MultiValue Dashboard

Programmer's Guide

MultiValue Dashboard Programmer's Guide Overview

MultiValue Dashboard is a MultiValue CGI application that delivers the output of individual "widgets" to the dashboard web page. Multiple dashboards may be defined, and each dashboard becomes a unique collection of widgets.

A MultiValue software developer can use MV Dashboard to produce graphical representations of data in the MultiValue system while leveraging his or her strengths as a BASIC programmer.

The following topics are presented:

Widget Programming Environment	Details the widget programming environment.
Widget Types	Describes and lists widget types.
Integration With Other Applications	Describes how to display dashboard widgets via URL calls from other applications.
Appendix I: Widget Output Type Codes	Details the widget output type codes for color and theme management.
Appendix II: Optional Parameters	Details the optional parameters for the widget types.
Appendix III: PDF Generation	Instructions for enabling widgets to generate PDF files.
Fusion Charts Documentation	Provides links to the Fusion Charts documentation.

Widget Programming Environment

Developing dashboard widgets requires the creation of a BASIC program to provide the following information:

1. The type of widget to be displayed, e.g., a chart, map, table, preformatted text, or HTML.
2. The widget title that will appear in the title bar of the widget that is displayed.
3. The width of the widget (i.e. 1=Narrow, 2=Medium, or 3=Wide)
4. The data used to formulate the chart or map, table, the raw HTML, or the preformatted text depending on the type of widget being created.

Additionally, the program may provide information to invoke other behaviors of the widget such as links, icons, or input prompts to appear in the widget display.

The following topics are presented:

Technical Description

The dashboard controller program, MVDB.MAIN, is the mainline program responsible for generating the dashboard HTML pages. This program calls each widget's BASIC subroutine in the appropriate sequence as specified by the active dashboard's configuration.

The MVDB.SUBS file is designed to contain widget programs. All widget program source code must begin as follows:

```
1. SUBROUTINE widget_subroutine_name
2. INCLUDE WBPD MVDB.INCLUDE
3. W$TYPE = "FCPIE" <- Or desired widget type
4. W$TITLE = "Widget Title" <- Or desired widget title
5. W$WIDTH = 1 <- Or desired widget width
6. IF G$QUERYMODE = 1 THEN RETURN
7. (the rest of your code to follow)
```

The dashboard controller program initializes a named common space that is available to each of the widget subroutines. The named common space is declared in WBPD MVDB.INCLUDE as follows:

COMMON /MVDB/ GLOBAL.INFO(100), GLOBAL.USER.DATA(20), GLOBAL.TEMP.DATA(20), WIDGET.INFO(100), WIDGET.USER.DATA(100)

GLOBAL.INFO(1...100) contains global variables that are assigned and used by the dashboard controller programs. Widget subroutines are not expected to modify the information in the global variable space. However, these variables can be useful to widget programs. EQUATES for the variables in GLOBAL.INFO(1...100) make the following variable names available:

G$USERID	The ID of the user currently logged in
G$CURRENT.DB	Current dashboard name
G$LOGIN.DATE	Date user logged in (from session)
G$LOGIN.TIME	Time user logged in (from session)
G$WIDGET.NAME	Name of the widget being executed
G$WIDGET.SUB	Name of the widget subroutine being called
The following parameters are corresponding, value-mark separated lists:	
G$WIDGET.NAME.LIST	List of all widget names on this dashboard in the order that they are to be executed.
G$WIDGET.SUB.LIST	List of subroutine names associated with each of the widgets.
G$WIDGET.CALL.TIME	Time/Date stamp that each widget was called. This value will be blank for all widgets that have not yet been executed.

G$WIDGET.CALL.DUR	Duration (in milliseconds) of each widget subroutine. This value will be blank for all widgets that have not yet been executed.
G$QUERYMODE	Set to "1" if used to query subroutines without running them. The administration module uses this flag to determine the type and width of each widget without requiring the entire widget program to run.
G$SUBMITTED	Set to "1" if the user clicked the submit button on this widget by use of the W$INPUT... parameters.
G$REMOTE.ADDR	IP Address of the user's computer on the network.
G$SERVER.NAME	Server name as referenced by the user's browser.
G$HTTP.PORT	HTTP Port in use (e.g. 81)
G$DRILLDOWN.MODE	Set to "1" if the widget is being executed in drill-down mode.
G$DD.REFERRER	In Drilldown mode, the name of the widget calling the drilldown.
G$DD.FROM.DB	In Drilldown mode, the name of the dashboard calling the drilldown.

Each widget subroutine may use these variables to determine (for example) who the user is, what widgets have been called before this widget and what widgets will be called after this widget.

GLOBAL.USER.DATA(1...20) contains 20 variables that are shared by all widget programs. The values of these variables are persistent with the user's session. As long as the user's session is valid, these variables will remain intact. Widget programs must be developed to coordinate the use of these global variable positions as all widget programs will be able to read and write values in this array.

GLOBAL.TEMP.DATA(1...20) contains 20 variables that are shared by all widget programs. The values of these variables are initialized each time the dashboard is displayed. Unlike GLOBAL.USER.DATA(20), these variables are not persistent with the user's session. Widget programs may share the information in this array, but widgets will only have access to values set by programs executed earlier in the sequence of subroutine calls.

WIDGET.USER.DATA(1...100) contains 100 variables that are visible only to the active widget program. The private variables remain persistent with the user's session. The dashboard controller program does not modify these variables except as directed by the widget program through link information or user input (described below). When the user logs out, or closes his or her browser window, these values are destroyed.

WIDGET.INFO(1...100) holds all information generated by a widget subroutine for the purpose of generating the visible display. The structure of this array is defined in WBPD MVDB.INCLUDE. EQUATES for the variables in WIDGET.INFO(1...100) make the following variable names available:

General Widget Definition

W$TYPE	Type of widget output (See Appendix for a complete list)
W$TITLE	Title text to appear on the widget title bar
W$WIDTH	1 = 1/3 column (Narrow), 2 = 2/3 column (Medium), 3 = Full Width (Wide)

The widget width does not determine the width of screen provided to this widget. It tells the dashboard designer how wide this widget would like to be. For pie charts and bar graphs, this will be used to determine the size of the column and the size of the chart inside the column. The dashboard controller will insert a widget of any size into any column, regardless of how well the widget will fit. A horizontal scroll bar will appear at the bottom of the widget if it is too wide to fit in the column.

Defining Links in a Widget

All widget programs can use the variables below to add links to the widget. When any of these links is clicked, the dashboard controller will re-run all of the widgets on the dashboard and re-display the page. However, the links can tell the dashboard controller to populate specific elements in WIDGET.USER.DATA(1...100) with specific information prior to executing this specific widget subroutine.

The link definition also includes the location(s) at which the links are to be displayed within the widget. Links may be displayed in any of the four corners of the widget window, and they may also be displayed as icons (without text) in the right-hand side of the widget's title bar. As an example, you may want to create a widget that shows sales information for a single month with a link in the upper-left-hand corner called "Previous Month," and a link in the upper-right-hand corner called "Next Month."

Adding a link to a widget requires the following variables to be assigned, and each variable may contain a value-mark-delimited list if multiple links are desired:

W$LINK.LABEL	The link text
W$LINK.LOCATION	The location(s) as quadrants 1-4, and/or "C" to display an icon in the corner of the widget. To display a single link in multiple locations, simply concatenate the locations into a single string. For example, "14C" will display the link in the upper-right-hand corner, the lower-right-hand corner, and as an icon in the title bar.
W$LINK.UD.POS	The WIDGET.USER.DATA array element number to store the user data value. This may include a sub-value-delimited list to populate multiple values in WIDGET.USER.DATA for a single link.
W$LINK.UD.VAL	The value to be stored in prescribed WIDGET.USER.DATA element. This may include a sub-value-delimited list to populate multiple values in WIDGET.USER.DATA for a single link.
W$LINK.ICON	This value may include the name of an icon image (in the "icons" directory of the "docs" folder) when using the "C" (corner) location.
W$LINK.ICON.TXT	When displaying a link as an icon in the title bar ("C") of the widget, this text will display when the user holds the mouse pointer over the icon.

Drill Down Widgets

Any widget can create a drill-down link to any other widget. If you want a link to load a widget other than the one in which the links are displayed, you may populate the "drill-down" widget information in the variables below. When a drill-down widget is invoked, it is displayed as a single, full-width widget on the page. No other widgets will be displayed along with a drill-down widget.

When using a link to invoke a drill-down widget, the values specified in W$LINK.UD.POS and W$LINK.UD.VAL will be applied to the WIDGET.USER.DATA variables for the drill-down widget, not the source (referring) widget.

W$LINK.DD.WIDGET	Name of the drill-down widget to load when the link is clicked.

The global variable G$DRILLDOWN.MODE is set to "1" when a widget is being executed as a drill-down widget. This allows a widget to determine whether or not it is being executed as a drill-down widget. Additionally, the global variables G$DD.REFERRER and G$DD.FROM.DB will be set to the name of the widget and dashboard from which the drill-down occurred.

External Links

This advanced feature allows you to configure the widget links as external links. When using external links, no information is fed back to the widget itself, and the widget producing the external link will not be notified that the link was clicked. Adding an external link is done by assigning the following variables, and by doing so the values in W$LINK.UD.POS, W$LINK.UD.VAL and W$LINK.DD.WIDGET are ignored.

W$LINK.URL	The link HREF value.
W$LINK.URL.OPTS	If provided, this string will be inserted into the <a> tag. For example, assigning this value with "target='_blank'" will cause the link to open in a new window.

NOTE on HTML type widgets: The W$LINK.DD and W$LINK.URL functionality is not available on an HTML type widget since the HTML must be fully formed and there is no formatting applied to it. If drilldowns are desired from an HTML type widget, the HTML must contain <a href> attributes defining the destinations and any parameters.

In the HTML widget, to have the word Sales highlighted so that clicking on it would drill down to the DISPLAY REV.GP SALES widget with a parameter of "REVENUES" you would replace the word Sales with this <a href> string:

Sales

When using an <a href> you are linking to a dashboard, not a widget, so it is necessary to create a dashboard with a one to one relationship with the desired widget.

User Input

This feature is very similar to adding links to a widget; however, unlike the links feature, this allows the user the ability to select or specify values.

There are five types of user input available:

1. The user may fill in a text field. (W$INPUT.TYPE = "TEXT")
2. The user may fill in a password field (W$INPUT.TYPE = "PASSWORD"), which behaves the same as text field input except that the typed characters are obfuscated by dots.
3. The user may select a value from a list of predetermined values. (W$INPUT.TYPE = "SELECT")
4. The user may select a date from a date picker pop-up: (W$INPUT.TYPE = "DATE")
5. The user may select or deselect a check box (W$INPUT.TYPE = "CHECKBOX") which sets WIDGET.USER.DATA() to "1" if selected or "0" if deselected. Note that the initial value will be "" and will not become "0" unless and until it has been deselected after being previously selected, and therefore should be evaluated as being equal to "1" or not equal to "1".

Each of the values listed below may contain multiple attributes in order to generate multiple input prompts. When the submit button is clicked, all input data values will be assigned to the prescribed WIDGET.USER.DATA element(s) prior to running the widget on the next page load. Input prompts are defined by populating the following variables:

When the user submits the form (clicks the "Go" button), the global variable G$SUBMITTED will be set to "1" when the widget subroutine is executed.

W$INPUT.PROMPT	The prompt text associated with this input
W$INPUT.TYPE	"TEXT", "PASSWORD", "SELECT", "DATE" or "CHECKBOX"
W$INPUT.SELOPTS	For SELECT type, this is a value-mark-delimited list of selection options presented to the user in a drop-down list.
W$INPUT.SELVALS	For SELECT type, this is a value-mark-delimited list of values associated with the options provided. The user does not see these values.
W$INPUT.DEFAULT	The default value to be displayed to the user. For SELECT type, this should be the value data, not the option text.
W$INPUT.UDPOS	The element of the WIDGET.USER.DATA array in which the user's input/selection will be stored.
W$INPUT.PARAMS	If specified, this string is added as an attribute to the <input> or <select> HTML tag. Multiple attributes for a single tag must be separated by spaces (For example: `W$INPUT.PARAMS<1,1>=\size=30 maxlength=20 style="color:white;background-color:red"\`) creates a text box with a red background that is sized at 30% of the available width. A maximum of 20 white characters are allowed.

W$INPUT.PROMPT.PARAMS	If specified, this string is used to style the prompt text specified in the W$INPUT.PROMPT string in the same way that the W$INPUT.PARAMS string is used to style the actual field itself.
W$INPUT.BUTTON	The name of the submit button. If not specified, "Go" will be used. This is only a single value – only one button is present to submit all input fields for any widget.

Additional Features

W$PRINTABLE	If set to 1, a print icon is added to the widget's title bar. Clicking the printer icon will open the widget in a new window and allow the user to print the widget.
W$PDFABLE	If set to 1, and the dashboard PDF feature is enabled (see Appendix III), a PDF icon is added to the widget's title bar. Clicking the PDF icon will render the widget as a PDF file and deliver it to the user's browser. The user will be able to save or view the generated PDF file.
W$HIDDEN	If set to 1, the widget is run by the dashboard but not displayed.
W$CHART.HEIGHT	The height in pixels of the chart. If left unspecified, Dashboard will automatically select a chart height based on the `CHART$WIDTH` setting. Note that `W$CHART.HEIGHT` only applies to pie, line, bar, area and column charts.

Widget Types

Each widget must declare its type and provide the necessary information for that widget type. Some widget types are unique, such as HTML, TEXT, and TABLE widgets. However, many chart and graph widget types are similar to one another in the information that they need in order to present the graphic display. For example, a line graph and a column chart are similar in that they involve plotting a series of values along a series of categories. Therefore, these two widget types require the same variables to be populated with information in the same way.

The following topics are presented:

Default Chart Options

Each widget type has a set of default options defined in the MVDB.CONTROL file. For example, the default settings for the FC3DPIE chart is stored in the FC3DPIE.SETTINGS record as follows:

```
<graph caption='[CHART.TITLE]' decimalPrecision='0' showNames='1'
showValues='0'showPercentageInLabel='0' pieYScale='45' pieRadius='[RADIUS]' animation='0'
pieFillAlpha='100' >
[CHART.DATA]
</graph>
```

The values in brackets [] are replaced with specific information when the widget is created. For example, CHART.TITLE comes from W$PIE.CAPTION while RADIUS and CHART.DATA are created by the Dashboard Controller program.

The chart parameters described in this record for each widget type can be overridden with the W$CHART.OPTIONS variable. Any chart-level parameters can be added in this record to specify default behaviors for each widget type. For example, the parameter "showPercentageInLabel" is set to "0" in the settings record. Note that the information in the ".SETTINGS" records does not include a complete list of the available settings for each chart.

Pie Charts

A pie chart widget can be created using the following widget types:

1. FC2DPIE - 2-dimensional Pie Chart
2. FC3DPIE - 3-dimensional Pie Chart
3. FCDOUGHNUT2S - 2-dimensional Doughnut Chart

The following variables are used to define a pie chart widget:

W$PIE.LABELS	VM-Array of labels for each pie slice
W$PIE.VALUES	VM-Array of values for each pie slice
W$PIE.VALUE.OPTS	VM-Array of *Optional Parameters* associated with each pie slice.
W$PIE.CAPTION	Text caption for the pie chart itself
W$PIE.COLORS	VM-Array of color choices to override default colors

Single-Series and Multi-Series Bar, Column, Line and Area Charts

Bar, column, line and area chart widgets can be created as single-series charts, or multi-series charts. A single-series chart is defined by a single list of data points, each with a corresponding value. For example, a monthly sales chart for a single year contains one data point for each of the twelve months; the sales volume. A multi-series chart is defined by more than one list of related data points, each with a corresponding value. An example of a multi-series chart would be monthly sales by product category for a single year. In such a chart, multiple data points represent one value for each category.

Line charts are most useful for representing trends, while bar and column charts provide an effective way to present comparative information. Bar charts are identical to column charts from a data perspective; however a column chart's vertical bars are better suited for presenting a progression toward a goal like sales, profit or margin. Bar charts present horizontal bars, which are useful in presenting information such as sales by geography or demographics in which goals may not be relevant.

A multi-series column cart may be represented as a "stacked column chart" in which the columns are represented in a vertical "stack" rather than a series of individual, adjacent vertical columns. Similarly, a multi-series line chart may be represented as an "area chart" in which each line's value represents a portion of the whole. Therefore, an area chart is ideal for presenting multi-series trend information, and a stacked column chart is ideal for presenting comparative data.

Single-series bar, column, line and area charts are created using any of the following types:

1. FCAREA2D – 2-dimensional Area Chart
2. FCBAR2D – 2-dimensional Bar Chart
3. FCCOLUMN2D – 2-dimensional Column Chart
4. FCCOLUMN3D – 3-dimensional Column Chart
5. FCLINE – Line Chart

Multi-series bar, column, line and area charts are created using any of the following types:

1. FCMSAREA2D – 2-dimensional Area Chart
2. FCMSBAR2D – 2-dimensional Bar Chart
3. FCMSCOLUMN2D – 2-dimensional Column Chart
4. FCMSCOLUMN3D – 3-dimensional Column Chart
5. FCMSLINE – Line Chart
6. FCSTACKEDAREA2D – 2-dimensional Stacked Area Chart
7. FCSTACKEDBAR2D – 2-dimensional Stacked Bar Chart
8. FCSTACKEDCOLUMN2D – 2-dimensional Stacked Column Chart
9. FCSTACKEDCOLUMN3D – 3-dimensional Stacked Column Chart

These charts are very similar, and the methods used in creating these widgets are virtually identical. The following variables are used to define both single- and multi-series charts:

W$BAR.LABELS	AM-Array of labels for each series of a multi-series chart. Single-series charts do not use this value.
W$BAR.VALUES	VM-Array of data point values for any series. For multi-series charts, the variable contains a two-dimensional array in which the attributes correspond with W$BAR.LABELS.
W$BAR.VALUE.OPTS	VM-Array of *Optional Parameters* associated with each value
W$BAR.CAPTION	Text caption for the chart itself
W$BAR.COLORS	VM-Array of color choices to override default colors
W$BAR.XLABELS	VM-Array of labels for each value on the X-axis. For multi-series charts, the variable contains a two-dimensional array in which the attributes correspond with W$BAR.LABELS
W$BAR.XMEMO	The text to appear beneath the X-axis of the chart
W$BAR.TMEMO	The text to appear alongside the Y-axis of the chart
W$BAR.LABEL.OPTS	AM-Array of *optional parameters* associated with the W$BAR.LABELS values
W$BAR.XLABELS.OPTS	VM-Array of *optional parameters* associated with the W$BAR.XLABELS values
Trendlines	
W$BAR.TREND.BEG	VM-Array of beginning points of trendlines
W$BAR.TREND.END	VM-Array of ending points of trendlines
W$BAR.TREND.OPTS	VM-Array of *optional parameters* for each trendline

Text Tables

Widgets can create HTML tables by setting the W$TYPE to "TABLE" and the following variables:

W$TABLE.COL.LABELS	VM-Array of column heading labels.
W$TABLE.COL.JUST	VM-Array of text-justification settings ("left", "right", "center").
W$TABLE.DATA	AM/VM-Array. Each AM is a table ROW, each VM is a column within the row.
W$TABLE.TOTALS	VM-Array of total values (optional) at the bottom of each column.

This type of widget allows for easy creation of HTML-based tables. However, if you need more control over the way that these tables are presented, use the HTML type widget to create more complex tables.

HTML Data

Widgets can create HTML data by setting the W$TYPE to "HTML" and assigning the W$HTML.DATA variable to the desired HTML content. It is not necessary to include <html>, <head> or <body> tags as these are already included in the outer template for every dashboard. If a widget includes a <head> tag in the HTML content, the widget will be rendered in an "iframe" when displayed on the dashboard to preserve any styles in the content. Otherwise the styles used by the widget are derived from the dashboard theme.

Text Data

Widgets can create text data by setting the W$TYPE to "TEXT" and assigning the W$TEXT.DATA variable to the desired text output. Text data will be displayed in a mono-spaced font and will be presented in <pre> tags on the HTML page.

Programming Notes

- Multiple widgets may be defined to call a single subroutine. The subroutine may use the G$WIDGET.NAME global variable to determine the name of the widget that is associated with the execution of the program.
- WIDGET.USER.DATA and GLOBAL.USER.DATA are stored and retrieved using MATWRITE and MATREAD statements. Therefore, you must not store attribute-mark delimited arrays in any of the associated dimensioned array positions.
- Widget programs must be designed to execute very quickly. Users may add several widgets to a single dashboard page, so it is important that the widgets run as fast as possible. Look at G$CALL.DUR to see the duration of each subroutine call. You must use the last widget in the dashboard to see the duration of each widget call before it.
- Widget programs must have no output to the screen. Any output generated by the widget programs can corrupt the HTTP response headers and the HTML structure.

Widget Testing and Debugging

MV Dashboard provides several tools to aid in debugging dashboard widgets: debug mode, logging and a special debugging execution environment.

Common widget programming errors are:

- Terminating your widget subroutine with a STOP instead of RETURN.
- Failing to catalog your widget subroutine
- Failing to open a file before reading data
- Using "ELSE STOP" on an OPEN statement when proper Q-pointers have not been configured in the dashboard account
- Calling a subroutine from your widget program that has not been cataloged in the dashboard account
- Having an extra END statement in your widget subroutine
- Allowing your widget subroutine to generate output to the screen

Using Widget Debug Mode

In normal usage, if dashboard controller program (MVDB.MAIN) detects that a widget has failed to execute properly, the widget is automatically placed into "debug mode" and removed from dashboard until it has been returned to "normal mode". An administrator can toggle the widget mode (normal or debug) by clicking the "i" icon on the widget control menu.

When a widget is in "debug mode", instead of displaying normal content, debugging information about the widget is displayed. This includes the name of the widget subroutine, any runtime errors generated by the widget subroutine (if the MultiValue platform supports logging of runtime errors), and variables from WIDGET.USER.DATA.

Using SUB.LOG.DEBUG.INFO to debug widgets

The SUB.LOG.DEBUG.INFO subroutine is provided to allow widget code to save important information in a log, for review after the widget has been run. Up to 100 lines of information may be saved in the widget log.

To enable the logging feature, you must create a log file. SUB.LOG.DEBUG.INFO writes the widget log in the WDB.DEBUG file. The data section of this file is not normally created by the MV Dashboard installation, only the dictionary. Log in to the MVDB account and manually create the data section for WDB.DEBUG. Add CALL SUB.LOG.DEBUG.INFO to your widget code to save important information in the log. The SUB.LOG.DEBUG.INFO subroutine requires two arguments: widget name, and message. When the message is saved in the log, a timestamp is written before the message text. It is possible to send a dynamic array as part of the message; each attribute is saved on its own line in the log.

To view the log, place the widget into debug mode, as described in the previous topic. The log contents are displayed after any runtime errors.

To disable logging, delete the data section of the WDB.DEBUG file.

Using MVDB.DEBUG to debug widgets

It is easier to identify and resolve problems by testing a new or modified widget program before running it in the dashboard environment. This can be done using the widget debugging program, MVDB.DEBUG.

The MVDB.DEBUG program requires the MVDB.DEBUG.INFO file. This file is not normally created by MV Dashboard installation. Log in to the MVDB account and manually create the MVDB.DEBUG.INFO file before using MVDB.DEBUG.

At TCL, type MVDB.DEBUG to start the testing process. This is a simple program that configures the widget runtime environment and calls your widget subroutine. It does not show you the results created by the widget program, the dashboard environment is the best place to see what your widget does. However, if your widget program encounters runtime warnings or errors, it can prematurely terminate the dashboard controller program.

The MVDB.DEBUG program provides an easy way for you to make sure your widget subroutine returns control back to the dashboard controller program and doesn't produce any screen output. This program will also tell you the execute duration of your widget subroutine in milliseconds.

The example below shows that the widget subroutine SUB.LISTU ran in 217 milliseconds and did not produce any screen output:

```
Calling SUB.LISTU

Call completed. Duration: 217
```

The following example shows a widget subroutine that is generating a runtime-error which will impact the display of the dashboard in a web browser:

```
Calling SUB.TEST2

[B10] in program "SUB.TEST2", Line 22:

      Variable has not been assigned a value; zero used.

0

Call completed. Duration: 283
```

Integration with Other Applications

MV Dashboard provides two special access methods by which other applications can invoke dashboard widgets by posting properly formatted URL strings. The first method uses a loginid of *webservice* and uses a fully formatted URL string containing the desired dashboard and/or widget and any parameters. The second method uses a loginid of *emailed* and invokes the dashboard to display reports previously generated by the MmultiValue application.

In order to use the *webservice* or *emailed* logins with MV Dashboard, the features must be enabled in the Administrative Configuration widget of the Administrator dashboard.

In addition, MV Dashboard automatically provides an Excel icon for downloading a csv file on those widget types that can logically transition to a spreadsheet. Those widget types including bar , column, area, line, pie, and doughnut charts.

Integration Using Webservice Login

To display any dashboard page like it had been selected by a user logged into MV Dashboard, an application would post a URL with three additional parameters added to the same URL generated when manually selecting that dashboard. Those additional parameters are:

- loginid – this would be the literal "webservice"
- passtime – this would be the current time 24 hour clock time represented as HHMMSS
    - This time will be validated against the system clock on the multi-server running MV Dashboard.
    - If the passtime is not within 10 minutes of the system clock the user will be taken to the dashboard login page with a message that the times are not in sync.
- password – this would be a calculation based on the current date and the passtime parameter
    - Reverse the order of the digits in the passtime parameter and add that to the MMDDYY of the date. The sum will be the password submitted in the URL.
    - For example, a login at 2:15:30 PM on 12-02-15 would reverse 141530 to get 035141 and add it to 120215 to get a password of 155356
    - An invalid password will take the user to the dashboard login page with a message to that affect.

The basic install of MV Dashboard comes with a Demo Financial Company dashboard. To pop a browser display of that dashboard from another application at 2:15:30 PM on 12-02-15, the URL post would be:

http://*ipaddress*:8180/dbc/MVDB.MAIN?dbname=Demo%20Financial%20Company&loginid=webservice&passtime=141530&password=155356

To get the URL string to which the three parameters should be added, simply invoke any dashboard and look at the URL window to see the URL that generated it.

Any dashboard invoked by this method will have full drilldown capability but the only option available on the tab bar will be Log Out.

NOTE: This functionality is only available during the first 30 days on a single user MV Dashboard installation. To use it after the 30 day period, it will be necessary to purchase a multi-user license.

You must enable the *webservice* login from the Administrative Configuration widget in the Administrator dashboard before this feature can be used.

Integration Using Emailed Login

MV Dashboard can be used as a vehicle for presenting reports generated by the host MultiValue server. An example where this might be used is to have nightly reporting processes save the images of reports as they are run and to email a link for each report that the recipient can click on to view the report.

The request for this capability came about because of periodic issues when attempting to include html columnar reports in the body of an email. Letting MV Dashboard present the report through the browser eliminates that issue.

The report images can be either plain text or html. They should be stored in the MVDB.EMAILED.HTML file in the MVDB account. The first 5 characters of the record id in this file should be the 5 digit internal date when the report was generated. All reports in this file for more than 10 days will be automatically deleted.

If a monthly sales report was saved in the MVDB.EMAILED.HTML file as 17503_MOSALES, the URL string to view that report through the dashboard would be:

http://*ipaddress*:8180/dbc/MVDB.MAIN?loginid=emailed&htmlid=17503_MOSALES

You must enable the *emailed* login from the Administrative Configuration widget in the Administrator dashboard before this feature can be used.

Appendix I: Widget Output Type Codes

Code	Description
TABLE	Displays output in an HTML table structure.
HTML	Applies no formatting to output. NOTE: See the section on Drill Down Widgets for details on how to drill down from inside of an HTML widget.
TEXT	Wraps output in <pre> tags.
FC2DPIE	2D Pie Chart
FCDOUGHNUT2D	2D Doughnut Chart
FC3DPIE	3D Pie Chart
FCCOLUMN2D	2D Column Chart
FCCOLUMN3D	3D Column Chart
FCLINE	Line Chart
FCBAR2D	2D Horizontal Bar Chart
FCAREA2D	2D Area Chart
FCMSCOLUMN2D	2D Columnar Chart for Multiple Series
FCMSCOLUMN3D	3D Columnar Chart for Multiple Series
FCMSLINE	Line Chart for Multiple Series
FCMSAREA2D	2D Area Chart for Multiple Series
FCMSBAR2D	2D Bar Chart for Multiple Series
FCSTACKEDAREA2D	2D Stacked Area Chart for Multiple Series
FCSTACKEDCOLUMN2D	2D Stacked Column Chart for Multiple Series
FCSTACKEDCOLUMN3D	3D Stacked Column Chart for Multiple Series
FCSTACKEDBAR2D	2D Stacked Bar Graph for Multiple Series

The following topics are also presented:Specifying Colors

Theme Management

Specifying Colors

The W3C has listed 16 color names that you can use: aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white, and yellow. If you want to use other colors, you must specify their RGB or HEX value.

The Hex Codes for these colors may also be used via the COLOR$BLACK, COLOR$RED, etc. EQUates that have been assigned in WBPD, MVDB.INCLUDE file.

Theme Management

The administrative configuration widget allows an alternate logo and theme colors to be specified. Theme colors can be specified for individual dashboards from within the dashboard definition page. This allows MV Dashboard to be branded with the user's corporate identity. A few themes are included as examples.

An alternate logo may be specified in two ways. A fully-qualified URL may be provided if the logo image to be used resides on a remote system. For example, a reference to a remote image file might look like this: "http://www.example.com/images/logo.jpg". Alternatively, an image file can be stored in the docroot/db/images folder under the install directory for mvappsvr (/usr/local/mvappsvr by default). In this case, the filename alone must be specified in the alternate logo image field. If an image file is saved in the images folder used by MV Dashboard, specify only the file name (e.g. "logo.jpg") as the alternate logo image.

There are two parts to creating a custom theme. First, create a new css file and store under the docroot/db/themes directory. Use one of the provided themes as a starting point. When the new css file is in place, you must add to the available theme configurations in the D3 account into which MV Dashboard was installed. Add a record to MVDB.THEMES. The record key is the theme name. In the first field, enter the name of the css file you created in the docroot/db/themes directory. In fields two through six, specify colors to be used in the charts. The chart colors should be specified in the format #rrggbb where rr, gg and bb are hexadecimal values between 00 and FF, inclusive. These colors will be used to by the FusionCharts and FusionWidgets as defaults when widget colors are not explicitly set by the widget subroutine.

When the new css file and MVDB.THEMES entry are in place, the new theme will become available for selection from the theme drop down list in the Administrative Configuration widget or dashboard definition page.

Appendix II: Optional Parameters

Many of the widget types allow for the specification of optional parameters. These optional parameters can be applied to the entire chart/graph using the W$CHART.OPTIONS variable, or they can be applied to individual values (such as a specific slice of a pie chart).

Optional parameters must be specified in the following format:

```
parameter='value'
```

The value for each parameter must not contain any of the following characters: quote marks (single or double), greater than symbol (>), less than symbol (<), or ampersand (&). You may use the "&#*nn*;" encoding method (where *nn* is the decimal value of the character) if you want to use these special characters.

If multiple optional parameters are specified, they must be separated by spaces. Each widget type has a list of optional parameters.

The following topics are presented:

Optional Parameters for Line, Bar, Area, Column, Pie and Doughnut Charts (2D & 3D)

The following topics are presented:

Background Properties

`bgColor="HexColorCode"`	Sets the background color for the chart. You can set any hex color code as the value of this attribute. Must be used in conjunction with bgAlpha. IMPORTANT: You CANNOT assign a # at the beginning of the hex color code as this corrupts the XML data passed back to the chart. This also prohibits the use of the COLOR$ equates assigned in the WBPD, MVDB.INCLUDE file. This is true for all color options for all FusionCharts.
`bgAlpha="NumericValue(0-100)"`	Sets the alpha (transparency) of the graph. Used to set the intensity of the background color defined in bgColor.

Font Properties

`baseFont="FontName"`	Sets the base font family of the chart font which lies on the canvas i.e., all the values and the names in the chart which lie on the canvas will be displayed using the font name provided here.
`baseFontSize="FontSize"`	Sets the base font size of the chart i.e., all the values and the names in the chart that lie on the canvas.
`baseFontColor="HexColorCode"`	Sets the base font color of the chart i.e., all the values and the names in the chart that lie on the canvas.

Charts and Axis Titles

`caption="String"`	Determines the caption that appears at the top of the chart.
`subCaption="String"`	Sub-caption of the chart.
`xAxisName="String"`	x-Axis text title (if the chart supports axis, not available for Pie and Doughnut charts).
`yAxisName="String"`	y-Axis text title (if the chart supports axis, not available for Pie and Doughnut charts).

Number Formatting Options

`numberPrefix="$"`	Using this attribute, you can add a prefix to all the numbers visible on the graph. For example, to represent all dollars figure on the chart, you could specify this attribute to ' $' to show like $40000, $50000.
`numberSuffix="p.a"`	Using this attribute, you can add a prefix to all the numbers visible on the graph. For example, to represent all figures quantified as per annum on the chart, you could specify this attribute as ' /a' to show 40000/a, 50000/a.

To use special characters for numberPrefix or numberSuffix, you'll need to URL Encode them. That is, suppose you want to have numberSuffix as% (like 30%), you'll need to specify it as under: numberSuffix='%25'

`formatNumber="1/0"`	This configuration determines whether the numbers displayed on the chart will be formatted using commas, e.g., 40,000 if formatNumber='1' and 40000 if formatNumber='0 '
`formatNumberScale="1/0"`	Configuration whether to add K (thousands) and M (millions) to a number after truncating and rounding it - e.g., if formatNumberScale is set to 1, 10434 would become 1.04K (with decimalPrecision set to 2 places). Same with numbers in millions - an M will added at the end.
`decimalSeparator="."`	This option specifies the character used as the decimal separator in a number.
`thousandSeparator=","`	This option specifies the character used as the thousands separator in a number.
`decimalPrecision="2"`	Number of decimal places all numbers on the chart are rounded to.

Hover Caption Properties

The hover caption is the tool tip that shows up when the user moves his/her mouse over a particular data item (column, line, pie, bar etc.).

`showhovercap="1/0"`	Option whether to show/hide hover caption box.
`hoverCapBgColor="HexColorCode"`	Background color of the hover caption box.
`hoverCapBorderColor="HexColorCode"`	Border color of the hover caption box.
`hoverCapSepChar="Char"`	The character specified as the value of this attribute separates the name and value displayed in the hover caption box.

Optional Parameters Unique to Column, Line, Area & Bar

The following topics are presented:

Canvas Properties for 2D Charts ONLY

`canvasBgColor="HexColorCode"`	Sets the background color of the canvas.
`canvasBgAlpha="NumericalValue(0-100)"`	Sets the alpha (transparency) of the canvas.
`canvasBorderColor="HexColorCode"`	Sets the border color of the canvas.
`canvasBorderThickness="NumericalValue(0-100)"`	Sets the border thickness (in pixels) of the canvas.

Canvas Properties for 3D Charts Only

`canvasBgColor="HexColorCode"`	Sets the background color of the canvas. The background of the canvas is the one behind the columns.
`canvasBaseColor="HexColorCode"`	Sets the color of the canvas base, the base on which the columns are placed.
`canvasBaseDepth="Numerical Value"`	Sets the height (3D Depth) of the canvas base.
`canvasBgDepth="Numerical Value"`	Sets the 3D Depth of the canvas background.
`showCanvasBg="1/0"`	Sets whether or not to show the canvas background.
`showCanvasBase="1/0"`	Sets whether to show the canvas base.

Chart Numerical Limits

`yAxisMinValue="value"`	Determines the lower limit of y-axis.
`yAxisMaxValue="value"`	Determines the upper limit of y-axis.

If you don't specify any of the above values, it is automatically calculated based on the data provided.

Generic Properties

`shownames="1/0"`	Can be set to 1 or 0. It sets the configuration whether the x-axis values (for the data sets) are displayed or not. By default, this attribute assumes the value 1, which means the x-axis names will be displayed.
`showValues="1/0"`	Can be set to 1 or 0. It sets the configuration whether the data numerical values will be displayed along with the columns, bars, lines, and pies. By default, this attribute assumes the value 1, which means the values will be displayed.
`showLimits="1/0"`	Option to show/hide the chart limit textboxes.
`rotateNames="1/0"`	Configuration that sets whether or not the category name text boxes would be rotated.
`animation="1/0"`	Sets whether the animation is played or whether the entire chart is rendered at once.
`showColumnShadow="1/0"`	Whether or not the 2D shadow for the columns would be shown. *(for 2D charts only)*
`showLegend="1/0"`	Sets whether the legend would be displayed at the bottom of the chart. *(for Multi-series charts only)*

Font Properties

`outCnvBaseFont="FontName"`	Sets the base font family for all the values and names in the chart that lie outside the canvas.
`outCnvBaseFontSze="FontSize"`	Sets the base font size of all the values and the names in the chart that lie outside the canvas.
`outCnvBaseFontColor="HexColorCode"`	Sets the base font color of all the values and the names in the chart that lie outside the canvas.

Number Formatting Options

`divLineDecimalPrecision="2"`	Number of decimal places all divisional line (horizontal) values on the chart are rounded to.
`limitsDecimalPrecision="2"`	Number of decimal places upper and lower limit values on the chart are rounded to.

Zero Plane (for 2D Charts)

The zero plane is a simple plane (line) that signifies the 0 position on the chart. If there are no negative numbers on the chart, you won't see a visible zero plane.

`zeroPlaneThickness="Numeric Value"`	Thickness (in pixels) of the line indicating the zero plane.
`zeroPlaneColor="Hex Code"`	The color for the zero plane.
`zeroPlaneAlpha="Numerical Value 0-100"`	The transparency for the zero plane.

Zero Plane (for 3D Charts)

The zero plane is a 3D plane that signifies the 0 position on the chart. If there are no negative numbers on the chart, you won't see a visible zero plane.

`zeroPlaneShowBorder="1/0"`	Whether or not the border of a 3D zero plane is plotted.
`zeroPlaneBorderColor="Hex Code"`	If the border is plotted, sets the border color for the plane.
`zeroPlaneColor="Hex Code"`	Color for the zero plane.
`zeroPlaneAlpha="Numerical Value 0-100"`	The intended transparency for the zero plane.

Divisional Lines (Horizontal)

Divisional Lines are horizontal or vertical lines running through the canvas. Each divisional line signifies a smaller unit of the entire axis thus aiding the users in interpreting the chart.

`numdivlines="NumericalValue"`	Sets the number of divisional lines to be drawn.
`divlinecolor="HexColorCode"`	Color of the grid divisional line.
`divLineThickness="NumericalValue"`	Thickness (in pixels) of the grid divisional line.
`divLineAlpha="NumericalValue0-100"`	Alpha (transparency) of the grid divisional line.
`showDivLineValue="1/0"`	Option to show/hide the textual value of the divisional line.

These only apply to 2D Charts

`showAlternateGridColor="1/0"`	Option on whether to show alternate colored horizontal grid bands.
`alternateHGridColor="HexColorCode"`	Color of the alternate horizontal grid bands.
`alternateGridAlpha="NumericalValue0-100"`	Alpha (transparency) of the alternate horizontal grid bands.

Divisional Lines (Vertical) for 2D Charts

numVDivLines="NumericalValue:	Sets the number of vertical divisional lines to be drawn.
VDivlinecolor="HexColorCode"	Color of vertical grid divisional line.
VDivLineThickness="NumericalValue"	Thickness (in pixels) of the line.
VDivLineAlpha="NumericalValue0-100"	Alpha (transparency) of the line.
showAlternateVGridColor="1/0"	Option on whether or not to show alternate colored vertical grid bands.
alternateVGridColor="HexColorCode"	Color of the alternate vertical grid bands.
alternateVGridAlpha="NumericalValue0-100"	Alpha (transparency) of the alternate vertical grid bands.

Optional Parameters for Trendlines

Trendlines are the horizontal lines spanning the chart canvas that aid in interpretation of data with respect to some previous pre-determined figure.

`startValue='NumericalValue'`	The starting y-axis value for the trendline. E.g. if you want to plot a slanted trendline from value 102 to 109, the startValue would be 102.
`endValue='NumericalValue'`	The ending y-axis value for the trendline. E.g. if you want to plot a slanted trendline from value 102 to 109, the endValue would be 109. If you do not specify a value for endValue, it will automatically assume the same value as startValue.
`color='HexCode'`	Color of the trendline and its associated text
`displayValue='StringValue'`	Displays a string caption for the trendline by its side. Example: displayValue='Last Month High'. When you don't supply this attribute, it automatically takes the value of startValue.
`Thickness='NumericalValue'`	Thickness of the trendline.
`isTrendZone='1/0'`	Whether or not the trend displays a line, or a zone (filled colored rectangle).
`showOnTop='1/0'`	Whether or not the trendline/zone is displayed over other elements of the chart.
`alpha='NumericalValue0-100'`	Alpha (transparency) of the trendline.

Optional Parameters Unique to Single Series & Pie Charts

`name="string"`	Determines the name by which the set of data is represented in the chart. In the above example, the value of this attribute is "Jan" therefore this set of data would be represented on the chart with the name "Jan". Example: <set name='Jan' …>
`value="NumericalValue"`	Determines the numerical value for the set of data according to which the chart would be built for the concerned set of data. Example: <set name='Jan' value='12345' …>
`color="HexCode"`	Determines the color for the concerned set of data in which it would appear in the graph. Example: <set name='Jan' value='12345' color='636363' …>
`hoverText="String value"`	Shows the abbreviated names on the x-axis and avoid cluttering or to make the chart look more legible. However, you still have the option of showing the full name as tool tip using this attribute. Like, in our example, we're showing the abbreviated form "Jan" on our x-axis, but the full word "January" is shown as the tool tip. Example: <set name='Jan' value='12345' color='636363' hoverText='January'…>
`link="URL"`	Defines the hotspots in your graph. The hotspots are links over the data sets. Please note that you'll need to URL Encode all the special characters (like ? and &) present in the link. All the server side scripting languages provide a generic function to URL Encode any string - like in ASP and ASP.NET, we've Server.URLEncode(strURL) and so on. Example: <set … link='ShowDetails.asp%3FMonth=Jan' …> To open a link in a new window, just put n- in front of the link e.g., link="n-ShowDetails.asp%3FMonth=Jan".
`alpha="Numerical Value 0-100"`	Determines the transparency of a data set. The range for this attribute is 0 to 100. 0 means complete transparency (the data set won't be shown on the graph) and 100 means opaque. This option is useful when you want to highlight a particular set of data. Example: <set … alpha='100' …>
`showName="1"`	Can have either the value of 0 or 1. A 1 indicates that the name of this data set will be displayed in the graph, 0 indicates it won't be displayed. This attribute is particular useful when you want to show/hide names of alternate data items or say every x (th) data item. Example: <set … showName="1" …>
`isSliced="1"`	Determines whether or not the pie appears as a part of the total circle

	or is sliced out as an individual item. (Applies only to 2D Pie Charts) Example: <set … isSliced="1" …>

Optional Parameters Unique to Multiseries & Stacked Charts

`name="String"`	Determines the category name displayed on the x-axis as the data label. In our example, we've specified the category names as names of six months (in abbreviated format).
`hoverText="String"`	Shows the abbreviated names on the x-axis (to avoid cluttering or to make the chart look more legible). However, you still have the option of showing the full name as tool tip using this attribute. Like, in our example, we're showing the abbreviated form "Jan" on our x-axis, but the full word "January" is shown as the tool tip.
`showName="1/0"`	Can have either the value of 0 or 1. A 1 indicates that the name of this data set will be displayed in the graph, 0 indicates it won't be displayed. This attribute is particular useful when you want to show/hide names of alternate data items or say every x (th) data item.
`seriesName="String"`	Denotes the name of the dataset series. If you're plotting a monthly sales analysis for the years 2004 and 2003, the seriesName for the first dataset would be 2004 and the second would be 2003. This is the value that's shown in the legend.
`color="Hex Color"`	Sets the color for that particular set of data.
`showValue="1/0"`	Sets the configuration on whether the values (for this particular data set) will be shown alongside the data sets. You can set this value for individual datasets to highlight the most prominent data.
`alpha="0-100"`	Sets the alpha (transparency) of the entire dataset. You can also later specify alpha at the <set> level to override this value. For example, <dataset seriesName='Sales – 2001' color='FFF123' alpha='80' ..> <set value='1'> <set value='2'> <set value='3' alpha='90'> </dataset> In the above data, the <set> elements with the value 1 and 2 will have an alpha of 80 on the graph, whereas the one containing 3 as its value will have alphas 90.

Dataset specific area properties (for Area Charts only)

`showAreaBorder="1/0"`	Configuration whether the border over the area would be shown or not.
`areaBorderThickness="NumericValue"`	Sets the thickness (in pixels) of the area border.
`areaBorderColor="Hex Color"`	Sets the color of the area border.
`areaAlpha="1-100"`	Transparency of the area fill.
`value="NumericalValue"`	Determines the numerical value for the set of data according to which the chart would be built for the concerned set of data. Example: <set name='Jan' value='12345' …>
`color="HexCode"`	Determines the color for the concerned set of data in which it would appear in the graph. This value here overrides the value specified at dataset level. Example: <set name='Jan' value='12345' color='636363' …>
`link="URL"`	This attribute defines the hotspots in your graph. The hotspots are links over the data sets. Please note that you'll need to URL Encode all the special characters (like ? and &) present in the link. All the server side scripting languages provide a generic function to URL Encode any string - like in ASP and ASP.NET, we've Server.URLEncode(strURL) and so on. Example: <set … link='ShowDetails.asp%3FMonth=Jan' …> To open a link in a new window, just put n- in front of the link e.g., link="n-ShowDetails.asp%3FMonth=Jan".
`alpha="Numerical Value 0-100"`	Determines the transparency of a data set. The range for this attribute is 0 to 100. 0 means complete transparency (the data set won't be shown on the graph) and 100 means opaque. This option is useful when you want to highlight a particular set of data. This value here overrides the value specified at dataset level. Example: <set … alpha='100' …>

Optional Parameters Unique to Pie Charts & Doughnut

The following topics are presented:

Generic Properties

`shownames="1/0"`	Option to show/hide the data names displayed alongside the pie.
`showValues="1/0"`	Options to show/hide the data values displayed along with the pies.
`showPercentageValues="1/0"`	If you've opted to show the data value, this attribute controls whether to show percentage values or actual values. NOTE: The normal state is to show percentages. Use `showPercentageValues='1'` to show values instead.
`showPercentageInLabel="1/0"`	If you've opted to show the data value, this attribute controls whether to show percentage values or actual values in the pie labels.
`animation="1/0"`	Sets whether or not the animation is played or the entire chart is rendered at once. *(Applies only to 2D Charts)*

Pie Properties

`pieRadius="Numeric Pixels"`	The best fit pie radius for the chart is automatically calculated. If you want to enforce radius values, you can set it using this attribute.
`pieSliceDepth="Numeric Value"`	Sets the 3D height (depth) of the pies on the chart (in pixels). *(applies only to 3D Pie Charts)*
`pieYScale="Numeric Value 30-100"`	This value sets the skewness of the pie chart (vertical slant). *(applies only to 3D Pie Charts)*
`pieBorderThickness="Numeric Value"`	Sets the border of each pie on the chart.
`pieBorderAlpha="0-100"`	Sets the border transparency for all the pie borders.
`pieFillAlpha="0-100"`	Sets the transparency for all the pies on the chart.

Name/Value Display Distance Control (Applies to 2D Pie Charts Only)

`slicingDistance="Numeric Value"`	Controls the distance between the sliced pie and the center of other pies.
`nameTBDistance="Numeric Value"`	Sets the distance of the name/value text boxes from the pie edge.

Pie Shadow Properties (Applies to 2D Pie Charts only)

`showShadow="1/0"`	Option to show/hide shadow.
`shadowColor="Hex Code"`	If you want to set your own shadow color, you'll need to specify that color for this attribute.
`shadowAlpha="0-100"`	Sets the transparency of the shadow.
`shadowXShift="Numeric Value"`	Sets the x shift of the shadow pie from the actual pie. That is, if you want to show the shadow 3 pixels right from the actual pie, set this attribute to 3. Similarly, if you want the shadow to appear on the left of the actual pie, set it to -3.
`shadowYShift="Numeric Value"`	Sets the y shift of the shadow pie from the actual pie. That is, if you want to show the shadow 3 pixel below the actual pie, set this attribute to 3. Similarly, if you want the shadow to appear above the actual pie, set it to -3.

Name/Value Display Distance Control (Applies Only to 2D Doughnut Charts)

`nameTBDistance="Numeric Value"`	Sets the distance of the name/value text boxes from the pie edge.

Pie Shadow Properties (Applies only to 2D Doughnut Charts)

`showShadow="1/0"`	Option to show/hide shadow.
`shadowColor="Hex Color"`	If you want to set your own shadow color, you'll need to specify that color for this attribute.

Optional Parameters Unique to 2D Line Charts

The following topics are presented:

Line Properties

`lineColor="Hex Code"`	If you want the entire line chart to be plotted in one color, set that color for this attribute.
`lineThickness="Numeric Value"`	Thickness of the line (in pixels).
`lineAlpha="0-100"`	Transparency of the line.

Line Shadow Properties

`showShadow="1/0"`	Option to show/hide the shadow.
`shadowColor="Hex Code"`	If you want to set your own shadow color, you'll need to specify that color for this attribute.
`shadowThickness="Numeric Value"`	Sets the thickness of the shadow line (in pixels).
`shadowAlpha="0-100"`	Sets the transparency of the shadow line.
`shadowXShift="Numeric Value"`	Sets the x shift of the shadow line from the chart line. That is, if you want to show the shadow 3 pixel right from the actual line, set this attribute to 3. Similarly, if you want the shadow to appear on the left of the actual line, set it to -3.
`shadowYShift="Numeric Value"`	Sets the y shift of the shadow line from the chart line. That is, if you want to show the shadow 3 pixel below the actual line, set this attribute to 3. Similarly, if you want the shadow to appear above the actual line, set it to -3.

Anchor Properties

Anchors (or the marker points) are the polygons that appear at the joint of two consecutive lines. On a line chart, the anchors are the elements that react to the hover caption and link for that particular data point. You can customize all the facets of the anchors using the properties below:

`showAnchors="1/0"`	Configures whether or not the anchors are shown on the chart. If the anchors are not shown, then the hover caption and link functions won't work.
`anchorSides="Numeric Value greater than 3"`	Sets the number of sides the anchor will have. E.g., an anchor with 3 sides would represent a triangle; with 4 it would be a square and so on.
`anchorRadius="Numeric Value"`	Sets the radius (in pixels) of the anchor. The greater the radius, the bigger the anchor size.
`anchorBorderColor="Hex Code"`	Border color of the anchor.
`anchorBorderThickness="Numeric Value"`	Thickness of the anchor border (in pixels).
`anchorBgColor="Hex Code"`	Background color of the anchor.
`anchorBgAlpha="Numeric Value"`	Alpha of the anchor background.
`anchorAlpha="Numeric Value"`	This function lets you set the transparency of the entire anchor (including the border). This attribute is useful, when you do not want the anchors to be visible on the chart, but you want the hover caption and link functionality. In that case, you can set anchorAlpha to 0.

Optional Parameters Unique to 2D Area Charts

The following topics are presented:

Area Properties

`showAreaBorder="1/0"`	Option to show/hide the border.
`areaBorderThickness="Numeric Value"`	Sets the thickness (in pixels) of the area border.
`areaBorderColor="Hex Color"`	Sets the color of the area border.
`areaBgColor="Hex Color"`	If you want the entire area chart to be filled with one color, set that color for this attribute.
`areaAlpha="0-100"`	Transparency of the area fill.

Divisional Lines (Horizontal)

Divisional Lines are horizontal or vertical lines running through the canvas. Each divisional line signifies a smaller unit of the entire axis thus aiding the users in interpreting the chart.

numdivlines="NumericalValue"	Sets the number of divisional lines to be drawn.
divlinecolor="HexColorCode"	Color of grid divisional line.
divLineThickness="NumericalValue"	Thickness (in pixels) of the grid divisional line.
divLineAlpha="NumericalValue0-100"	Alpha (transparency) of the grid divisional line.
showDivLineValue="1/0"	Option to show/hide the textual value of the divisional line.
showAlternateHGridColor="1/0"	Option show/hide alternate colored horizontal grid bands.
alternateHGridColor="HexColorCode"	Color of the alternate horizontal grid bands.
alternateHGridAlpha="NumericalValue0-100"	Alpha (transparency) of the alternate horizontal grid bands.

Divisional Lines (Vertical)

`numVDivLines="NumericalValue"`	Sets the number of vertical divisional lines drawn.
`VDivlinecolor="HexColorCode"`	Color of vertical grid divisional line.
`VDivLineThickness="NumericalValue"`	Thickness (in pixels) of the line.
`VDivLineAlpha="NumericalValue0-100"`	Alpha (transparency) of the line.
`showAlternateVGridColor="1/0"`	Option to show/hide alternate colored vertical grid bands.
`alternateVGridColor="HexColorCode"`	Color of the alternate vertical grid bands.
`alternateVGridAlpha="NumericalValue0-100"`	Alpha (transparency) of the alternate vertical grid bands.

Appendix III: PDF Generation

MV Dashboard can be configured to generate PDF versions of most widgets. An external PDF conversion utility is used to transform widget HTML into a PDF file, which is then sent to the user's browser for display or saving.

In order to use the PDF feature, an appropriate HTML to PDF conversion utility must be installed on the same server as MV Dashboard. The converter must be accept a command line containing the input and output file names and must run without any prompting. Zumasys has tested and recommends the open-source 'wkhtmltopdf' conversion program for use with MV Dashboard.

The PDF Configuration widget is available from the Administrator dashboard. Specify the following items, then click the Save button.

- Enable PDF Generation - set this to "yes" to enable MV Dashboard widgets to generate PDF files. Note: the widget subroutine must set W$PDFABLE=1 to enable this feature. When PDF generation is enabled here (Administrator dashboard), and in the widget subroutine, a PDF icon is added to the widget toolbar. Clicking the icon invokes the PDF converter command, and if successful, sends the generated PDF file to the user's browser.
- PDF Converter Command – enter the actual command line to invoke the PDF converter. This should include the full path to the converter executable file, any required command line options, and tokens for the input file and output file. The tokens are [INPUT_FILE] and [OUTPUT_FILE]. Note that for some Windows MultiValue platforms, the command line may require "cmd.exe /c" before the actual converter command.
- Directory for PDF files – enter the full path of a directory that will contain the generated PDF files. During PDF generation, the source HTML is also temporarily written to this directory. For MultiValue platforms which support directory-type files, this file would be the path to the directory for the NATIVE.CONTENT file in the MVDB account. For other platforms, it can be any valid directory path
- Delete PDF files after - select the number of days before PDF files will be automatically deleted. If you select "immediately" for this setting, the PDF files will be deleted 30 to 90 minutes after generation.

When PDF generation is enabled and the Save button is clicked, the PDF Configuration widget will attempt to create the directory for PDF files automatically.

For MultiValue platforms which support directory-type files (jBASE, QM, Unidata, Universe), the dictionary of the NATIVE.CONTENT file will be created as a normal hashed file in the MVDB account, and the data section of the NATIVE.CONTENT file will be created as a directory file.

For other MV platforms (D3), the directory specified in the PDF configuration will be created (if possible). The dictionary of the NATIVE.CONTENT file will be created as a normal hashed file in the MVDB account, and the data section of NATIVE.CONTENT will be an OSFI Q-pointer to the specified directory.

wkhtmltopdf

Zumasys has tested MV Dashboard with the open-source wkhtmltopdf package. While you are free to use other conversion programs with MV Dashboard, wkhtmltopdf is the only program which has been tested. The PDF Configuration widget will supply default values for the converter command based on the default installation of wkhtmltopdf. If you install wkhtmltopdf in another location be sure to adjust the path in the converter command line.

wkhtmltopdf Installation (Windows)

Download the 32 bit installation package for wkhtmltopdf from: http://download.gna.org/wkhtmltopdf/0.12/0.12.3/wkhtmltox-0.12.3_msvc2013-win32.exe. Install in the default location of C:\Program Files (x86)\wkhtmltopdf.

wkhtmltopdf Installation (Linux)

The Linux version of wkhtmltopdf requires the *X.org 75dpi fonts* package to be installed before installing wkhtmltopdf. If your Linux installation does not have this package installed, please download and install it before installing wkhtmltopdf.

```
yum -y install xorg-x11-fonts-75dpi
```

Then download the appropriate version of wkhtmltopdf based on your platform. Zumasys recommends installing the 0.12.2.1 version of wkhtmltopdf. While not the latest version, it is available as an rpm package. Download wkhtmltopdf from download.gna.org/wkhtmltopdf. For example, to download the 64 bit Centos6 version of wkhtmltopdf, use this commad:

```
wget http://download.gna.org/wkhtmltopdf/0.12/0.12.2.1/wkhtmltox-0.12.2.1_linux-centos6-amd64.rpm
```

Install using:

```
rpm -Uvh wkhtmltox-0.12.2.1_linux-centos6-amd64.rpm
```

Substitute the actual filename in the above command line. The PDF Configuration widget assumes the following location for the wkhtmltopdf executable: /usr/local/bin/wkhtmltopdf

After installing wkhtmltopdf be sure to test it from a command line to ensure there are no missing dependencies. For example:

```
wkhtmltopdf http://www.google.com.ph google.pdf
```

Fusion Charts Documentation

For complete documentation, see these websites:

http://www.fusioncharts.com/free/docs

Copyright

Document Version: 1.5.4

Date: November 7, 2016