **Agri4Cast Software Portal** (https://agri4cast.jrc.ec.europa.eu/DataPortal/Index.aspx?o=s.)

Introduction

2

This chapter describes the main scope of Map Data Visualizer that allows creating and displaying maps and graphs either for bulletins or for working on models simulations (created through BioMA Spatial).

It is organized into the following topics:

See also

What Map Data Visualizer is used for

Map Data Visualizer (MDV) allows inspecting and visually analyzing the outputs of simulations.

These are important tasks in order to understand a model's behaviour.

MDV can act as a **map** visualizer for showing one indicator spatially or as a **graph** visualizer for showing values for specific areas as time series.

See "Viewing data as a Map" on page 12 and "Viewing data as a graph" on page 29 for detailed instructions.

How MDV can be used

The MDV tool can be used in two ways:

- **As a standalone tool**, to select and load a specific output.
- **As a BioMA plug-in**, both from within the BioMA Spatial and the BioMA Point graphical user interfaces. After running a simulation, you can use the MDV plug-in to analyze the results.

Available functions are the same in both cases; however, the way you select the data to analyze differs slightly.

See also:

- "Installing the application" on page 7
- "Workspace Overview" on page 9
- "Using MDV to create graphs and maps" on page 11

Installing the application

Prerequisites

In order to install and run Map Data Visualizer, the following prerequisites must be fulfilled:

Hardware prerequisites

- Operative System: Windows XP/Vista/7 (32 or 64 bit)

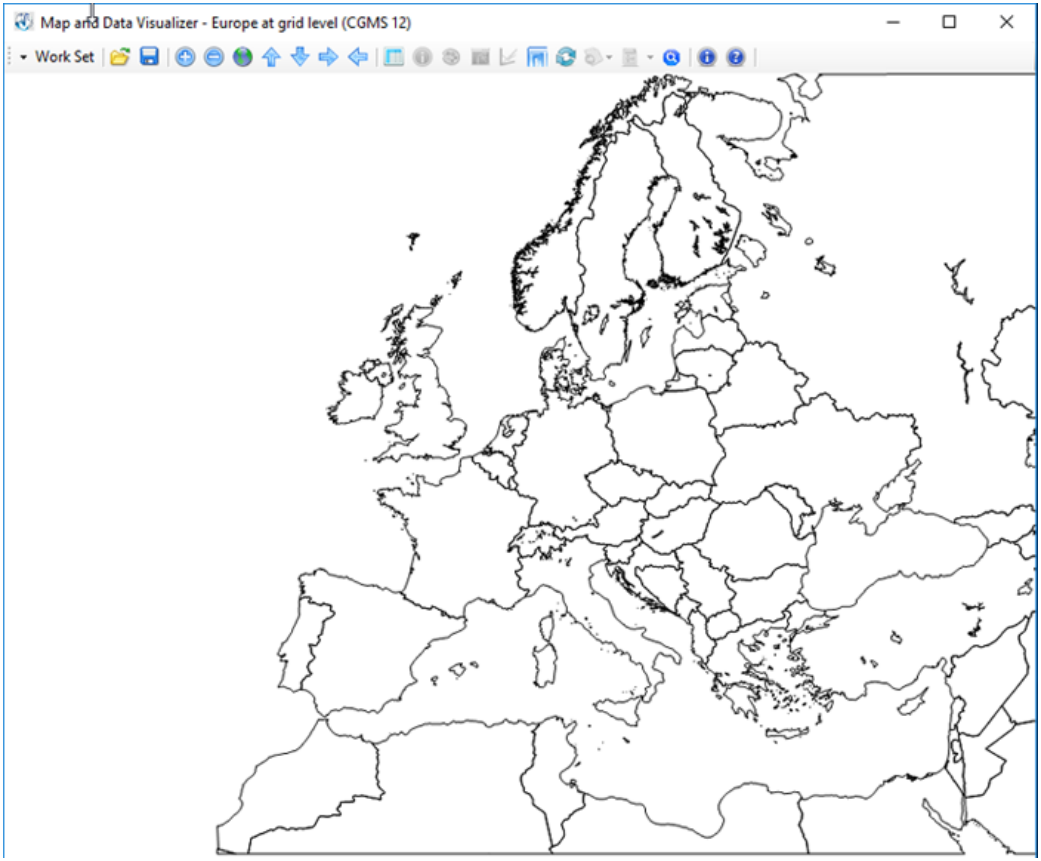
Software prerequisites

- **NET 4.5 Framework**. To install it, click here. Follow the product's documentation, if needed.

Installation procedure

Do the following:

1. Go to your installation folder.
2. Run the **setup.exe** file. The main window of MDV will be launched:



For information, see:

- “Workspace Overview” on page 9

Workspace Overview

The following table summarizes the actions you can perform by using the Map Data Visualizer toolbar.

A link is provided to the relevant topic, when needed:

Toolbar button	What you can do
	Save the workspace as it is in that moment (map, query, legenda, etc.). This allows you to re-load the same workspace without the need of specifying the parameters.
	Magnify, reduce, and view the whole map, respectively. For the same purpose, you might also use the mouse wheel.
	Move the map up, down, right, and left, respectively. For the same purpose, you might also use the mouse wheel.
	Open the selection form that allows specifying the parameters of the simulation to analyze. See “Viewing data as a Map” on page 12 and “Viewing data as a graph” on page 29.
	Display the Web mapper in the Map Visualizer window.
	Open the popup that shows information on the selected data. To select the data, click the desired area on the map.
	Change the default legend and customize the map layout, respectively. See “Changing the Map Visualizer’s map layout” on page 19.
	See the selected data as a graph either by selecting specific cells or not, respectively. See “Viewing data as a graph” on page 29.
	Refresh the start and end dates for multiple queries in a single operation. See “Refreshing more queries dates at once” on page 36.
	Export the map in a variety of file formats by selecting it from the dropdown list. See “Exporting the map” on page 22.

Toolbar button	What you can do
	Create a XML report of a map's configuration, that is, create maps in batch mode. This allows you using the same configuration to create a number of similar maps, without the need of specifying the same settings more times. See "Creating a report from a map's configuration" on page 23.
	Viewing the log of all the executed queries, inspecting and checking for errors, if any. See "Viewing the Query log file" on page 40.

Related topics:

- "Using MDV to create graphs and maps" on page 11

Using MDV to create graphs and maps

3

Map Data Visualizer allows viewing the data in two ways:

- As a **Map**
- As a **Graph**

In both cases, the first operation to perform is selecting the **Work Set**, that is, the working area, which includes specific data and queries in order to select the data you want to see.

Then, you can specify and customize how data are displayed.

For detailed information:

- “Viewing data as a Map” on page 12
- “Changing the Map Visualizer’s map layout” on page 19
- “Exporting the map” on page 22
- “Creating a report from a map’s configuration” on page 23
- “Viewing data as a graph” on page 29
- “Refreshing more queries dates at once” on page 37
- “Viewing the Query log file” on page 41

Viewing data as a Map

The way you select the data show differs slightly if you're using MDV as a BioMA plug-in, or as a stand-alone application.

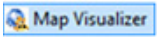
The first operation to perform is selecting the maps.

Selecting the data to view

This step allows you to select the maps to work on by choosing the database you want to connect to, that is, the database to query and the typology of data contained.

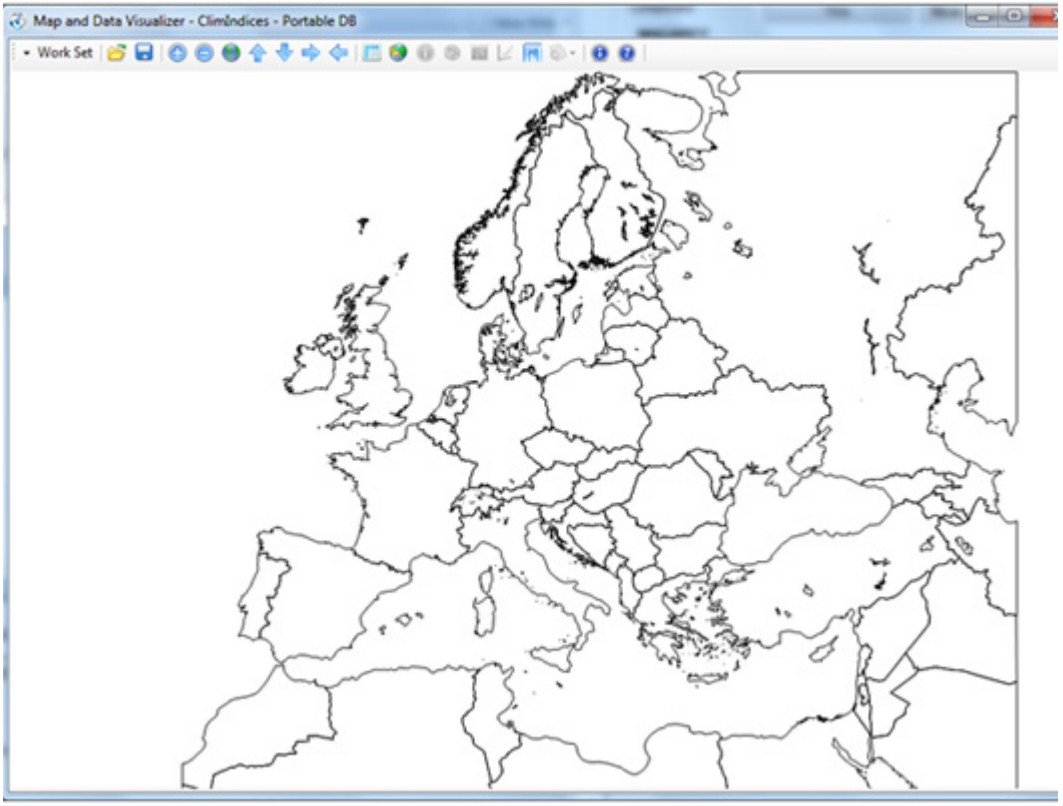
1 Launch Map Data Visualizer (MDV). (See “Installing the application” on page 7, if needed).

Note:

If you are using MDV as a BioMA plug-in, click the 

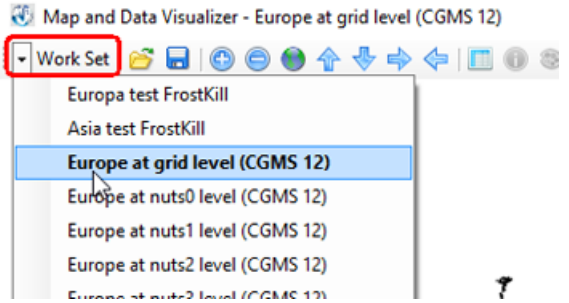
button in the application’s toolbar.

The **Map Data Visualizer** window is displayed:



(See “Workspace Overview” on page 9 for a description of all the toolbar buttons).

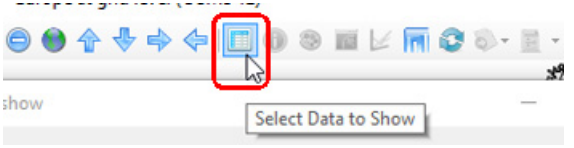
2 Specify the work area: from the **Work Set** dropdown list, select the database you want to connect to, that is, the database to query and the typology of data contained:



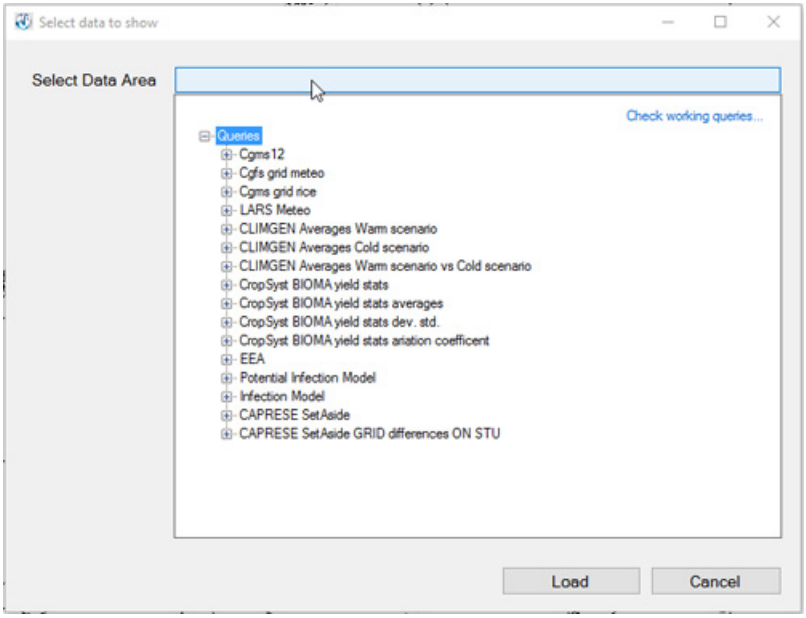
Note:

If you are using MDV as a BioMA plug-in, the **Work Set** is already set according to the simulation you are working at.

3 To see the available maps, click the **Show data as a Map** button on the MDV toolbar.

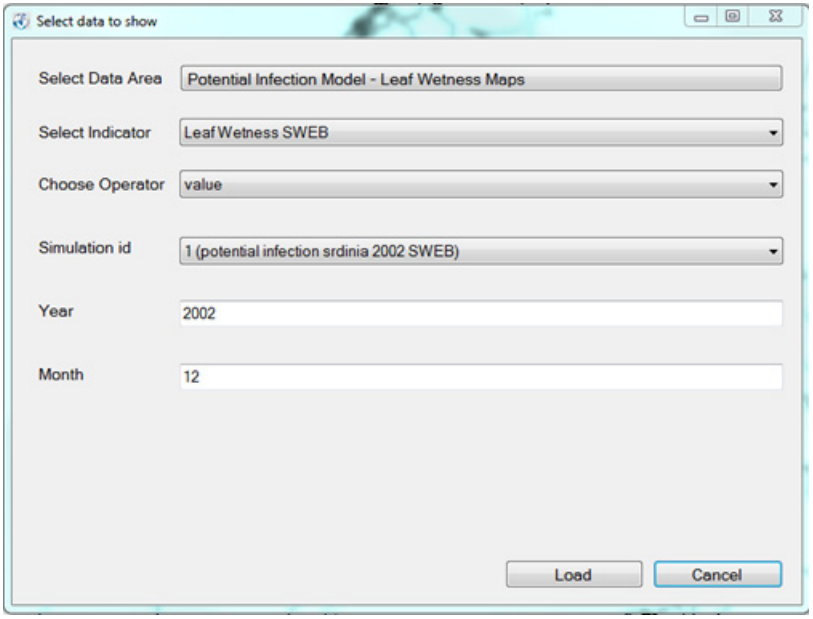


4 In the **Select data to show** window that is displayed, click the **Select Data Area** field, and then expand the **Queries** list to select the maps:



5 The **Select Indicator** field will be displayed in the window. Make your selection to display in the window other fields to set.

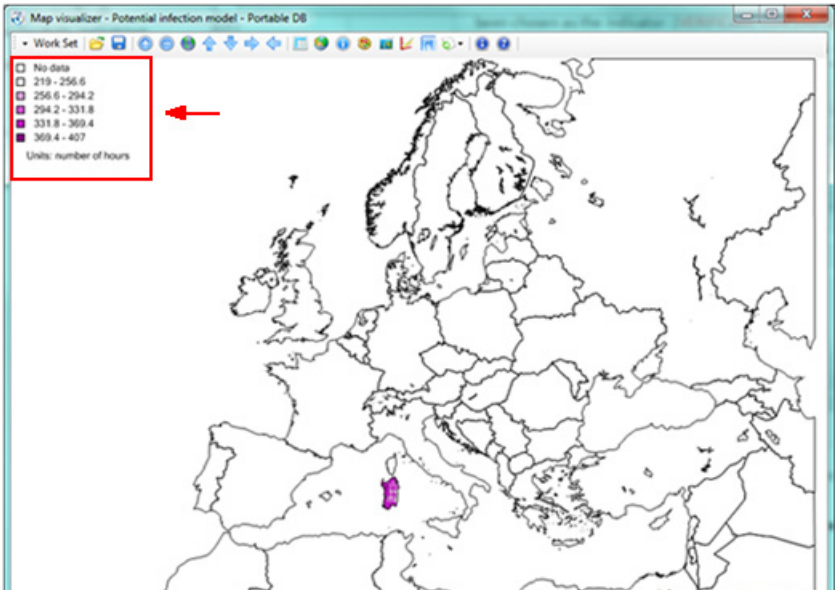
The following shows an example where the **Leaf Wetness SWEB** has been chosen as the indicator:



Note: The available fields depend on the selection. The list of the available indicators is defined in a set of XML files that can be edited by expert users during the configuration phase of the plug-in. This editing allows adding new indicators and modifying or deleting the existing ones.

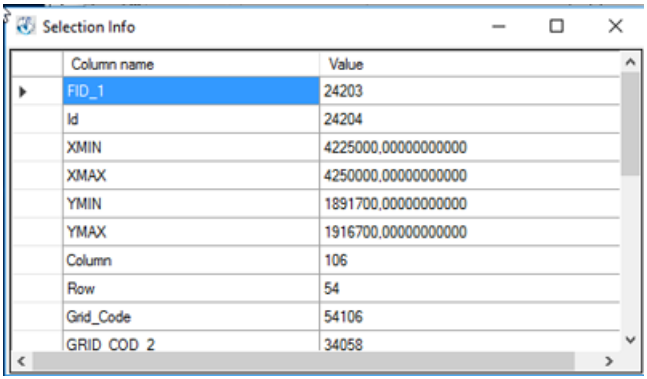
6 When all parameters have been specified, click **Load**. (See “Creating a report from a map’s configuration” on page 23 if you want to save the configuration as an XML report for further reuse).

The selected indicator is displayed in the map with a default legend:



7 To inspect the values in the map, click the **Inspect Values** button in the toolbar and then click the map:

8 A popup window will display detailed information on the selection:



(For a description of all toolbar buttons, see “Workspace Overview” on page 9).

See also:

- “Changing the Map Visualizer’s map layout” on page 19
- “Exporting the map” on page 22
- “Creating a report from a map’s configuration” on page 23

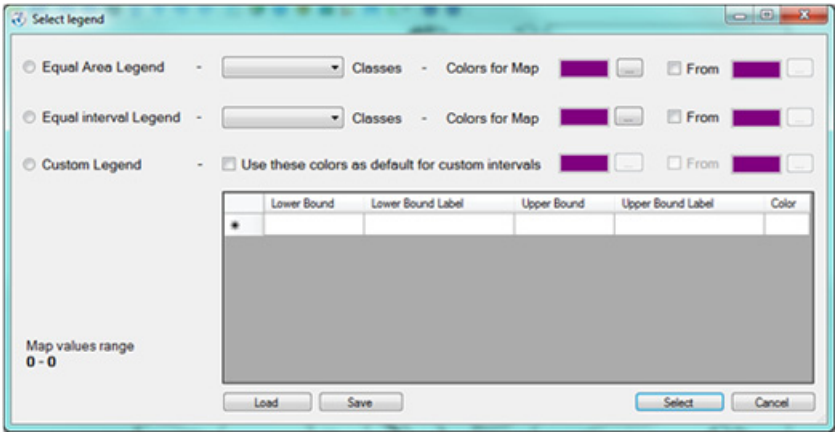
Related topics:

- “Viewing data as a graph” on page 29
- “Refreshing more queries dates at once” on page 37

Changing the Map Visualizer's map layout

To set the legend to apply to the map:

1 Click the **Set legend** button on the MDV toolbar. The following dialog is displayed:



Note: If you already created and saved a legend, click the **Load** button and then select the .leg file you want to use.

2 Make your selection, considering that:

- **Equal Area Legend** - This option allows selecting fro the dropdown list the number of classes to be displayed. The data will be splitted so as to display an uniform distribution of data on the map.
- **Equal interval Legend** - This option allows selecting fro the dropdown list the number of classes to be displayed. The data will be splitted and their values divided into equal intervals.
- **Custom Legend** - This option allows the user to define the number of classes to be visualized and their bounds, as well as to define a custom range of colors.

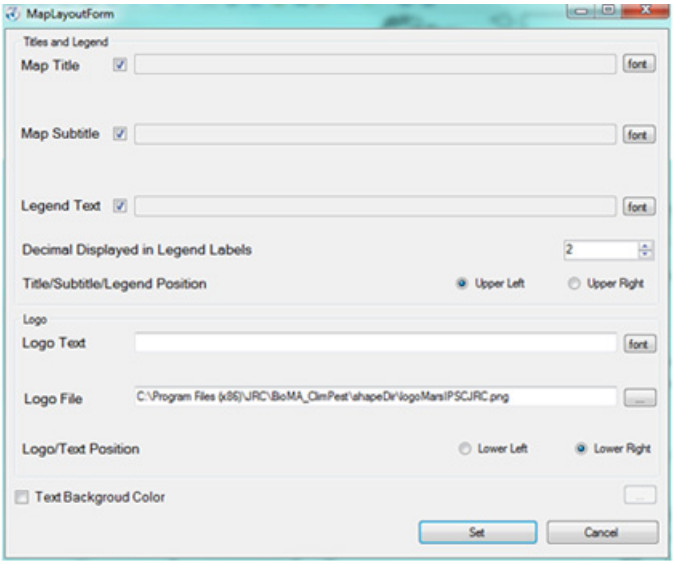
3 Use the **Colors for map** function to set the desired ranges of colors for the legend. If you selected **Custom Legend**, use the table to set the bounds and their colors.

4 When finished, do one or both of the following:

- Click the **Save** button to save the legend as a `.leg` file for a future re-use. You will then use the **Load** button to use the legend.
- Click **Select** to apply your settings and to return to the map.

To add information to the map:

1 Click the  **Set Map Layout** button on the MDV toolbar. The following dialog is displayed:



2 Do the following, as needed:

- To specify a title for the map, deselect the **Map Title** checkbox to enable the text entering mode. Use the **font** button to select the font.
- To specify a subtitle for the map, deselect the **Map Subtitle** checkbox the enable the text entering mode. Use the **font** button to select the font.
- To specify a text for the legend (e.g., the units displayed), deselect the **Legend Text** checkbox to enable the text entering mode.
- Specify the number of **Decimal Displayed** and the position for the texts you entered.
- Specify a **Logo Text** and select a **Logo File**, as well as their position in the map.

3 When you are finished, click **Set** to return the map and see your changes.

Tip:

To further customize the graphic layout of the map (e.g., to specify a position, or a colour, or a size of the Map Title, advanced users can edit the XML template that is provided with the MDV (see "Defining the maps' layouts" on page 55).

Related topics:

- "Changing the Map Visualizer's map layout" on page 19
- "Exporting the map" on page 22
- "Creating a report from a map's configuration" on page 23
- "Viewing data as a graph" on page 29
- "Refreshing more queries dates at once" on page 37

Exporting the map

To export the map into a file:

1 On the MDV toolbar, click the  **Export Map** button.

2 From the dropdown list select the desired format. The available options are:

- **Pdf Image Export** - Exports the map as PDF file
- **Png Image Export** - Exports the map as an image in PNG format
- **Csv Data Export** - Exports the data showed in the map in a CSV file.
- **Shp File Export** - Exports the data and the geographical information in a shapefile.

Note:

There might be other options that allow creating exports with specific custom configurations. However, these are only available if you defined them in the **Mapper.config** configuration file of Map Data Visualizer (see "Defining the maps' layouts" on page 55).

3 Choose a name for your file and click **Save**.

Related topics:

- "Viewing data as a Map" on page 12
- "Changing the Map Visualizer's map layout" on page 19
- "Creating a report from a map's configuration" on page 23
- "Viewing data as a graph" on page 29
- "Refreshing more queries dates at once" on page 37

Creating a report from a map's configuration

Once you have configured the parameters for the data to show as a map, you can save the configuration for a further reuse.

Let's suppose, for example, that you want to create a map that shows the same weather data for more Countries.

Rather than having to manually repeat the same procedure many times, you can save the map's configuration as a XML report and then use it as a basis for other maps.

To configure the map:

1 Launch **Map Data Visualizer**.

2 Select the data you want to see as a graph, that is:

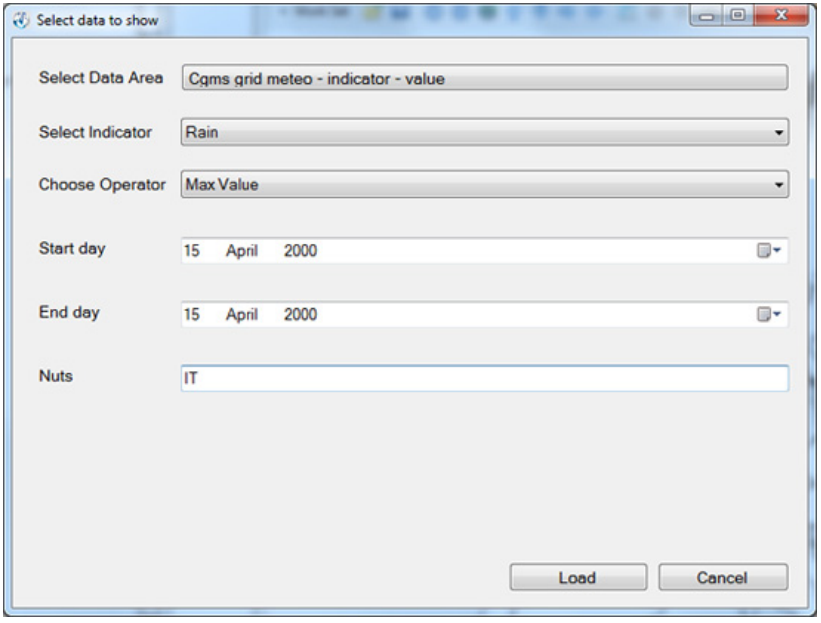
a. Select a **Work Set**.

b. Select the **Query** in the **Select data to show** window.

(For detailed instructions, see section "Viewing data as a Map" on page 12).

The following shows an example where we want to view a map for the rain values in Italy for a specific period of time:

Figure 1 Select data to show window



3 Once you have set all the fields as required, click **Load**. The map displays in the MDV window.

4 Customize the map's legend as required, then save it as a `.leg` file (see "To set the legend to apply to the map:" on page 19 for detailed instructions).

To create and save the XML report:

1 Create a XML file based on your configurations. Use the following example as a guideline (in red the tags you must set):

Example

```
<?xml version="1.0" encoding="utf-8" ?>
<Report name="RainByCountryReport" workDir="d:\maps"
   outtype="" outfile="">

   <QuerySet name="Rain by country report">


    <Map name="RainIT"  index="1"
  mapLayout="AvemacDefault" mapLegend="leg_perc_35">
     <Query name="Cgms grid meteo - indicator - value"
  indicator="Rain">
      <Filter name="Start day" value="15/04/2000" />
      <Filter name="End day" value="15/04/2000" />
      <Filter name="Nuts" value="IT" />
      <Operator value="Max Value" />
      <Groupby value="" />
    </Query>
    </Map>
</Query>
    </Map>
</QuerySet>
```

Where:

XML tag	Set as
Report name	A unique name of your choice for the report.
workDir	The directory where the maps will be saved.
QuerySet name	A name for the query configuration.
Map name	A map's title.

XML tag	Set as
mapLayout	The name of the map's graphic layout. This layout can be based on the XML templates (pdf and png) provided with the installation kit. Optionally, these templates can be editable to further customize the layout. (See "Editing the XML layout template" on page 56).
mapLegend	The name of the legend that defines the map's colours. Legends are saved as .leg files. In our example, is **leg_perc_35** . (For information on creating a legend file, see "To set the legend to apply to the map:" on page 19). The legend files to be used in a report must be defined within the XML report file (see Step 7, here below).
Query name	The selection you made in the **Select Data Area** dropdown list within the **Select data to show** window (see Figure 1 on page 23)**.**
Filter name	The parameters you set in the **Start day**, **End day**, and **Nuts** fields within the **Select data to show** window (see Figure 1 on page 23)**.**
Operator value	The selection you made in the Choose Operator dropdown list within the **Select data to show** window (see Figure 1 on page 23)**.**

2 Save the XML file with a meaningful name.

3 Go to the MDV installation folder and open the **Mapper.config** file, that is the main configuration file of Map Data Visualizer

4 In the `<ReportSection>` of the file, enter the name of the XML report you created. (For detailed instructions on how to do it, see "Mapper.Config file" - "Defining the reports" on page 53).

Note:

You don't need to repeat this step again in the future. The report must be entered in the Mapper.config file only once.

5 In the newly created XML report add the following section for each Country you want to view the data for. For this purpose, change as required the values that are highlighted in red (in our example, we are creating the same map for Austria):

Example

```
<Map name="RainAT"  index="1" mapLayout="AvemacDefault"
   mapLegend="leg_perc_35">
      <Query name="Cgms grid meteo - indicator - value"
   indicator="Rain">
      <Filter name="Start day" value="15/04/2000" />
      <Filter name="End day" value="15/04/2000" />
      <Filter name="Nuts" value="AT" />
      <Operator value="Max Value" />
      <Groupby value="" />
     </Query>
     </Map>
</Query>
    </Map>
</QuerySet>
```

6 Add as many sections as needed.

7 Finally, at the end of the XML report, after the **</QuerySet>** closing tag, enter the legend (.leg) file names that you want to use. See the following example:

Example

```
</QuerySet>

<MapLayout name="AvemacDefault"
   template="JRCMapGridPngTemplate.xml"/>
 <MapLegend name="YieldPercent"
   file="leg_PercentCentredOnZero.leg" />
<MapLegend name="leg_YieldGap" file="leg_YieldGap.leg"
   />
<MapLegend name="YieldGap" file="leg_YieldGap.leg" />
<MapLegend name="YieldAbsCV"
   file="leg_YieldCV_absdif_all.leg" />
<MapLegend name="leg_default" file="leg_default.leg" />
<MapLegend name="legend_default_eqArea"
   file="legend_default_eqArea.leg" />
<MapLegend name="leg_perc_35" file="leg_perc_35.leg" />
<MapLegend name="leg_diftemp" file="leg_diftemp.leg" />

</Report>
```

Where:

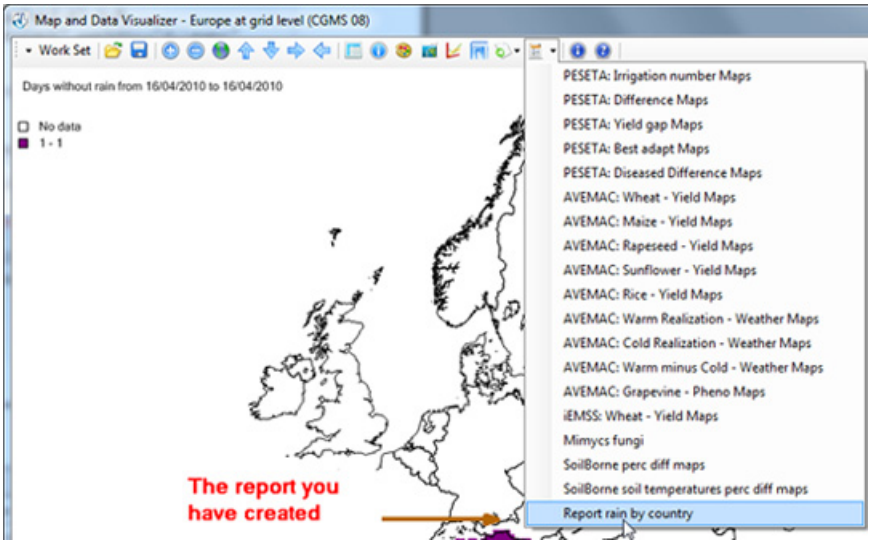
XML tag	Set as
mapLayout	The name of the map's graphic layout, which is based on the XML template.
template	The editable XML template as defined within the **`mapper.config`** file (see "Editing the XML layout template" on page 56).
mapLegend	The name of the legend that defines the map's colours. These settings are saved as a `.leg` file.
file	The physical `.leg` file that includes the map legend settings.

8 Save the XML file again.

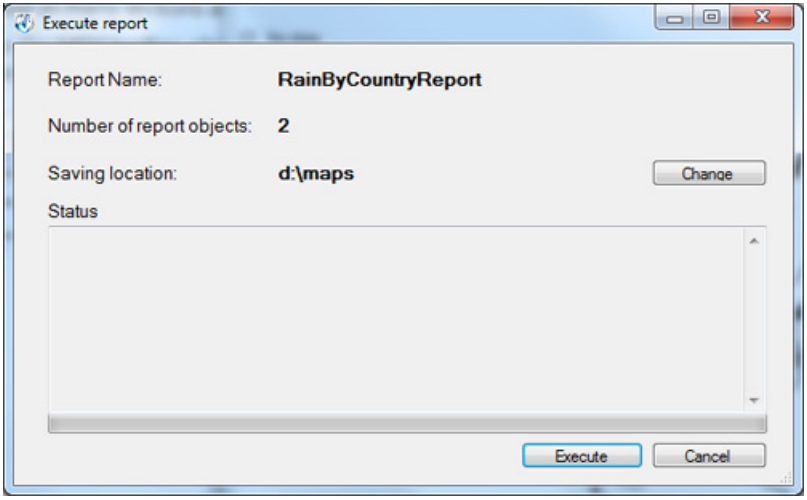
To run the report and create the maps:

1 Restart MDV.

2 In the toolbar, click the  **Create Report** button. The report you created is listed in the dropdown list:



3 Select the report, then wait for the following window to display:



Note that the **Number of report objects** corresponds to the elements you entered in the XML report (in our example, these are Italy and Austria).

4 Click **Change** next to the **Saving location** field if you want to modify the folder where the maps will be saved.

5 Click **Execute**, then wait for the maps creation to complete.

Note that this operation may require few minutes, depending on the number of maps you are creating.

Related topics:

- "Viewing data as a Map" on page 12
- "Changing the Map Visualizer's map layout" on page 19
- "Exporting the map" on page 22
- "Viewing data as a graph" on page 29
- "Refreshing more queries dates at once" on page 37

Viewing data as a graph

Once the map is displayed and a **Work Set** has been selected, the user can view the data as a graph.

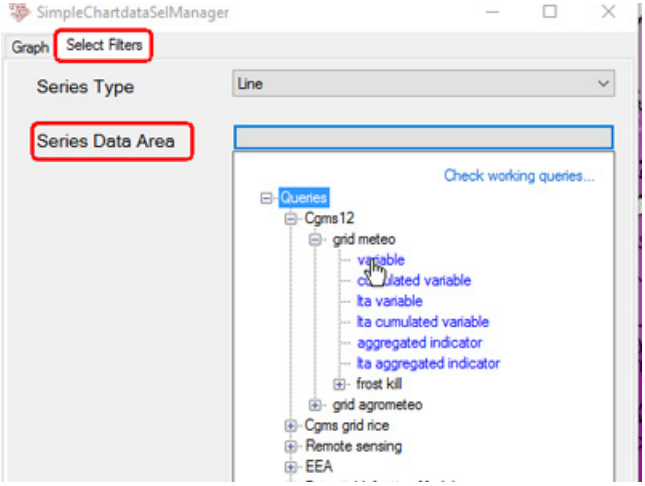
In particular:

Viewing the graph by selecting the grid

To select a specific geographical element and draw a graph:

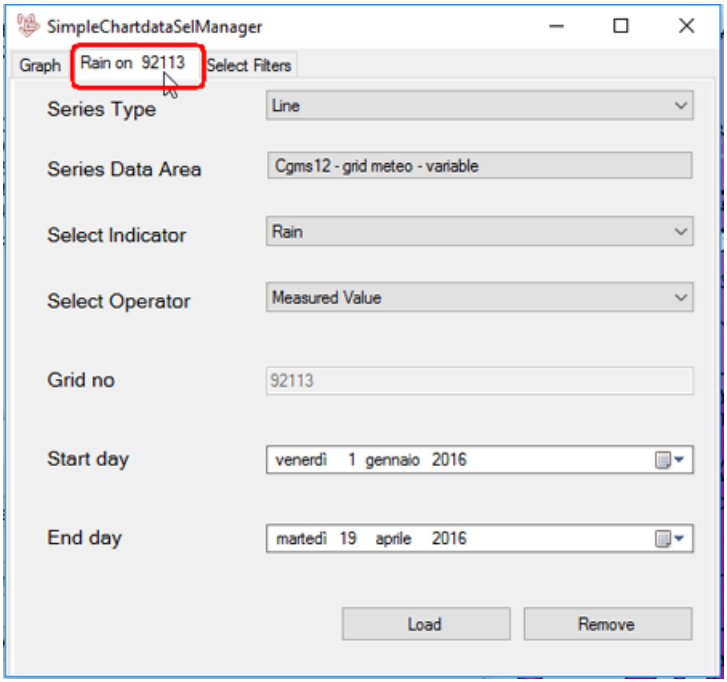
1. Select the **Work Set** (see "Selecting the data to view" on page 12).
2. Click the **See Graphs for the Item Selected** button. Note that the mouse pointer changes to an arrow for you to select the element(s) in the map.
3. Click the desired elements on the map. A popup menu is displayed:

4. Select the **Show Graphs for the Item/s Selected** menu item. The window to select the filters will be displayed.
5. Select the type of graph from the **Series Type** dropdown list (**Line** or **Bar**) and the data to show from the **Series Data Area** dropdown list:

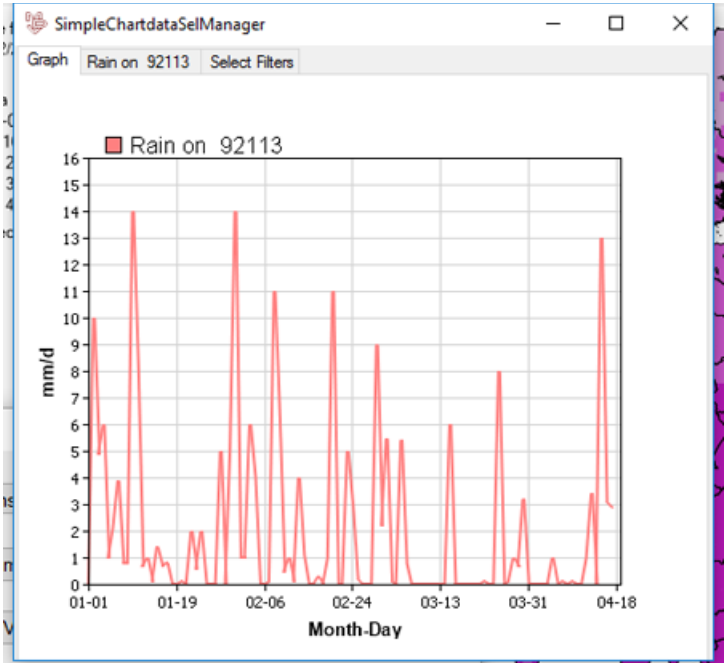


6 Fill-in the other fields as required. Note that, since you selected an area in the map, the **Grid** parameter is already set to the selected geographical location.

7 When finished, click **Load**. As a result a new tab is created (in this example, the **Rain on 92113** (**Grid no**.):



8 Select the **Graph** tab to view the result:



For the geographical element selected, the graph visualizer shows one or more indicators in function of time.

Note that you can set both the chart style and the line style, being this latter the name of the series displayed; it is particularly useful if you are creating a graph with more lines (you can display up to then lines).

9 To create more graphs, e.g. for data comparison, go back to the **Select Filters** tab and repeat the procedure.

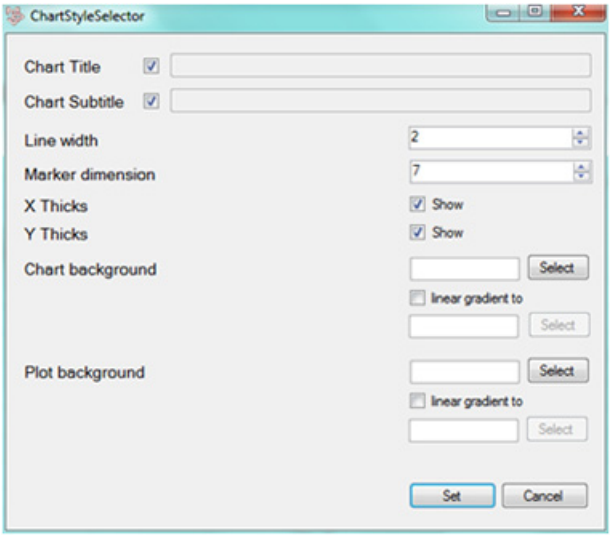
See also:

- “Setting the chart style” on page 33
- “Setting the line style” on page 34

Setting the chart style

To set the chart style:

1 Right-click the chart and then select, from the popup menu, **Set chart style**. The following dialog is displayed:



2 Set the parameters to show the chart as required. Note that:

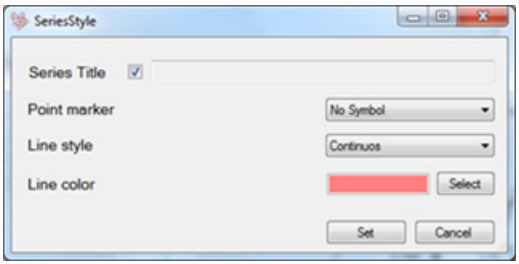
- To enter a specific **Title/Subtitle** for your chart, deselect the relevant checkbox and enter the name.
- To set the **Line width** and the **Marker dimension**, use the dropdown box.
- To **Show** the grid for the **X** and **Y** coordinates, leave the relevant checkbox selected.
- To set the a color for the background, click **Select** next to **Chart background**, and then choose a color from popup. You can also set a gradient by clicking **Select** next to the **linear gradient to** checkbox.
- To set a color for the plot, click **Select** next to **Plot background**, and then choose a color from popup. You can also set a gradient by clicking **Select** next to the **linear gradient to** checkbox.

3 When you are satisfied with your settings, click **Set** to return to the chart and see your changes.

Setting the line style

To set the line style:

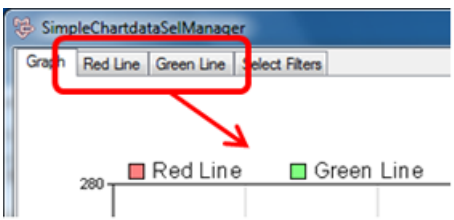
1 Right-click the default series name above the graph (**New Series**), and then select **Set line style**. The following dialog is displayed:



2 Deselect the checkbox next to **Series Title** to enable the text editing mode and then specify the desired name for the series.

3 Set the other parameters, e.g. the **Line color**, as required and then click **Set** to return to your graph.

Note that the tab name in the graph changes to reflect your settings:



4 Use the tabs to verify the source of the displayed data, if required.

Tip:

To end the selection mode, click the  **Show Graphs for the items selected** button on the MDV toolbar.

Showing a graph without selecting the items

To show a graph without selecting the geographical elements:

Note: When using this method be sure that you know the location ID, which must be specified as the **Grid** field in the graph selector window.

To display a graph for the active map, without selecting a specific element, do the following:

1. Click the **Show Graphs without select the item on the map** button.
2. The graph selector window is displayed. Follow the instructions as described in the previous section “Viewing the graph by selecting the grid” on page 29.

Related topics:

- “Viewing data as a Map” on page 12
- “Changing the Map Visualizer’s map layout” on page 19
- “Exporting the map” on page 22
- “Refreshing more queries dates at once” on page 37

Refreshing more queries dates at once

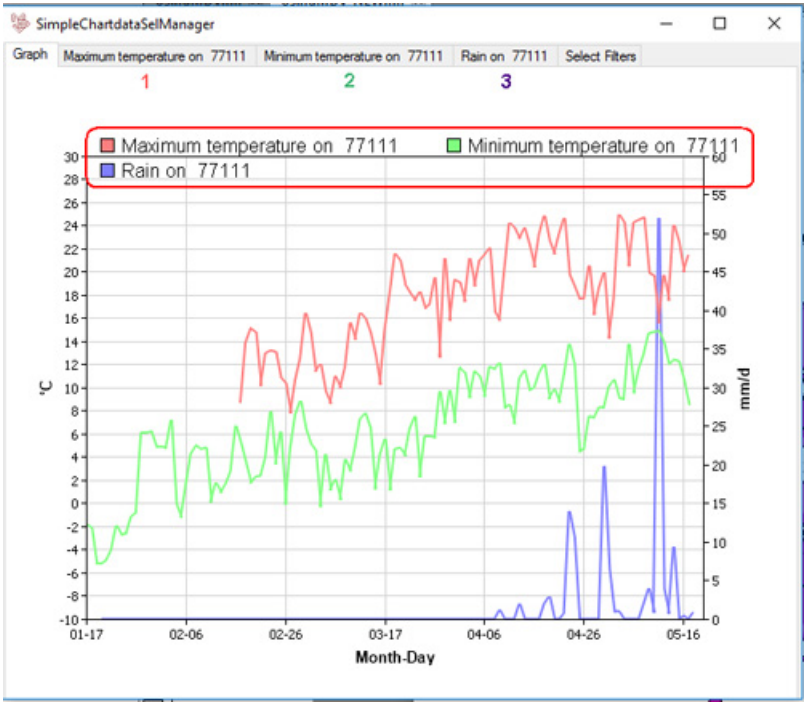
This function allows refreshing and update the start and end dates for a number of queries in a single operation.

By using it you can reuse the items (graphs and maps) without the need to rework them one by one.

In this section, an easy-to-follow example is provided that shows how to use this function.

Example

Let's suppose that you created more items for data comparison (see "Viewing data as a graph" on page 29 for further information):

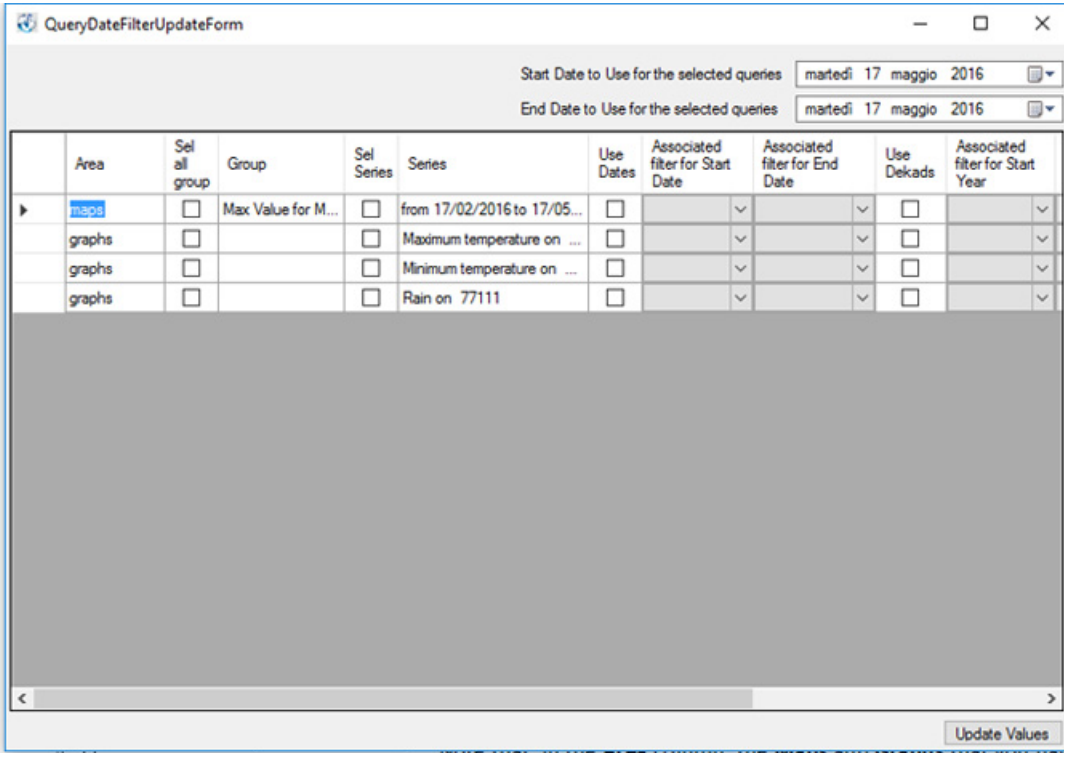


For example, you have created a graph to observe the temperature in the arable lands of a specific Country, and then another to observe the rain, and you want to reuse this set of graphs for a different time range.

To refresh the queries dates in one single operation:

1 On the MDV toolbar, click the **Refresh queries start end dates** button.

The following is displayed:



All the performed queries are displayed in the list.

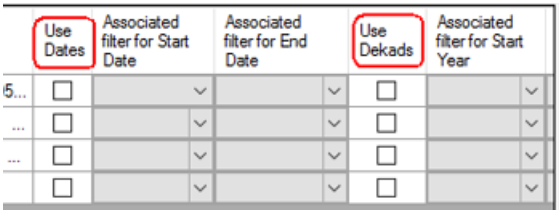
Note that, in the **Area** column, the **Maps** and **Graphs** that you have created for a map are listed and organized by **Group**.

2 Do one of the following:

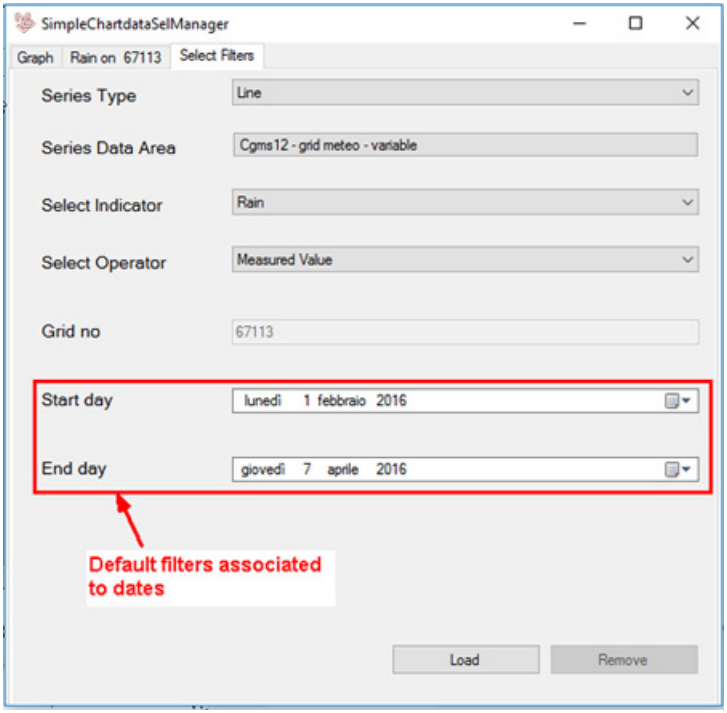
a. To select a single graph, click the corresponding checkbox in the **Sel. Series** column.

b. To select all the graphs at once, click a **graphs** checkbox in the **Sel. all group** column.

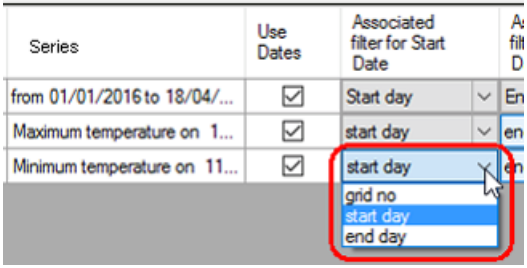
3 Depending on the query, select the **Use Dates** (to use a single days) or the **Use Dekads** checkbox to enable the dropdown lists.



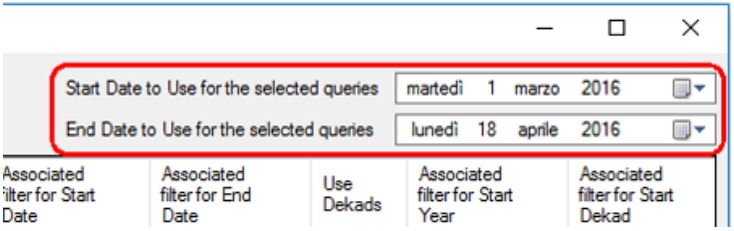
4 From the dropdown lists, select the filter that must be used for start and end dates association: that is, select the fields that were used in the query to specify the dates in the **Select Filters** window:



In the query above, the default fields (**Start day** and **End day**) have been used. In this case, in the Refresh Queries window, the following must be selected:



5 When finished, set the new dates you want to use for the selected queries as shown below:



6 Click the **Update Values** button in the upper right corner of the window to update the selected queries in one single operation.

See also:

- “Viewing the Query log file” on page 41

Viewing the Query log file

It is possible to display the log of all the performed queries:

1. On the MDV toolbar, click the **Query log** button. A window is displayed listing all the performed queries.
2. Inspect the queries list and/or check for errors, as needed.
3. Once done, click the **Refresh** button to clear the list.

Related topics:

- “Viewing data as a Map” on page 12
- “Viewing data as a graph” on page 29
- “Refreshing more queries dates at once” on page 37
- “Using the configuration files to customize MDV” on page 43

Using the configuration files to customize MDV

4

Note:

This topic is targeted to advanced users who create and modify the configuration files in order to customize the application's environment,

Map Data Visualizer allows for customization: as an advanced user, you can use the XML configuration files to add specific environments to the available worksets.

The following table summarizes the provided files:

Configuration file (XML)	Description
"Mapper.Config file" on page 44	Workset and environment definition (GIS layers, query files links, database connections).
"Defining the query files" on page 60	The queries to the database.

Related topics:

- "Using MDV to create graphs and maps" on page 11

Mapper.Config file

The `Mapper.config` XML file is the main configuration file of Map Data Visualizer as it includes the worksets definitions.

About the worksets

The worksets determine the available environments that the user can choose when using MDV. A workset is composed by three main elements: its name, the GIS layers (shapefiles), and a list of queries that allow extracting the data according to the shapefiles.

The following sections describe how to customize each element:

- “Defining the Workset” on page 45
- “Defining the GIS Layers” on page 47
- “Defining the query files links” on page 49
- “Defining the database connections” on page 51
- “Defining the reports” on page 53
- “Defining the maps’ layouts” on page 55
- “Defining other settings” on page 58

Related topics:

- “Defining the query files” on page 60

Defining the Workset

The `ConfigurationSetsSection` of the `mapper.config` file allows defining the available worksets.

You can add a workset by adding a new element to the `ConfigurationSettings` list.

The following shows the relevant section within the `mapper.config` file:

```
<ConfigurationSetsSection>
    <ConfigurationSettings>

     <add name="ClimIndices - Portable DB" default="true"
layers="Europe State Boundaries,Europe Grids"
queries="CGMS meteo grid maps Portable DB,CGMS meteo grid
graphs Portable DB,ClimIndices maps Portable
DB,ClimIndices graphs Portable DB" />
      <add name="Potential infection model - Portable DB"
default="true" layers="Europe State Boundaries,Europe
Grids" queries="CGMS meteo grid maps Portable DB,CGMS
meteo grid graphs Portable DB,Climpest maps Portable
DB,Climpest graphs Portable DB" />

    </ConfigurationSettings>
  </ConfigurationSetsSection>
```

How to edit the file

To add a workset, add a new element `name` to the list. In the following, the mandatory attributes are highlighted in red bold:

```
      <add name="Potential infection model - Portable DB"
default="true" layers="Europe State Boundaries,Europe Grids"
queries="CGMS meteo grid maps Portable DB,CGMS meteo grid
graphs Portable DB,Climpest maps Portable DB,Climpest
graphs Portable DB" />
```

Where:

- The attribute `name` identifies the workset

- The attribute `default` specifies if this is the default workset that the user will be displayed at the application startup (`true` or `false`)
- The attribute `layers` defines the shapefiles that will be displayed
- The attribute `queries` includes a list of names associated to the query files

Other files:

- “Defining the GIS Layers” on page 47
- “Defining the query files links” on page 49
- “Defining the database connections” on page 51
- “Defining the reports” on page 53
- “Defining the maps’ layouts” on page 55
- “Defining other settings” on page 58

Defining the GIS Layers

The `LayerSetsSection` list of the `mapper.config` file allows defining the layers that a workset can include. You can add a layer by adding a new element to the list.

The following shows the relevant section within the `mapper.config` file:

```
<LayerSetsSection>
    <LayerSettings>

      <add name="Europe State Boundaries" type="Info"
layerFile="NUTS0_RECODED1.shp" reference="true"
coordUnits="Meter" upperLeftX="1594018"
upperLeftY="5416162" lowerRightX="7687600"
lowerRightY="765000" />

      <add name="Europe Grids" type="Data"
layerFile="grid_25km_clip.shp" reference="false"
mappingColumn="Grid_Code" />

    </LayerSettings>
  </LayerSetsSection>
```

How to edit the file

To add a layer, add a new element `name` to the list. In the following, the mandatory attributes are highlighted in red bold:

```
     <add name="Europe State Boundaries" type="Info"
layerFile="NUTS0_RECODED1.shp" reference="true"
coordUnits="Meter" upperLeftX="1594018"
upperLeftY="5416162" lowerRightX="7687600"
lowerRightY="765000" />
```

Where:

- The attribute `name` is the logical name that can be associated to the `layers` attribute (name) as defined in the workset configuration (see “Defining the Workset” on page 45)

- The attribute `type` can be either a `Info` value, or a `Data` value field. A `Info` layer is always displayed in the MDV window, while a `Data` layer is only displayed after a user has selected a query and an indicator. A `Data` layer is used for mapping the resulting values. Each workset must only include one `Data` layer, but can include more `Info` layers.
- The attribute `layerFile` is the physical file associated to the selected shapefile.
- The attribute `reference` can either be `true` or `false`. This property can only be applied to one `Info` layer; it specifies the geographical area that will be used in the viewer. At least one `Info` layer must include this type of information. The geographical area can be further specified using the `coordUnits` attributes (`Meter` or `DecimalDegre`) and the `upperLeftX`, `upperLeftY`, `lowerRightX`, `lowerRightY` coordinates in the system that is defined by the `coordUnits` property.
- The attribute `mappingColumn` is only used for `Data` type layers. It defines the attributes of the `Data` layer's polygons (column of the `dbf` associated to the shapefile) that will be used as a key for mapping the values as these are extracted from the query.

Other files:

- "Defining the Workset" on page 45
- "Defining the query files links" on page 49
- "Defining the database connections" on page 51
- "Defining the reports" on page 53
- "Defining the maps' layouts" on page 55
- "Defining other settings" on page 58

Defining the query files links

The queries that are associated to the workset are defined in the `DataSetsSection`. Each query is configured as an element of this list.

The following shows the relevant section within the `mapper.config` file:

```
<DataSetsSection>
    <DataSets>

      <add name="CGMS meteo grid maps Portable DB"
dataSetGroup="maps"
      dataSetFile="./plugins/
DataQueryFrameworkConfigurationFiles/
CGMSGridMeteoMapQueries.xml" dataConnection="xxx"
      dataConnectionDef="MeteoCGMSPortatileConnection" />

      <add name="CGMS meteo grid graphs Portable DB"
dataSetGroup="graphs"
      dataSetFile="./plugins/
DataQueryFrameworkConfigurationFiles/
CGMSGridMeteoGraphQueries_sqlserver.xml"
dataConnection="xxx"
      dataConnectionDef="MeteoCGMSPortatileConnection" />

      <add name="Climpest maps Portable DB"
dataSetGroup="maps"
      dataSetFile="./plugins/
DataQueryFrameworkConfigurationFiles/
PotentialInfectionModelMapsQueries.xml"
dataConnection="xxx"
     dataConnectionDef="ClimPest-ClimIndices DBPortatile"
/>

    </DataSets>
  </DataSetsSection>
```

How to edit the file

To add a query, add a new element `name` to the list. In the following, the mandatory attributes are highlighted in red bold:

```
      <add name="CGMS meteo grid maps Portable DB"
      dataSetGroup="maps"
      dataSetFile="./plugins/
DataQueryFrameworkConfigurationFiles/
CGMSGridMeteoMapQueries.xml" dataConnection="xxx"
      dataConnectionDef="MeteoCGMSPortatileConnection" />
```

Where:

- The attribute `name` is the logical name that can be associated to the `queries` attribute (name) of a workset.
- The attribute `dataSetGroup` specifies if the indicators to which the associated query file refers will be displayed in maps or in graphics. Admitted values: `maps` or `graphs`.
- The attribute `dataSetFile` is the physical file where the queries are defined.
- The attribute `dataConnectionDef` is the logical name of the database connection that the queries must use.

Other files:

- “Defining the Workset” on page 45
- “Defining the GIS Layers” on page 47
- “Defining the database connections” on page 51
- “Defining the reports” on page 53
- “Defining the maps’ layouts” on page 55
- “Defining other settings” on page 58

Defining the database connections

The database connections associated to a dataset object are defined in the `connectionStrings` list. Each element of the list corresponds to a connection.

The following shows the relevant section within the `mapper.config` file:

```
 <connectionStrings>


   <add name="climpestdbmeteo" connectionString="Data
Source=CLIMPESTDB;User
ID=lars4read;Password=xxx;Unicode=True"
providerName="System.Data.OracleClient" />

   <add name="climindices" connectionString="Data
Source=CLIMPESTDB;User
ID=climindices;Password=xxx;Unicode=True"
providerName="System.Data.OracleClient" />

    <add name="climpestdb" connectionString="Data
Source=CLIMPESTDB;User
ID=climpest;Password=xxx;Unicode=True"
providerName="System.Data.OracleClient" />

    <add name="Meteo DBPortatile" connectionString="Data
Source=.\PortableDB\LarsGeneratedMeteoDatabase.sdf;Max
Database Size=4000"
providerName="System.Data.SqlServerCe.3.5" />

    <add name="ClimPest-ClimIndices DBPortatile"
connectionString="Data
Source=.\PortableDB\LocalDatabase1.sdf;Max Database
Size=4000"
providerName="System.Data.SqlServerCe.3.5" />


  </connectionStrings>
```

How to edit the file

To add a new database connection, add a new element `name` to the list. In the following, the mandatory attributes are highlighted in red bold:

```
    <add name="climpestdbmeteo" connectionString="Data
Source=CLIMPESTDB;User
ID=lars4read;Password=xxx;Unicode=True"
providerName="System.Data.OracleClient" />
```

Where:

- The attribute `name` is the logical name that can be associated to the `dataConnectionDef` attribute (name) of a `DataSet`.
- The attribute `connectionString` defines the parameters that will be used to connect to the database. The format of these parameters must comply with the `providerName` object.
- The attribute `providerName` is a third-party provider that MDV will use to connect to the database. Admitted values (providers) are: `System.Data.OracleClient, System.Data.SqlServerCe.3.5, System.Data.SqlClient, System.Data.OleDb`.

   Please note that the MDV setup does not install any provider. This means that the provider(s) that you want to use must be already installed in the machine that runs MDV.

Other files:

- “Defining the Workset” on page 45
- “Defining the GIS Layers” on page 47
- “Defining the query files links” on page 49
- “Defining the reports” on page 53
- “Defining the maps’ layouts” on page 55
- “Defining other settings” on page 58

Defining the reports

The XML reports that you create by using the  **Create Report** function of MDV (see “Creating a report from a map’s configuration” on page 23) must be defined in the `ReportSection`.

Each report is configured as an element of this list.

The following shows the relevant section within the `mapper.config` file:

```
<ReportSection>
    <ReportTemplates>

      <add name="PESETA: Irrigation number Maps"
workset="Europe at grid level (CGMS 08)"
reportTemplateFile="PESETAIrrigation.xml" />

     <<add name="PESETA: Difference Maps" workset="Europe
at grid level (CGMS 08)"
reportTemplateFile="PESETAAdaptationDiffer.xml" />

     <add name="PESETA: Yield gap Maps" workset="Europe at
grid level (CGMS 08)"
reportTemplateFile="PESETAAdaptationYieldGap.xml" />

     <add name="Report rain by country" workset="Europe at
grid level (CGMS 08)"
reportTemplateFile="RainByCountryReport.xml" />

    </ReportTemplates>
  </ReportSection>
```

How to edit the file

To add a report, add a new element `name` to the list. In the following, the mandatory attributes are highlighted in red bold:

```
     <add name="Report rain by country" workset="Europe at
grid level (CGMS 08)"
reportTemplateFile="RainByCountryReport.xml" />
```

Where:

- The attribute `name` is the name that you want to display in the list when the **Create Report** button is clicked in MDV.
- The attribute `workset` is the **Work Set** to which the report is linked to in MDV. (That is, when the **Create Report** button is clicked, the report will be displayed if this **Work Set** has been selected).
- The attribute `reportTemplateFile` is the physical XML file that you created.

Other files:

- "Defining the Workset" on page 45
- "Defining the GIS Layers" on page 47
- "Defining the query files links" on page 49
- "Defining the database connections" on page 51
- "Defining the maps' layouts" on page 55
- "Defining other settings" on page 58

Defining the maps' layouts

To define the graphic layout of a map, users can take advantage of the graphical user interface that is provided when clicking the **Set Map Layout** button in MDV. (See "Changing the Map Visualizer's map layout" on page 19).

However, advanced users can edit the XML templates that define the layouts. (See "Editing the XML layout template" on page 56).

To make the templates available when a user clicks the **Export Map** function of MDV (see "Exporting the map" on page 22), these must be defined in the `LayoutTemplatesSection` of the `mapper.config` file.

Each template is configured as an element of this list.

The following shows the relevant section within the `mapper.config` file:

```
 <LayoutTemplatesSection>
    <LayoutTemplates>

      <add name="Jrc grid meteo template for pdf"
type="map" exporttype="pdf"
file=".\JRCMapGridPdfTemplate.xml" />

      <add name="Jrc grid meteo template for png"
type="map" exporttype="png"
file=".\JRCMapGridPngTemplate.xml" />

      <add name="Jrc graph template" type="graph"
exporttype="png" file=".\JRCMapTemplate.xml" />

    </LayoutTemplates>
  </LayoutTemplatesSection>
```

How to edit the file

To add a template, add a new element `name` to the list. In the following, the mandatory attributes are highlighted in red bold:

```
      <add name="Jrc grid meteo template for pdf"
type="map" exporttype="pdf"
file=".\JRCMapGridPdfTemplate.xml" />
```

Where:

- The attribute `name` is the name of the map layout template that you want to display in the list when the **Export Map** button is clicked in MDV.
- The attribute `type` is the type of element you are entering
- The attribute `exporttype` defines the format of the export file
- The attribute `file` is the physical XML template that defines the layout's look and that you can edit as it is described here below.

Editing the XML layout template

The following shows the `JRCMapGridPdfTemplate.xml` configuration file. It is a template that is provided with MDV and that you can edit to further customize your layout.

```
<?xml version="1.0" encoding="utf-8" ?>
<adornment name="JRC grid meteo" resolution="Medium">


  <element name="maptitle" type="label" value="title"
posx="240" posy="150" font="ARIAL"
fontsize="24" fontstyle="Bold" fontcolorr="0" fontcolorg="0"
fontcolorb="0"/>


  <element name="mapsubtitle" type="label"
value="subtitle" posx="240" posy="170" font="ARIAL"
fontsize="12" fontstyle="Bold" fontcolorr="0"
fontcolorg="0" fontcolorb="0"/>


  <element name="logo" type="image"
value=".\logo_ec4mapper_long.png" posx="705" posy="235" />


 <element name="legend" type="legend" value="legendtitle"
posx="240" posy="200"
showNoData="false" legsquaredimension="8" font="ARIAL"
fontsize="8" fontstyle="Regular"
fontcolorr="0" fontcolorg="0" fontcolorb="0"/>


 <!--  <element name="copyright" type="label" value=" "
posx="280" posy="615" font="Microsoft
Sans Serif" fontsize="8" fontstyle="Regular"
fontcolorr="0" fontcolorg="0" fontcolorb="0"/>-->
</adornment>
```

Where:

- The attribute `adornment` is the name of your choice for the layout
- The attribute `resolution` defines the image resolution. Available options are `High` - `Medium` - `Low`.
- The `element` `name` allows defining other attributes to be displayed in the map. This allows further customizing the settings that you define via the MDV graphical user interface ( **Set Map Layout** button). For example: `maptitle, mapsubtitle, logo, legend, copyright`.
- The attribute `type` defines the type of element you are displaying within the map. This can be one of the following:
  - `label`
  - `image`
  - `legend`
- The attribute `value` allows specifying the content that will be displayed in the map layout for the selected type of element: `title`, `subtitle`, name of the image (e.g., `logo_ec4mapper_long.png`), or a free text string.
- The attributes `posx` and `posy` respectively define the horizontal and vertical physical positions of the element in the map layout.
- The attribute `font` , as well as the other font-related attributes (such as, `fontsize`, `fontsyle`, etc.) defines the text formatting.

Other files:

- “Defining the Workset” on page 45
- “Defining the GIS Layers” on page 47
- “Defining the query files links” on page 49
- “Defining the database connections” on page 51
- “Defining the reports” on page 53
- “Defining other settings” on page 58

Defining other settings

The following shows other settings that can be defined in the `mapper.config` file.

The editable sections are highlighted in red bold:

```
<appSettings>
     <add key="shapeFileDir" value=".\shapeDir" />
     <add key="logoFile"
value=".\shapeDir\logoMarsIPSCJRC.png" />
</appSettings>


 <MappingSetsSection>
    <MappingSettings>
      <add mapField="Grid_Code"
graphRelatedField="grid,grid_no,grid
no,id_weather_grid,Grid no" />
      <add mapField="REG_CODE" graphRelatedField="Nuts
code" />
      <add mapField="NURGCDL2" graphRelatedField="Nuts
code" />
    </MappingSettings>
  </MappingSetsSection>
```

Where:

- The section `appSettings` defines the working directory where you want to save the files that can be associated to the layers. To connect a file to a layer, you can use the item with key `shapeFileDir`; to specify the logo file that will be displayed on the maps, you can use the item with key `logoFile`.
- The section `MappingSetsSection` defines the rules that will be used to connect the elements to be displayed in a map with the elements to be displayed in a graph when using the  (**Show Graphs for the items selected**) function. The elements that are defined in this section state that, if the displayed data layer includes a `mapField` field, or if `graphRelatedField` includes a query's input field of `Graph` type, when using the  button, the association between the query's input field and the element that is displayed in the map is not editable.

Other files:

- “Defining the Workset” on page 45
- “Defining the GIS Layers” on page 47
- “Defining the query files links” on page 49
- “Defining the database connections” on page 51
- “Defining the reports” on page 53
- “Defining the maps’ layouts” on page 55

Defining the query files

This section describes the two versions of the definition's syntax of the queries used by MDV.

In this section:

Related topics:

- "Mapper.Config file" on page 44

Query file definition - Version 2

The Version 2 of the definition's syntax of the query used by MDV, consists in the definition tag of the queries list.

```
<?xml version="1.0" encoding="utf-8" ?>
<Queries  xmlns="urn:data_query_framework_schema"
connectionstring="[connection]" provider="[provider]">
<DataQueryFrameworkEngine version="2" />


   <Query name="Grid grapevine - Climate - Tonietto"
xAxisUnit="" comparator="" value="select
[operator]([indicator])  from tonietto_index_vitis t ">
  <QueryIndicators>
     <QueryIndicator name="Tonietto classification" yAxisUnit=""
sql="(classification)"/>


     </QueryIndicators>
     <Key  value="id_weather_grid" datatype="number"/>
     <Filters>
        <Filter name="weather_series" type="string"
sql="weather_series =[weather_series]" query="select 'baseline'
from dual union select 'hadcm3_a1b' from dual union select
'had_a1b50' from dual union select 'hadcm3_b1' from dual union
select 'had_b150' from dual "  />
        <Filter name="year" type="number"  sql="year =[year]" />
        <Filter name="variety" type="string"  sql="variety
=[variety]" query="select 'Cabernet Sauvignon' from dual union
select 'Syrah' from dual union select 'Chardonnay' from dual"  /
>
       <Filter name="model_id" type="number"  sql="model_id
=[model_id]" query="select '2' from dual union select '3' from
dual" />
     </Filters>
     <Operators>
       <Operator name="value" value="" />
     </Operators>
     <Groups>


     </Groups>
   </Query>


 </Queries>
```

Description and mandatory elements (Version 2)

The upper section is always mandatory, where:

```
<Queries  xmlns="urn:data_query_framework_schema"
connectionstring="[connection]" provider="[provider]">
```

It consists in the definition tag for the queries list, which includes the parser declaration for reading and executing the query file:

```
<DataQueryFrameworkEngine version="2" />
```

A list of the query elements that specify which datasets can be retrieved and displayed via the mapper is also mandatory, as shown below:

```
<Query name="meteo data - rainfall queries – cumulated
indicators" xAxisUnit="day" comparator="" value="select
[operator]([indicator]) cgms_08_eur.grid_weather ">
    <QueryIndicators>
      <QueryIndicator name="rain" sql="rainfall" yAxisUnit="mm/
m^2"/>
      <QueryIndicator name="rain1" sql="rainfall" yAxisUnit="mm/
m^2"/>
    </QueryIndicators>
    <Key value="day" datatype="datetime"/>
    <Filters>
      <Filter name="end day" type="datetime"   sql="day &lt;
[end day]"/>
      <Filter name="start day" type="datetime"   sql="day &gt;
[start day]" />
      <Filter name="grid" type="string"  query="select
distinct(grid_no) from cgms_08_eur.grid" sql="grid_no=[grid]"/>
    </Filters>
    <Operators>
      <Operator name="cumulated" value="sum" />
      <Operator name="max" value="max" />
      <Operator name="min" value="min" />
    </Operators>
```

```
    <Groups>
      <Group name="year" xAxisUnit="year" value="to_char(day,
'yyyy')" datatype="number"/>
      <Group name="grid"  value="grid_weather.grid_no"
datatype="string"/>
      <Group name="month" xAxisUnit="month" value="to_char(day,
'MM')" datatype="number" />
    </Groups>
   </Query>
```

See the following topics for a description of the XML tags that compose the query:

Query object

```
<Query name="meteo data - rainfall queries – cumulated
indicators" xAxisUnit="day" comparator="" value="select
[operator]([indicator]) cgms_08_eur.grid_weather ">
```

Each **Query** type object includes the following attributes:

- **name** - It allows to identify the query within the software’s dropdown menu. The query’s name can be separated by an hyphen punctuation mark (-), which is then used in the dropdown menu of the mapper: for this purpse, a tree structure is used whose root is the left- most portion of the name (preceding the hyphen), while the child is the right-most (following the hyphen).
- **xAxisUnit** - It defines the measure unit of the X axis if the query is being used withing a graph.
- **comparator** - Not used.
- **value** - It corresponds to the select and form structure of the query to be implemented. This structure includes two placeholders: `[operator]` and `[indicator]`, which will be replaced when actually building the query. Both tags are mandatory and, mostly, will be populated as empty fields. The `value` object includes the initial

skeleton of the parser-managed query; the `where` and the `group` portions will be automatically added to this skeleton via the XML tags that will be subsequently defined.

QueryIndicator tag

```
    <QueryIndicators>
      <QueryIndicator name="rain" sql="rainfall" yAxisUnit="mm/
m^2"/>
```

The **<QueryIndicators>** tag includes the list of the indicators from which the user will select the actual indicator to be shown in the query. This will replace the `[indicator]` placeholder (described previously).

Each **<QueryIndicator>** element includes the following attributes:

- **name** - Indicator's logical name that will be shown by the application in the query's composition window.
- **xAxisUnit** - It defines the indicator's measure unit.
- **sql** - SQL fragment that wil be automatically replaced by the parser in the `[indicator]` placeholder.

Key tag

```
  <Key value="day" datatype="datetime"/>
```

The **<Key>** tag is an additional field that the query extracts in the `select` portion. This tag will be used as a value for the X axis, in case of either a graph or a spatial object identifier.

It includes the following attributes:

- **value** - Query column to be used as a key.
- **datatype** - Datatype of the key column (admitted values: `number`, `string`, `datetime`).

Filters tag

```
<Filters>
    <Filter name="end day" type="datetime"   sql="day &lt; [end
day]"/>
    <Filter name="grid" type="string"  query="select
distinct(grid_no) from cgms_08_eur.grid" sql="grid_no=[grid]"/>
```

The **<Filters>** tag of the parser-generated query, contains the list of the filters. Each `<Filter>` element defines a filter for the generated query and it has an **and** automatic connection with other elements.

Each **<Filter>** element includes the following attributes:

- **name** - Logical name of the filter that is displayed in the query's composition window of the application.
- **type** - Datatype of the user-entered filter (admitted values: `number`, `string`, `datetime`)
- **sql** - SQL fragment that will be automatically added to the `where` portion of the parser-generated query. Normally, this SQL fragment normally consists in an expression to which is assigned the value of the filter (**[weather_series]**) and a **weather_series = [weather_series]** column.
- **Query** - If the list of filter values must be limited, and if these values are to be extracted from the database somehow, it is possible to define a free query that the parser will invoke in order to manage the field population.

Operators tag

```
<Operators>
      <Operator name="cumulated" value="sum" />
```

If the query to be run must include an aggregation operator, you can use the **<Operators>** tag to define the relevant list.

Each **<Operator>** element includes the following attributes:

- **name** - Logical name of the aggregation operator that will be shown by the application in the query's composition window.
- **value** - SQL fragment that will be replaced in the `[operator]` placeholder.

Groups tag

```
<Groups>
      <Group name="year" xAxisUnit="year" value="to_char(day,
'yyyy')" datatype="number"/>
```

The **<Groups>** tag allows defining the criteria based upon which to execute the aggregation so enabling the **<Operator>** defined above. In this case, the new `group by` operator redefines the **Key** tag shown above (see "Key tag" on page 65).

Each **<Group>** element includes the following attributes:

- **name** - Logical name of the aggregation criteria that will be show by the application in the query's composition window.
- **xAxisUnit** - Measure unit that redefines the one specified in the **<Query>** tag because of the aggregation operation (see "Query object" on page 64, xAxisUnit attribute).

- **value** - SQL fragment that will include the portion to be added to the `group by` attribute.
- **datatype** - Datatype of the new key (`group by` operator) that you defined in order to aggregate the records (admitted values: `number`, `string`, `datetime`).

Related topics:

- “Query file definition - Version 3” on page 68
- “Mapper.Config file” on page 44

Query file definition - Version 3

The Version 3 of the definition's syntax of the query used by MDV, consists in the definition tag of the queries list.

```
<?xml version="1.0" encoding="utf-8" ?>
<Queries  xmlns="urn:data_query_framework_schema"
connectionstring="[connection]" provider="[provider]">
<DataQueryFrameworkEngine version="3" />
   <Query name="Cgms grid meteo - indicator - value"
xAxisUnit="" comparator="" title="[operator] for [indicator]"
subtitle="from [Start day] to [End day]">
     <QueryIndicators>
       <QueryIndicator name="Average temperature" yAxisUnit="Dec
Degrees" sql="round([operator]((gy.maximum_temperature +
gy.minimum_temperature)/2),3)"/>
       <QueryIndicator name="Rain" yAxisUnit="mm of rain"
sql="round([operator](gy.rainfall),3)"/>
       <QueryIndicator name="Vapour pressure" yAxisUnit="hPa"
sql="round([operator](gy.vapour_pressure),3)"/>
       <QueryIndicator name="Wind speed" yAxisUnit="m/s"
sql="round([operator](gy.windspeed),3)"/>
       <QueryIndicator name="E0" yAxisUnit="mm/d"
sql="round([operator](gy.E0),3)"/>
       <QueryIndicator name="ES0" yAxisUnit="mm/d"
sql="round([operator](gy.ES0),3)"/>
       <QueryIndicator name="ET0" yAxisUnit="mm/d"
sql="round([operator](gy.ET0),3)"/>
       <QueryIndicator name="Calculated radiation"
yAxisUnit="Kj/(d*m*m)"
sql="round([operator](gy.calculated_radiation),3)"/>
       <QueryIndicator name="Snow depth" yAxisUnit="cm"
sql="round([operator](gy.snow_depth),3)"/>
     </QueryIndicators>
     <Key  value="gy.grid_no" datatype="number"/>
     <Tables>
       <Table name="gy" sql="grid_weather gy"/>
       <!--<Table name="lc" sql="cgms_08_forall.crop_landcover
lc"/>-->
     </Tables>
     <Filters>
       <Filter name="join" type="string"  sql="gy.grid_no in
(select distinct grid_no from nuts_grids ng where
lower(ng.nuts_code_05) in (select lower(column_value) from
table(split( [Nuts] ))))" hidden="true"/>       <Filter
```

```
name="Start day" type="datetime"  sql="gy.day &gt;= [Start day]"
/>
       <Filter name="End day" type="datetime"  sql="gy.day &lt;=
[End day]" />
       <!--<Filter name="Crop" type="string" query="select
lc.crop_name from cgms_08_forall.crop_landcover lc
                   where lc.landcover_id is not null
                   order by
                   c.crop_name" sql="lc.crop_name = [Crop]"/>-->
       <Filter name="Nuts" type="string"  sql="" />
     </Filters>
     <Operators>
       <Operator name="Max Value" value="max" />
       <Operator name="Min Value" value="min" />
       <Operator name="Avg Value" value="avg" />
       <Operator name="Cum Value" value="sum" />
     </Operators>
     <Groups>
       <Group name="Grid number" xAxisUnit="" value="gy.grid_no"
datatype="number" hidden="true"/>
     </Groups>
   </Query>

 </Queries>
```

See also:

- Description and mandatory elements (Version 3)

Description and mandatory elements (Version 3)

The upper section is always mandatory:

```
<Queries  xmlns="urn:data_query_framework_schema"
connectionstring="[connection]" provider="[provider]">
```

It consists in the definition tag for the queries list, which includes the parser declaration for reading and executing the query file:

```
<DataQueryFrameworkEngine version="3" />
```

A list of the query elements that specify which datasets can be retrieved and displayed via the mapper is also mandatory, as shown below:

```
<Query name="Cgms grid meteo - indicator - value" xAxisUnit=""
comparator="" title="[operator] for [indicator]" subtitle="from
[Start day] to [End day]">
     <QueryIndicators>
       <QueryIndicator name="Average temperature" yAxisUnit="Dec
Degrees" sql="round([operator]((gy.maximum_temperature +
gy.minimum_temperature)/2),3)"/>
       <QueryIndicator name="Rain" yAxisUnit="mm of rain"
sql="round([operator](gy.rainfall),3)"/>
       <QueryIndicator name="Vapour pressure" yAxisUnit="hPa"
sql="round([operator](gy.vapour_pressure),3)"/>
       <QueryIndicator name="Wind speed" yAxisUnit="m/s"
sql="round([operator](gy.windspeed),3)"/>
       <QueryIndicator name="E0" yAxisUnit="mm/d"
sql="round([operator](gy.E0),3)"/>
       <QueryIndicator name="ES0" yAxisUnit="mm/d"
sql="round([operator](gy.ES0),3)"/>
       <QueryIndicator name="ET0" yAxisUnit="mm/d"
sql="round([operator](gy.ET0),3)"/>
       <QueryIndicator name="Calculated radiation"
yAxisUnit="Kj/(d*m*m)"
sql="round([operator](gy.calculated_radiation),3)"/>
       <QueryIndicator name="Snow depth" yAxisUnit="cm"
sql="round([operator](gy.snow_depth),3)"/>
     </QueryIndicators>
     <Key  value="gy.grid_no" datatype="number"/>
     <Tables>
       <Table name="gy" sql="grid_weather gy"/>
       </Tables>
     <Filters>
       <Filter name="join" type="string"  sql="gy.grid_no in
(select distinct grid_no from nuts_grids ng where
```

```
lower(ng.nuts_code_05) in (select lower(column_value) from
table(split( [Nuts] ))))" hidden="true"/>
       <Filter name="Start day" type="datetime"  sql="gy.day
&gt;= [Start day]" />
       <Filter name="End day" type="datetime"  sql="gy.day &lt;=
[End day]" />
       <Filter name="Nuts" type="string"  sql="" />
     </Filters>
     <Operators>
       <Operator name="Max Value" value="max" />
       <Operator name="Min Value" value="min" />
       <Operator name="Avg Value" value="avg" />
       <Operator name="Cum Value" value="sum" />
     </Operators>
     <Groups>
       <Group name="Grid number" xAxisUnit="" value="gy.grid_no"
datatype="number" hidden="true"/>
     </Groups>
   </Query>
```

See the following topics for a description of the XML tags:

- “Query object” on page 71
- “QueryIndicator tag” on page 72
- “Key tag” on page 72
- “Tables tag” on page 72
- “Filters tag” on page 73
- “Operators tag” on page 73
- “Groups tag” on page 74

Query object

```
<Query name="Cgms grid meteo - indicator - value" xAxisUnit=""
comparator="" title="[operator] for [indicator]" subtitle="from
[Start day] to [End day]">
```

Each **Query** type object includes the following attributes:

- **name** - It allows to identify the query within the software’s dropdown menu. The query’s name can be separated by an hyphen punctuation mark (-), which is then used in the dropdown menu of the mapper: for this purpse, a tree structure is used whose root is the left- most portion of the name (preceding the hyphen), while the child is the right-most (following the hyphen).

- **xAxisUnit** - It defines the measure unit of the X axis if the query is being used withing a graph.
- **comparator** - Not used.
- **title** - The title that will be generated by the parser for the created query. The tag can include placeholders that will be replaced by the user-selected values during the query generation.
- **subtitle** - The subtitle that will be generated by the parser for the created query. The tag can include placeholders that will be replaced by the user-selected values during the query generation.

QueryIndicator tag

```
<QueryIndicators>
  <QueryIndicator name="Average temperature" yAxisUnit="Dec
Degrees" sql="round([operator]((gy.maximum_temperature +
gy.minimum_temperature)/2),3)"/>
```

The **<QueryIndicators>** tag includes the list of the indicators from which the user will select the actual indicator to be shown in the query.

Each **<QueryIndicator>** element includes the following attributes:

- **name** - Indicator's logical name that will be shown by the application in the query's composition window.
- **xAxisUnit** - It defines the indicator's measure unit.
- **sql** - SQL fragment that wil be used by the parser to compose the query. The fragment can include `[operator]` type placeholders, which will be replaced by the parser when generating the query.

Key tag

```
<Key  value="gy.grid_no" datatype="number"/>
```

The **Key** tag is an additional field that the query extracts in the `select` portion. This tag will be used as a value for the X axis, in case of either a graph or a spatial object identifier.

Each **Key** element includes the following attributes:

- **value** - Query column to be used as a key.
- **datatype** - Datatype of the key column (admitted values: `number`, `string`, `datetime`).

Tables tag

```
<Tables>
  <Table name="gy" sql="grid_weather gy"/>
  </Tables>
```

The **<Tables>** tag defines the list of the tables that will be interrogated by the generated query.

Each **<Table>** element includes the following attributes:

- **name** - Logical name of the table to be used in the query.
- **sql** - SQL fragment that identifes the physical table to be used in the query.

Filters tag

```
<Filters>
      <Filter name="join" type="string"  sql="gy.grid_no in
(select distinct grid_no from nuts_grids ng where
lower(ng.nuts_code_05) in (select lower(column_value) from
table(split( [Nuts] ))))" hidden="true"/>
```

The **<Filters>** tag of the parser-generated query, contains the list of the filters. Each <Filter> element defines a filter for the generated query and it has an **and** automatic connection with other elements.

Each **<Filter>** element includes the following attributes:

- **name** - Logical name of the filter that is displayed in the query's composition window of the application.
- **type** - Datatype of the user-entered filter (admitted values: number, string, datetime)
- **sql** - SQL fragment that will be automatically added to the where portion of the parser-generated query. Normally, this SQL fragment normally consists in an expression to which is assigned the value of the filter (**[end day]**) and a **gy.day <= [End day]** column.
- **query** - If the list of filter values must be limited, and if these values are to be extracted from the database somehow, it is possible to define a free query that the parser will invoke in order to manage the field population.
- **hidden** - It can be true or false. It defines if the filter will be hidden to the user or not. Normally, if hidden, the filter will be used for the join operation (not user-configurable).

Operators tag

```
<Operators>
  <Operator name="Max Value" value="max" />
```

If the query to be run must include an aggregation operator, you can use the **<Operators>** tag to define the relevant list.

Each **<Operator>** element includes the following attributes:

- **name** - Logical name of the aggregation operator that will be shown by the application in the query's composition window.
- **value** - SQL fragment that will be replaced in the `[operator]` placeholder within **QueryIndicator** (if any).

Groups tag

```
<Groups>
      <Group name="year" xAxisUnit="year" value="to_char(day,
'yyyy')" datatype="number"/>
```

The **<Groups>** tag allows defining the criteria based upon which to execute the aggregation so enabling the **<Operator>** defined above. In this case, the new `group by` operator redefines the **Key** tag shown above (see "Key tag" on page 72).

Each **<Group>** element includes the following attributes:

- **name** - Logical name of the aggregation criteria that will be show by the application in the query's composition window.
- **xAxisUnit** - Measure unit that redefines the one specified in the **<Query>** tag because of the aggregation operation (see "Query object" on page 71, xAxisUnit attribute).
- **value** - SQL fragment that will include the portion to be added to the `group by` attribute.
- **datatype** - Datatype of the new key (`group by` operator) that you defined in order to aggregate the records (admitted values: `number`, `string`, `datetime`).
- **hidden** - It can be `true` or `false` and it defines if the field will be hidden to the user or not.

Related topics:

- "Query file definition - Version 2" on page 61
- "Mapper.Config file" on page 44