Mango Javascript API User Guide

API Version 1.0

Infinite Automation Systems - December 8, 2014

Overview

This document provides an overview of how to create dashboards using Point Values from Mango Automation. An understanding of Javascript is expected.

The system is designed to aid in the creation of HTML/Javascript pages to display data from a Mango Automation instance.

The API has been designed with the following model in mind.

- Input Configurations control the data that is returned from Mango
- Data Providers manipulate the returned data
- Display Configurations display the manipulated data.

The normal workflow for a page is:

1. A user chooses data points and the date ranges and rollups on a page
2. Data Providers are actioned to retrieve and optionally manipulate the data
3. Data Providers use their linked Display Configurations to display the data

Data Points can be selected manually via the Mango REST API or by use of the Grouping and Matching Javascript API. See below for more information on this.

Accessing Pages

Private Dashboards

Private dashboards require users to be logged into Mango Automation. If a user that is not logged in attempts to access a private dashboard they are presented with a login page. Private pages must be post-fixed with the .shtm extension.

The module's web/private folder is mapped to the Mango URL:
http://{host}:{port}/private-dashboards

So for example a page located in web/private/demo/page.shtm will have the url:
http://{host}:{port}/private-dashboards/demo/page.shtm

Public Dashboards

Public dashboards are accessible without a Mango Login. Public pages should be postfixed with the .html extension.

The module's web/public folder is mapped to the Mango URL:
http://{host}:{port}/public-dashboards

So for example a page located in web/public/demo/page.shtm will have the url:
http://{host}:{port}public-dashboards/demo/page.shtm

Javascript Regex

Javascript regex is similar to standard Regex but not exactly the same. Regex patterns are defined within slashes /pattern/.

Match Mango exactly: /Mango/

Match anything that starts with Mango: /Mango.*/

Match anything that ends with Mango: /.*Mango/

More information can be found here:
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_Expressions

Data Objects

This section lists the structure of some of the common objects used in the API.

Point Value Time

To get an empty Point Value Time Object one can call mangoApi.pointValues.createNew().

```
{
        annotation: String,
        dataType: 'ALPHANUMERIC', 'BINARY', 'MULTISTATE', or 'NUMERIC',
        value: String, Boolean, or Numeric,
        timestamp: Numeric ms since Epoch (Date().getTime())
}
```

Numeric Point Statistics

```
{
        minimum: PointValueTime,
        maximum: PointValueTime,
        average: Number,
        integral: Number,
        sum: Number,
        first: PointValueTime,
        last: PointValueTime,
        count: Number
}
```

Discrete Point Statistics

```
{
        startsAndRuntimes: Array of StartAndRuntime
        first: PointValueTime,
        last: PointValueTime
}
```

Supplemental StartAndRuntime Object

```
{
        runtime: Number,
        proportion: Number,
        starts: Number
}
```

Data Providers

Data providers are used to retrieve data for a page. All data providers have the same basic functionality but differ in the types of data that is transported.

Common Members

The following is a list of the members that are common to all data providers.

Data Provider ID

Each provider has an Integer ID that is used to reference it across the page. This is useful when linking a Display to a data provider.

Point Configurations

Each provider has a list of Point Configurations. These define what points are used in its operations. For example a data provider with 2 points will always load the values for both of its points when its load() method is called. It will also always store the value in both points when the put() method is called.

Listeners

Each provider has a list of listeners that are notified of new data when the load() method is called. A listener can be any javascript object with both onLoad(data, point) and onClear() methods but is usually a Display component.

Common Methods

The following is a list methods common to all data providers.

load(options , error)

The load method takes 2 parameters:

options - Any javascript object that varies depending on the data provider type
error - A specific error method that will be called if something goes wrong

When load is called the data is retrieved from Mango, then the manipulateData() method is for the data provided is called, then the data is passed to each listener via its onLoad() method.

clear(clearConfigurations)

The clear method takes 1 parameter:

clearConfigurations - boolean to indicate the provider should clear out its Point Configurations

When clear is called the data provider optionally clears all of its point configurations and then notifies all Listeners via their onClear() method.

manipulateData(data, point)

This method can optionally be provided to any Data Provider to allow manipulation of the retrieved data.

Parameters:

data - Array of Point Value Time objects retrieved by the data provider
point - Point object that relates to the data

Returns:

The data that will be sent to the Listeners via their onLoad method.

An example use for this would be to use a Data Provider to compute the total for a group of point values it has retrieved.

put(options, error)

Method used to store data into Mango.

Parameters:

options - javascript object with boolean member refresh to indicate a refresh of the listeners with the value that was put. ie. options = {refresh: true}

error - A specific error method that will be called if something goes wrong

addListener(listener)

Add a listener to the Data Provider. As mentioned previously a Listener has 2 methods:

onLoad(data, point)
onClear()

addDataPoint(dataPointConfiguration)

Add a data point configuration to the Data Provider.

Point Value Data Provider

A point value data provider loads Point Value Time Objects for date ranges and allows aggregation of the returned data.

Data Type Provided

PointValueTime Array

load(options , error)

```
options = {
        from: Date,
        to: Date,
        rollup: One of ‘AVERAGE’, ‘FIRST’, ‘LAST’, ‘MINIMUM’, ‘MAXIMUM’, ‘SUM’, ‘COUNT’,
        timePeriodType: ‘YEARS’, ‘MONTHS’, ‘WEEKS’, ‘DAYS’, ‘HOURS’, ‘MINUTES’,
        timePeriods: Number
}
```

The rollup, timePeriodType and timePeriods can be set to null if no aggregation is desired.

Historical Data Provider

Historical data providers are used to collect an Array of Point Value Time Objects via the number of historical samples. For example one can return the latest 100 points using this data provider.

Data Type Provided

PointValueTime Array

load(options, error)

```
options = {
        historicalSamples - Number
}
```

Realtime Point Value Data Provider

This data provider is used to receive updates whenever a Point is changed. There is an additional member for this data provider:

loadMostRecent - Boolean to indicate whether to load the most recent sample before we register for point updates. This is useful if the point is infrequently updated and data needs to be displayed on the page.

Data Type Provided

PointValueTime Array

load(options, error)

options = {}

Statistics Data Provider

This data provider is used to retrieve statistics.

Data Type Provided

Statistics Object

load(options, error)

```
options = {
        from: Date,
        to: Date,
}
```

Display Components

Consider each component to be a building block for the API. Each component type can be used alone or in conjunction with other components. Each component is defined in its own file but contains 2 parts.

1. Component Configuration - This is the definition that is used on the page

2. Component Implementation - This is the code that is used to display the data and will not normally be seen by the person building the page.

Bar Chart Configuration

A Bar Chart displays bar graphs. All point data for the chart must be contained within one array. The default settings for a Bar Chart compute the total for each data point supplied by the linked Data Providers and display each as a bar.

Example Graph Data

Each bar is contained within one item in the resulting Array. For example, to plot 1 bar on a chart, the array would look like:

```
[{
        label: 'Data Set 1'
        bar1: Number,
}]
```

To plot multiple bars on the chart for each series the Array would look like:

```
[{
        label: 'Data Set 1'
        bar1: Number,
        bar2: Number
}]
```

To plot multiple sets of bars on the chart for each series the Array would look like:

```
[{
        label: 'Data Set 1'
        bar1: Number,
        bar2: Number
},{
        label: 'Data Set 2'
        bar1: Number,
        bar2: Number
}]
```

Usage

Construct a Bar Chart Configuration with the constructor:

```
new BarChartConfiguration('divId', [dataProviderIds], AmChartJSON,
mangoChartMixins, options)
```

Parameter	**Type**	**Description**
divId	String	Id of div to place chart
dataProviderIds	Array of Integer	Ids for data providers used on chart

amChartJson	Object	JSON Chart configuration
mangoChartMixins	Object	Extensions for the created Display Component.
options	Object	Any additional options for the Configuration.

AmChart JSON

This is a simple example to display one bar for each data point. Using the dataPoint.name member for the category label and the chart data item's 'value' member for the data.

```
{
        "categoryField": "name",
        "graphs": [
                {
                  "balloonText": "Average:[[value]]",
                  "fillAlphas": 0.8,
                  "id": "AmGraph-1",
                  "lineAlpha": 0.2,
                  "title": "Average",
                  "type": "column",
                  "valueField": "value"
                }
            ],
            "valueAxes": [
                {
                  "id": "ValueAxis-1",
                  "position": "top",
                  "axisAlpha": 0
                }
            ],
   }
```

Mango Chart Mixins

The chart mixins are commonly used to override the onLoad method for this display. They can also be used to override anything in the resulting MangoBarChart Object.

For example:

```
{
     onLoad: function( data, dataPoint){
          //Perform new custom operation here
     }
}
```

See the example pages for more details on how this can be achieved.

Options

The options are most commonly used to map the data from a provider into the chart data array. They can also be used to override anything in the BarChartConfiguration Object.

One example would be where the amChartJson contains a graph with valueField: ‘series1’ and a data point with name TestPoint.

One Item in the array would then be:

```
{
        series1: Number,
        timestamp: Number
}
```

The corresponding mapping in the options is:

```
{
        dataPointMappings: [{
                nameEndsWith: ‘TestPoint’,
                valueField: ‘series1’
        }]
}
```

The available mapping options are:

- nameEndsWith
- nameStartsWith
- xidEndsWith
- xidStartsWith

If no mappings are used, all point values are stored to the ‘value’ member in the member of the array:

```
{
        value: Number,
        timestamp: Number
}
```

Javascript File

The file containing all code for this component is at web/mango/v1/barChart.js

Gauge Chart Configuration

A Gauge Chart displays data in a gauge. All point data for the chart is contained within one element in the data array.

Example Graph Data

For example, to show 17.1 on the gauge, the array would look like:

```
[{
        value: 17.1
}]
```

Usage

Construct a Gauge Chart Configuration with the constructor:

```
new GaugeChartConfiguration('divId', [dataProviderIds], AmChartJSON, mangoChartMixins, options)
```

Parameter	**Type**	**Description**
divId	String	Id of div to place chart
dataProviderIds	Array of Integer	Ids for data providers used on chart
amChartJson	Object	JSON Chart configuration
mangoChartMixins	Object	Extensions for the created Display Component.
options	Object	Any additional options for the Configuration.

AmChart JSON

By normally this can be left empty: {}

Mango Chart Mixins

The mangoChartMixins are normally used to override the valueAttribute and the onLoad method but can be used to override anything in the resulting MangoGaugeChart Object.

```
{
        valueAttribute: 'value',
        onLoad: function( data, dataPoint){
                this.amChart.arrows[0].setValue(data[0][this.valueAttribute]);
                this.amChart.axes[0].setBottomText(this.renderValue(data[0]));
                this.amChart.validateData();
        }
}
```

The value attribute represents the member of the incoming data that will be used to display the value in the gauge, the default is 'value'.

onLoad can also be overridden here in the case that a special operation needs to be performed on the incoming data.

Options

The options are not normally required for this display component but can be used to override anything in the GaugeChartConfiguration Object.

Javascript File

The file containing all code for this component is at web/mango/v1/gaugeChart.js

Pie Chart Configuration

A Pie Chart Configuration displays pie graphs. All point data for the chart must be contained within one array. The default settings for a Pie Chart compute the total for each data point supplied by the linked Data Providers and display each as a slice.

Example Graph Data

Each slice is contained within one item in the resulting Array. For example, to plot 1 slice on a chart, the array would look like:

```
[{
        name: 'Data Set 1'
        total: Number,
}]
```

To plot multiple slices on the chart for each series the Array would look like:

```
[{
        name: 'Data Set 1'
        total: 1
},{
        name: 'Data Set 2'
        total: 2
}]
```

Usage

Construct a Pie Chart Configuration with the constructor:

```
new PieChartConfiguration('divId', [dataProviderIds], AmChartJSON,
mangoChartMixins, options)
```

Parameter	Type	Description
divId	String	Id of div to place chart
dataProviderIds	Array of Integer	Ids for data providers used on chart
amChartJson	Object	JSON Chart configuration
mangoChartMixins	Object	Extensions for the created Display Component.}
options	Object	Any additional options for the Configuration.

AmChart JSON

This is a simple example to display a slice with the label from DataPoint.xid and the value from chart data.average

```
{
            valueField: "average",
            titleField: "xid",
            titles: [{
                text: "Averages",
                size: 16
            }],
            depth3D: 15,
            angle: 30
}
```

Mango Chart Mixins

The chart mixins are commonly used to override the onLoad method for this display but can be used to override anything in the resulting MangoPieChart Object

For example:

```
{
        onLoad: function( data, dataPoint){
                //Perform new custom operation here
        }
}
```

See the example pages for more details on how this can be achieved.

Options

The options are not normally required for this display component but can be used to override anything in the PieChartConfiguration Object.

Javascript File

The file containing all code for this component is at web/mango/v1/pieChart.js

Serial Chart Configuration

A Serial Chart displays line graphs. All point data for the chart must be contained within one array. This means that each series must contain a separate identifier within each array item.

Example Graph Data

For example, to plot 2 series on a chart each item in the array must contain:

```
{
        series1: Number,
        series2: Number,
        time: Date,
}
```

Usage

Construct a Serial Chart Configuration with the constructor:

```
new SerialChartConfiguration(‘divId’, [dataProviderIds], AmChartJSON,
mangoChartMixins, options)
```

Parameter	**Type**	**Description**

divId	String	Id of div to place chart
dataProviderIds	Array of Integer	Ids for data providers used on chart
amChartJson	Object	JSON Chart configuration
mangoChartMixins	Object	Extensions for the created Display Component.
options	Object	Any additional options for the Configuration.

AmChart JSON

This example will display one graph that uses the chart data item’s ‘temperature’ member for the y value and the chart data item’s ‘timestamp’ value for the x value.

```
{
        titles: [{
           id: "Title-1",
           size: 15,
           text: "Degress F "
        }],
        categoryField: "timestamp",
        graphs: [{
           title: "Temperature",
           valueAxis: "temp-axis",
           bullet: "square",
           bulletSize: 6,
           lineColor: "green",
           lineThickness: 1,
           negativeLineColor: "red",
           type: "smoothedLine",
           valueField: "temperature"
        }],
        valueAxes: [{
           id: "temp-axis",
           title: " Degress F",
           position: "left"
        }],
}
```

Mango Chart Mixins

The chart mixins are commonly used to override the onLoad method for this display. They can also be used to override anything in the resulting MangoSerialChart Object.

For example:

```
{
    onLoad: function( data, dataPoint){
        //Perform new custom operation here
    }
}
```

See the example pages for more details on how this can be achieved.

Options

The options are most commonly used to map the data from a provider into the chart data array. They can also be used to override anything in the SerialChartConfiguration Object.

One example would be where the amChartJson contains a graph with valueField: ‘series1’ and a data point with name TestPoint.
One Item in the array would then be:

```
{
    series1: Number,
    timestamp: Number
}
```

The corresponding mapping in the options is:

```
{
    dataPointMappings: [{
        nameEndsWith: ‘TestPoint’,
        valueField: ‘series1’
    }]
}
```

The available mapping options are:

- nameEndsWith
- nameStartsWith
- xidEndsWith
- xidStartsWith

Javascript File

The file containing all code for this component is at web/mango/v1/serialChart.js

Simple Display Configuration

A Simple Display Configuration displays nothing by default but can be used as a template to build new display components. For example a simple display component could be created to update a div with new information anything a data provider is refreshed.

Usage

Construct a Simple Display Configuration with the constructor:

```
new SimpleDisplayConfiguration([dataProviderIds], mangoDisplayMixins,
options)
```

Parameter	Type	Description
dataProviderIds	Array of Integer	Ids for data providers used on chart
mangoDisplayMixins	Object	Extensions for the created Display Component.
options	Object	Any additional options for the Configuration.

Mango Display Mixins

The display mixins must be used to override the onLoad and onClear methods for this display. For example:

```
{
        onClear: function(){
                //Perform clearing of component here
        },
        onLoad: function( data, dataPoint){
                //Perform display of data here
        }
}
```

Options

The options are not normally used for this component but can be used to override anything in the SimpleDisplayConfiguration Object.

Javascript File

The file containing all code for this component is at web/mango/v1/simpleDisplay.js

Starts and Runtime Statistics Configuration

A Starts and Runtime Statistics displays statistics for data providers of non-numeric type points (Multistate, Binary and Alphanumeric).

Example Graph Data

```
{
        startsAndRuntimes: Array of Starts and Runtime Statistics
        first: Number,
        last: Number,
}
```

The data displayed is a StartsAndRuntimeStatistics Object.

Usage

The component works by expecting 3 HTML elements prefixed with the divPrefix string used in the configuration. Not all 3 elements are required and data will only be displayed in the provided elements. Any HTML element that can contain text can be used (i.e. div or span).

For example if we set the divPrefix = ‘Stats’ then:

Starts and Runtime Table:
<table id=”StatsStartsAndRuntimes”></table>

First Point in the set
<div id=”StatsFirst”></div>

Last Point in the statitics:
<div id=”StatsLast”></div>

Construct a Starts and Runtime List Statistics Configuration with the constructor:

```
new StartsAndRuntimeListConfiguration(‘divPrefix’, [dataProviderIds],
mangoMixins, options)
```

Parameter	**Type**	**Description**
divPrefix	String	Prefix for all required Divs
dataProviderIds	Array of Integer	Ids for data providers used on chart
mangoMixins	Object	Extensions for the created Display Component.
options	Object	Any additional options for the Configuration.

Mango Mixins

The chart mixins are commonly used to override the onLoad method for this display. They can also be used to override anything in the resulting MangoStartsAndRuntimeList Object.

For example:

```
{
        onClear: function(){
                //Custom clear code here
        },
        onLoad: function( data, dataPoint){
                //Perform new custom operation here
        },
        renderPointValueTime: function(pvt){
                //Specific rendering of a Point Value Time Object
        },
        renderValue: function(number){
                //Specific renerding of a number
        },
        renderTime: function(timestamp){
                //Specific rendering of a timestamp into a Date String
        }
}
```

See the example pages for more details on how this can be achieved.

Options

The options are not normally required for this component but can be used to override anything in the StartsAndRuntimeListConfiguration.

Javascript File

The file containing all code for this component is at web/mango/v1/startsAndRuntimeListStatistics.js

Statistics Configuration

A Statistics displays statistics for data providers of Numeric type points.

Example Graph Data

```
{
        minimum: Number,
        maximum: Number,
        average: Number,
        integral: Number,
        sum: Number,
        first: Number,
        last: Number,
        count: Number
}
```

The data displayed is a Statistics Object.

Usage

The component works by expecting 8 HTML elements prefixed with the divPrefix string used in the configuration. Not all 8 elements are required and data will only be displayed in the provided elements. Any HTML element that can contain text can be used (i.e. div or span).

For example if we set the divPrefix = ‘Stats’ then:

Minimum Point
<span id=”StatsMin”></span>

Maximum Point
<span id=”StatsMax”></span>

Average Value
<div id=”StatsAverage”></div>

Integral Value
<div id=”StatsIntegral”></div>

Sum Value
<div id=”StatsSum”></div>

Count Value
<div id=”StatsCount”></div>

First Point in the set
<div id=”StatsFirst”></div>

Last Point in the statitics:
<div id=”StatsLast”></div>

Construct a Statistics Configuration with the constructor:

```
new StatiticsConfiguration(‘divPrefix’, [dataProviderIds],
mangoMixins, options)
```

Parameter	**Type**	**Description**
divPrefix	String	Prefix for all required Divs
dataProviderIds	Array of Integer	Ids for data providers used on chart
mangoMixins	Object	Extensions for the created Display Component.
options	Object	Any additional options for the Configuration.

Mango Mixins

The chart mixins are commonly used to override the onLoad method for this display. They can also be used to override anything in the resulting MangoStatistics Object.

For example:

```
{
	onClear: function(){
		//Custom clear code here
	},
	onLoad: function( data, dataPoint){
		//Perform new custom operation here
	},
	renderPointValueTime: function(pvt){
		//Specific rendering of a Point Value Time Object
	},
	renderValue: function(number){
		//Specific renerding of a number
```

```
        },
        renderTime: function(timestamp){
                //Specific rendering of a timestamp into a Date String
        }
}
```
See the example pages for more details on how this can be achieved.

Options
The options are not normally required for this component but can be used to override anything in the StatisticsConfiguration.

Javascript File
The file containing all code for this component is at web/mango/v1/statistics.js

Statistics Bar Chart Configuration
A Bar Chart displays bar graphs using Statistics Objects.

Each resulting item in the chart data array represents one group of bars. By defining the graphs in the amChartJson one can display some or all of the available statistics for a point.

Example Graph Data
```
{
        minimum: Number,
        maximum: Number,
        average: Number,
        integral: Number,
        sum: Number,
        first: Number,
        last: Number,
        count: Number
}
```

Usage
Construct a Statistics Bar Chart Configuration with the constructor:

```
new BarChartConfiguration('divId', [dataProviderIds], AmChartJSON,
mangoChartMixins, options)
```

Parameter	Type	Description
divId	String	Id of div to place chart
dataProviderIds	Array of Integer	Ids for data providers used on chart
amChartJson	Object	JSON Chart configuration
mangoChartMixins	Object	Extensions for the created

		Display Component.
options	Object	Any additional options for the Configuration.

AmChart JSON

This is a simple example to display one bar for the minimum value for each data point.

```
{
        categoryField: "name",
             graphs: [
                    {
                       "balloonText": "Minimum:[[value]]",
                       "fillAlphas": 0.8,
                       "id": "AmGraph-1",
                       "lineAlpha": 0.2,
                       "title": "Minimum",
                       "type": "column",
                       "valueField": "minimum"
                    }],
        valueAxes: [
                    {
                       "id": "ValueAxis-1",
                       "position": "top",
                       "axisAlpha": 0
                    }
                 ],
}
```

Mango Chart Mixins

The chart mixins are commonly used to override the onLoad method for this display. They can also be used to override anything in the resulting MangoBarChart Object.

For example:

```
{
        onLoad: function( data, dataPoint){
                //Perform new custom operation here
        }
}
```

See the example pages for more details on how this can be achieved.

Options

Normally there are no options for this component but they can be used to override anything in the StatisticsBarChartConfiguration Object.

Javascript File

The file containing all code for this component is at web/mango/v1/statisticsBarChart.js\

Input Configurations

Input configurations are used to create HTML input widgets. Each input component is defined in its own file but contains 2 parts.

1. Input Component Configuration - This is the definition that is used on the page
2. Input Component Implementation - This is the code that is used to display the widget and collect the data from it.

DateTimePicker Configuration

A DateTimePicker configuration is used to generate a widget that collects a date and time from the User.

Usage

Create an HTML **<input>** type Element on the page with the desired id.

Construct a DateTimePickerConfiguration with the constructor:

```
new DateTimePickerConfiguration(id, mangoInputMixins, options)
```

Parameter	Type	Description
id	String	Id for HTML Element to use
mangoInputMixin	Object	Extensions for the created Input Component.
options	Object	Any additional options for the Configuration.

Mango Input Mixins

The input mixins are not normally used and can be left empty: {}.

Options

The display mixins must be used to override the onChange method. They can also be used to set a default value. For example:

```
{
        defaultValue: new Date(),
        onChange: function( date, picker, owner){
                //Do something with the selected date here
        }
}
```

Javascript File

The file containing all code for this component is at web/mango/v1/dateTimePicker.js

Input Configuration

A Input configuration is used to generate a widget that collects data from any <input> element.

Usage

Create an HTML **<input>** type Element on the page with the desired id.

Construct a InputConfiguration with the constructor:
`new InputConfiguration(id, mangoInputMixins, options)`

Parameter	**Type**	**Description**
id	String	Id for HTML Element to use
mangoInputMixin	Object	Extensions for the created Input Component.
options	Object	Any additional options for the Configuration.

Mango Input Mixins

The input mixins are not normally used and can be left empty: {}.

Options

The display mixins must be used to override the onChange method. They can also be used to set a default value. For example:

```
{
        defaultValue: “default”,
        onChange: function( value,  owner ){
                //Do something with the changed value
        }
}
```

Javascript File

The file containing all code for this component is at web/mango/v1/input.js

List View Configuration

A List View Configuration is used to generate a widget that monitors the <li> from within an <ul> HTML element set for clicks and changes values accordingly.

Usage

Create an HTML **<ul>** type Element on the page with the desired id.

Construct a InputConfiguration with the constructor:
`new ListViewConfiguration(id, mangoInputMixins, options)`

Parameter	**Type**	**Description**
id	String	Id for HTML Element to use
mangoInputMixin	Object	Extensions for the created Input Component.

options	Object	Any additional options for the Configuration.

Once the object is created it is possible to add additional <li> elements by using the listViewConfiguration.addItem(label, id) method.

Mango Input Mixins

The input mixins are not normally used and can be left empty: {}.

Options

The display mixins must be used to override the onChange method. They can also be used to set a style for the generated <li> elements. For example:

```
{
        styleClass: ‘my-css-li-class’,
        options: [
                {label: ‘item 1’, value: 1},
                {label: ‘item 2’, value: 2},
        ],
        onChange: function( value,  owner ){
                //Do something with the changed value
        }
}
```

Javascript File

The file containing all code for this component is at web/mango/v1/listView.js

Rollup Configuration

A Rollup Configuration is used to generate a drop down list with Rollup options.

Usage

Create an HTML **<select>** type Element on the page with the desired id.

Construct a InputConfiguration with the constructor:

```
new RollupConfiguration(id, mangoInputMixins, options)
```

Parameter	**Type**	**Description**
id	String	Id for HTML Element to use
mangoInputMixin	Object	Extensions for the created Input Component.
options	Object	Any additional options for the Configuration.

Once the object is created it is possible to add additional <option> elements by using the rollupConfiguration.addItem(label, id, selected) method. By default all possible rollups are included in the list and are not required to be added in the options.

Mango Input Mixins
The input mixins are not normally used and can be left empty: {}.

Options
The display mixins must be used to override the onChange method. They can also be used add options and choose the currently selected option. For example:

```
{
        selected: 0, //Index of item to select
        placeholder: 'text for unselected input',
        options: [
                {label: 'AVERAGE', value: 'AVERAGE'},
                {label: 'MAXIMUM', value: 'MAXIMUM'},
                {label: 'MINIMUM', value: 'MINIMUM'},
                {label: 'SUM', value: 'SUM'},
                {label: 'FIRST', value: 'FIRST'},
                {label: 'LAST', value: 'LAST'},
                {label: 'COUNT', value: 'COUNT'}
         ],

        onChange: function( value,  owner ){
                //Do something with the changed value
        }
}
```

Javascript File
The file containing all code for this component is at web/mango/v1/rollupPicker.js

Select Configuration
A Select Configuration is used to generate a drop down list with custom options.

Usage
Create an HTML **<select>** type Element on the page with the desired id.

Construct a InputConfiguration with the constructor:
`new SelectConfiguration(id, mangoInputMixins, options)`

Parameter	**Type**	**Description**
id	String	Id for HTML Element to use
mangoInputMixin	Object	Extensions for the created Input Component.
options	Object	Any additional options for the Configuration.

Once the object is created it is possible to add additional <option> elements by using the selectConfiguration.addItem(label, id, selected) method.

Mango Input Mixins

The input mixins are not normally used and can be left empty: {}.

Options

The display mixins must be used to override the onChange method. They can also be used add options and choose the currently selected option. For example:

```
{
        selected: 0, //Index of item to select
        placeholder: 'text for unselected input',
        options: [
                {label: 'One', value: '1'} ,
                {label: 'Two', value: '2'} ,
        ],

        onChange: function( value,  owner ){
                //Do something with the changed value
        }
}
```

Javascript File

The file containing all code for this component is at web/mango/v1/select.js

Select Picker Configuration

A Select Picker Configuration is used to generate a drop down list with custom options from a JQuery Select Picker widget.

Usage

Create an HTML **<select>** type Element on the page with the desired id.

Construct a InputConfiguration with the constructor:

`new SelectPickerConfiguration(id, mangoInputMixins, options)`

Parameter	**Type**	**Description**
id	String	Id for HTML Element to use
mangoInputMixin	Object	Extensions for the created Input Component.
options	Object	Any additional options for the Configuration.

Once the object is created it is possible to add additional <option> elements by using the selectPickerConfiguration.addItem(label, id, selected) method.

Mango Input Mixins
The input mixins are not normally used and can be left empty: {}.

Options
The display mixins must be used to override the onChange method. They can also be used add options and choose the currently selected option. For example:

```
{
        selected: 0, //Index of item to select
        placeholder: ‘text for unselected input’,
        options: [
                {label: ‘One’, value: ’1’} ,
                {label: ‘Two’, value: ’2’} ,
        ],
        onChange: function( value,  owner ){
                //Do something with the changed value
        }
}
```

Javascript File
The file containing all code for this component is at web/mango/v1/selectPicker.js

Time Period Type Picker Configuration
A Time Period Type Picker Configuration is used to generate a drop down list with Time Period Type Options.

Usage
Create an HTML **<select>** type Element on the page with the desired id.

Construct a InputConfiguration with the constructor:
`new TimePeriodTypePickerConfiguration(id, mangoInputMixins, options)`

Parameter	**Type**	**Description**
id	String	Id for HTML Element to use
mangoInputMixin	Object	Extensions for the created Input Component.
options	Object	Any additional options for the Configuration.

Once the object is created it is possible to add additional <option> elements by using the timePeriodTypePickerConfiguration.addItem(label, id, selected) method. By default all available Time Period Types are included in the drop down.

Mango Input Mixins
The input mixins are not normally used and can be left empty: {}.

Options

The display mixins must be used to override the onChange method. They can also be used add options and choose the currently selected option. For example:

```
{
        selected: 0, //Index of item to select
        placeholder: ‘text for unselected input’,
        options: [
                        {label: 'MINUTES', value: 'MINUTES'},
                        {label: 'HOURS', value: 'HOURS'},
                        {label: 'DAYS', value: 'DAYS'},
                        {label: 'WEEKS', value: 'WEEKS'},
                        {label: 'MONTHS', value: 'MONTHS'},
                        {label: 'YEARS', value: 'YEARS'}
            ], //Not using yet 'MILLISECONDS', 'SECONDS',

        onChange: function( value,  owner ){
                //Do something with the changed value
        }
}
```

Javascript File

The file containing all code for this component is at web/mango/v1/timePeriodTypePicker.js

Dashboard Templater

The highest level component is the Dashboard Templater. Using this component simplifies the amount of code required on a page at the expense of some flexibility.

The Templater has added functionality that allows templating of pages to re-use components when changing between groups of similar data points. For example if Mango is monitoring 3 Machines that all have a Current and Voltage you can switch between displaying the data for a given machine on the same Display Components.

The Templater is the control center that will monitor all Input Configurations and automatically refresh the Display Configurations.

Usage

The templater is able to create inputs on the page to control the refreshing of Display Components. If the following HTML Elements are provided the appropriate inputs will be created.

HTML Element	ID	Description
<select>	customPeriodSelect	Creates a drop down list where the user can select pre-defined dates such as Today, 7 Days and 30 Days

<input>	startDate	Date Selection widget for start date
<input>	endDate	Date Selection widget for end date
<select>	rollup	Select drop down for Rollups
<select>	timePeriodType	Select drop down for Time Period Type
<input>	timePeriod	Input for time periods
<select>	groups	Select drop down for matched groups

In addition to providing the HTML element definitions on the page it is required to setup a templater configuration consisting of Display Configurations, Point Match Configurations, Group Configurations and optionally DataProviders. This configuration is used in the creation of the DashboardTemplater Object.

```
new DashboardTemplater(configuration);
```

Configuration

The following table is a list of **optional** configuration elements to provide the templater.

Member	Type	Description
debug	boolean	Show debug messages in browser's Javascript console
displayConfigurations	Array of DisplayConfiguration	List of all display component configurations for the templater
pointConfigurations	Array of DataPointMatchConfiguration	List of all data point match configurations the templater should use
groupConfiguration	Array of DataPointGroupConfiguration	List of all group match configurations the templater should use
groups	Array of DataPointGroup	Normally used internally to hold all matched groups but can be manually set if the templater is not going to be actioned to perform a group
endDate	Date	Defaults to now and is used

		during a refresh()
startDate	Date	Defaults to 12hrs ago and is used during a refresh()
rollup	String	Defaults to ‘AVERAGE’ and is used during a refresh()
timePeriodType	String	Defaults to ‘HOURS’ and is used during a refresh()
timePeriods	Integer	Defaults to 1 and is the number of TimePeriodTypes used in a refresh()
historicalSamples	Integer	Defaults to 10 and is used for Historical Data providers in a refresh()
customPeriodConfiguration	SelectConfiguration	*See below
startDateConfiguration	DateTimePickerConfiguration	User input to collect the start date for a refresh()
endDateConfiguration	DateTimePickerConfiguration	User input to collect the end date for a refresh()
rollupConfiguration	RollupConfiguration	User input for the Rollup used in a refresh()
timePeriodTypeConfiguration	TimePeriodTypeConfiguration	User input for time period type used in a refresh()
timePeriodConfiguration	InputConfiguration	User input for time periods used in a refresh()
groupSelectConfiguration	SelectConfiguration	User input to choose a group for the refresh()
displayManager	DataDisplayManager	Controls display configuration mappings
pointMatcher	DataPointMatcher	Controls matching of data points from groups
groupMatcher	PointHierarchyGrouper	Controls grouping of data points
providersToRefresh	Array of Integer	set to null to refresh all data providers during a refresh() otherwise set to array of desired data provider IDs to refresh.

The customPeriodConfiguration can be used to pre-define regularly selected time ranges. The default is to use Today, 7 Days, 30 Days and This Year. This input provides a quick way for a user to select date ranges.

Methods

Method Name	Description
customPeriodChanged	called when the custom period input is changed
startDateChanged	called when the startDate input is changed
endDateChanged	called when the endDate input is changed
rollupChanged	called when the rollup input is changed
timePeriodTypeChanged	called when the time period type input is changed
timePeriodChanged	called when the time period input is changed
onMatch	called when a Data Point Match is made
onGroup	called when a Group is matched
groupChanged	called when the group input is changed
refresh(providerIds, templater)	called with a list of provider IDs and the templater to use and will refresh all display configurations linked to the provided Data Provider IDs. If providerIds is null all data providers within the templater are refreshed
put(ids, value)	Have one or many data providers put a value into Mango. ids - Array of Data Provider IDs value - JSON { value: PointValueTimeModel, refresh: boolean to refresh displays with this data}
invalidateChartSize(chartDivIds, templater)	Helper for AmCharts display to force a resize
showError(jqXHR, textStatus, errorThrown, mangoMessage)	Used for all errors to display error in a div with id “errors”

Javascript File

The file containing all code for this component is at web/mango/v1/dashboardTemplater.js

Data Point Group Configuration

Data Point Group Configurations are used to create Groups of data points from the Point Hierarchy. These Groups are then used in the Group Select widget.

Usage

Create a new Group Configuration with the code:
`new DataPointGroupConfiguration(config)`

Configuration

The configuration is a javascript object with the following properties and options.

groupBy
A String of ‘Folder’, DataPoint or ‘All’ that controls how the matching is applied.

If set to All then no matching is done and all data points are included in the resulting group.
If set to ‘Folder’ then the match configurations are applied to the Folder of the data point in the Point Hierarchy.
If set to ‘DataPoint’ then the match configurations are applied to the Data Points directly.

label
String that can be used instead of a labelAttribute to statically define the label for a group of points.

labelAttribute
The attribute from the object that will be used to create the label. If groupBy is set to:

‘Folder’ - then labelAttribute can be ‘name’, ‘id’ or ‘path’,
‘DataPoint’ - then labelAttribute can be ‘xid’, ‘deviceName’ ‘pointFolderId’ or ‘dataSourceXid’

matchConfigurations
An Array of MatchConfiguration objects. The regex member is optional but if provided will be matched against the contents of the matchAttribute member. The available matchAttribute values depend on the groupBy value.

If groupBy is set to:

‘Folder’ then matchAttribute can be ‘path’ to match on the Folder path of a point or ‘name’ or ‘id’ of the Folder for the Data Point.

‘DataPoint’ then matchAttribute - can be ‘xid’, ‘deviceName’, ‘pointFolderId’, or ‘dataSourceXid’

When groupBy = ‘All’ there is no need for any match configurations.

Example JSON

```
config = {
        groupBy: 'Folder',
        labelAttribute: 'name',
        matchConfigurations: [{
           matchAttribute: 'path',
           regex: /Target Folder/
        }]
 }
```

Javascript File

The file containing all code for this component is at web/mango/v1/dataPointGroup.js

Data Point Match Configuration

Data Point Match Configurations are used control how DataPoints within DataPointGroups are linked to Data Providers.

Usage

Create a new Match Configuration with the code:
`new DataPointMatchConfiguration(providerId, matchConfigurations, options)`

Provider ID

The provider ID references the ID of an existing or yet-to-be-created Data Provider. If the Data Provider already exists in the Templater then it is used for this Data Point otherwise a new Data Provider is created.

Match Configurations

The match configurations is Array of PointMatchConfiguration Objects.

matchAttribute
Can be 'xid', 'deviceName', 'pointFolderId', or 'dataSourceXid'

regex
Javascript regex that will be applied to the match attribute member of the data point to ensure a match.

Options

The options are used to override anything in the DataPointMatchConfiguration Object and are generally used to override the providerType property. For example setting the options parameter to { providerType: 'Statistics' } would ensure when the data provider for this match is created it is a StatisticsDataProvider. Available options are:

- 'Statistics'
- 'PointValue'
- 'RealTime'
- 'Historical'

Javascript File

The file containing all code for this component is at web/mango/v1/dataPointMatcher.js

Appendix

Swagger

To view the Mango Rest API output and explore the options point your browser to:
http://{host}:{port}/swagger/index.html

You must enable swagger before you start Mango Automation to view these pages. See the env.properties file for the settings to enable Swagger.

Web Sockets

The realtime updates for the Mango Javascript API use Web Sockets. See here for more information:
https://developer.mozilla.org/en-US/docs/WebSockets