

DAQConnect Corporation

5/3/2012

Copyright © 2012 DAQConnect Corporation. All rights reserved worldwide.

Information in this document is subject to change without notice.

All brand and product names are trademarks of their respective holders.

This manual: Copyright © 2012 DAQConnect Corporation. All rights reserved worldwide.

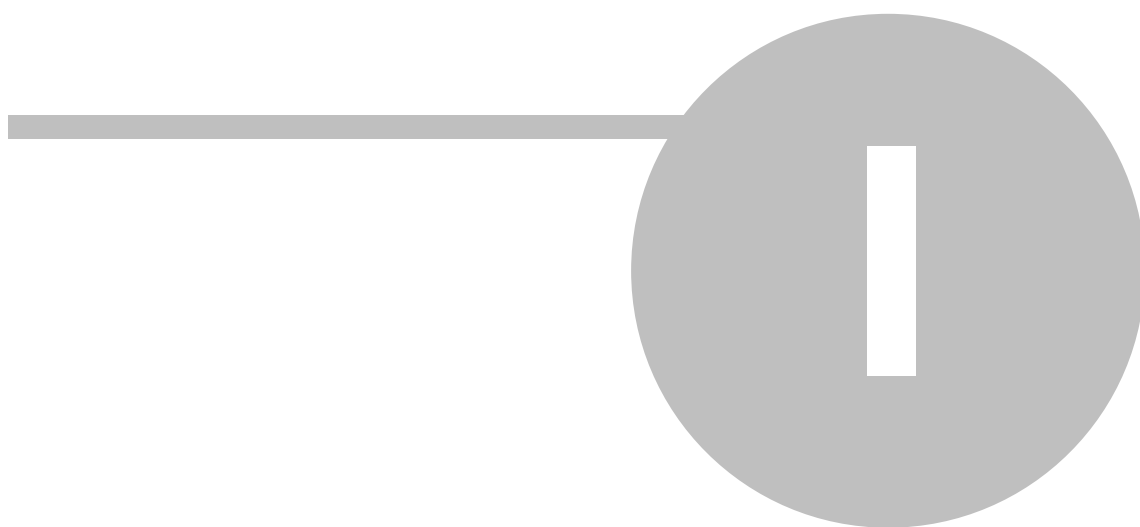
No portion of this manual may be copied, modified, translated, or reduced into machine-readable form without the prior written consent of DAQConnect Corporation.

Table of Contents

Introduction	6
Welcome	6
System Requirements	6
Basics	6
Definitions	8
Data Source	8
Tag	8
Browser Client	8
Page	8
Control	8
Using	10
Application screen.....	10
Menus	10
Basics	10
Data Sources and Tags.....	10
Overview	10
Data Source Properties.....	12
Alerts and Messages.....	13
Overview	13
Visuals	13
Controls	13
Overview	13
Control ID	14
Conditional properties	14
The Controls	14
Static Text	14
Value Display	15
Symbol	15
Trend Graph	16
Angular Gauge	17
Linear Gauge	19
Text Button	20
Date Edit	21
Time Edit	21
Slider	21
Alarm Status	22
Alarm History	22
Custom Control	22
Overview	22
Examples	23
Change Page Combo.....	23

Symbols	24
Overview	24
Pages	25
Overview	25
Page Properties	25
Page Variables	26
Embedding pages	26
Edit	27
Overview	27
Data	28
Data Sets	28
Overview	28
Data Set Properties	29
Alarms	29
Overview	29
Editing Alarms	30
Editing Alarm Groups.....	31
Utilities	33
Utilities	33
Overview	33
Account	34
Overview	34
Control capability	34
Expressions	37
Overview	37
Referencing tags	37
General Math Functions and Operators.....	38
Date and Time functions.....	38
Script	41
Script Overview	41
Variables	41
Accessing on-screen controls from script.....	41
Moving and resizing from script.....	42
Changing styles from script.....	42
Downloading data from script.....	42
Remote Connectors	45
RESTful interface	45
Windows DLL	46
DAQFactory	48

Introduction



Introduction

Welcome

With this tool you can view real-time data over the internet from a web browser. The data can come from a computers or devices that is securely behind a firewall without having to expose it to the Internet. The data can come from multiple sources and be presented on custom pages that you design in the browser using various controls. Historical data is kept on our servers to allow you to view historical trends in addition to current data.

System Requirements

There are two parts to this tool: the data source and the client browser. Data sources are available only for the PC running Windows and several embedded hardware platforms, but we are constantly adding new data sources and do so based on customer request, so please tell us if you need one for a different system. A data source can even be on an embedded data acquisition device, so if you are a hardware manufacturer, please contact us to see about adding support to your device to allow your device to communicate directly with our system.

On the data source side, the requirements are minimal. Communication with our servers just needs an internet connection that allows traffic on port 443 (SSL or HTTPS). This is the normal web browser port for doing secure web browsing, such as purchasing something on a secure web site, so more than likely if you can browse the internet from your PC, then the data source will be able to communicate with our servers. Data sources on PCs may be part of a larger software package which may have additional requirements. Please see the product documentation for more information.

To view your real-time data, you will need a web browser capable of running JavaScript. This is not operating system specific, so you can view your data on a Windows PC, Mac, Linux, even a PDA provided the browser supports modern Javascript. Most common browsers will work fine, and the best way to find out if it works is to simply give it a try. If, when you go to view a Page you do not see anything, then the browser either has Javascript disabled, or does not support Javascript and so won't work. Unfortunately, each browser manufacturer uses slightly different flavors of Javascript, so there is the possibility that more obscure browsers won't work. In this case, please use one of the more common ones like Firefox or Chrome. For best performance, we recommend Chrome as it has the fastest implementation of JavaScript. Firefox is a close second, with IE running significantly slower.

Basics

This tool consists of three parts: data sources, our servers, and client browsers. Data sources are software on a computer or data acquisition device that collect data and send it to our servers. You can have one data source or many data sources sending our servers data. Our servers collect this data and make it available for real time display and historical trending. Client browsers connect to our servers, much the same way one would browse any Internet site, and can view the data on user-designed screens.

Definitions



Definitions

Data Source

a piece of software that runs on a computer that collects data from other software and sends it to our servers. The data source can be in many different forms. It can be embedded in a software tool such as the OPC Quick Client or DAQFactory. It is available as a DLL and other library forms to allow you to add a connector to your own software. It can also be embedded in a data acquisition device to allow data to flow directly from the device to our servers without an intermediary computer. Each data source in your account has a unique ID that is a string of numbers and letters such as: 7C782B26-D936-4F6B-8822-315CA6EFF7D6. This ID tells both our servers and your data source where the data is coming from and who it is for. Typically you do not need to worry about the ID and it is automatically assigned.

Tag

the name of an input or output value. This could correspond to an input or output pin on a data acquisition device, or it could be a variable in software, or value from a device such as the Lat and Lon on a GPS. On our servers, tags can build up historical readings, up to a limit determined by your account, to allow you to view changes in the value over time.

Browser Client

a web browser running on a computer, PDA or other device that supports JavaScript and connects into our server to view the data in real time.

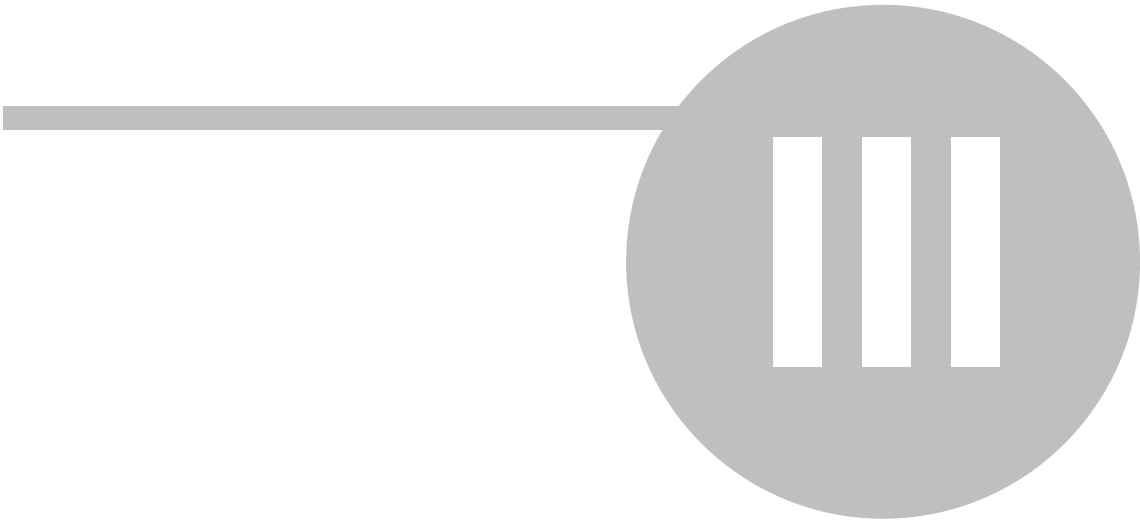
Page

a screen that you can design using various controls to display the data collected by our servers. You can have multiple pages to display your data in different ways, or different subsets of your data. Pages are viewed in web browser.

Control

an element on a page that either displays your data or provides background text or images. You assemble multiple controls of various types to create your pages. Controls have various properties to allow you to configure each one.

Using



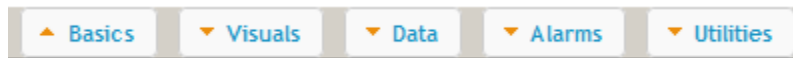
Using

Application screen

This application is a web app that runs in your browser. All the functionality is contained within a single web page. You can access it by logging in from our home page.

The application page consists of a top banner with a few buttons and menus and a large canvas to allow you to design and display your pages:

By default, the application starts in edit mode. To make any changes to your screens, connectors, data sets, or your account, you must be in edit mode. You can still view live data in edit mode. There is also run mode, which allows you to manipulate some of the screen controls such as sliders and edit boxes. To switch between run and edit mode, click on the button labeled "Run" or "Edit" at the top right corner of the page. When you are in edit mode, a menu appears along the top of the screen:



To view any of the menu items, simply click on the menu button. This will cause a floating popup window to appear. You can drag the window around as needed and even pin it so it doesn't disappear when you click on something else by clicking on the pin at the top right corner of the window. Within each window are panels that can be expanded or contracted by clicking on the orange title bar. Each is discussed in detail in the following sections.

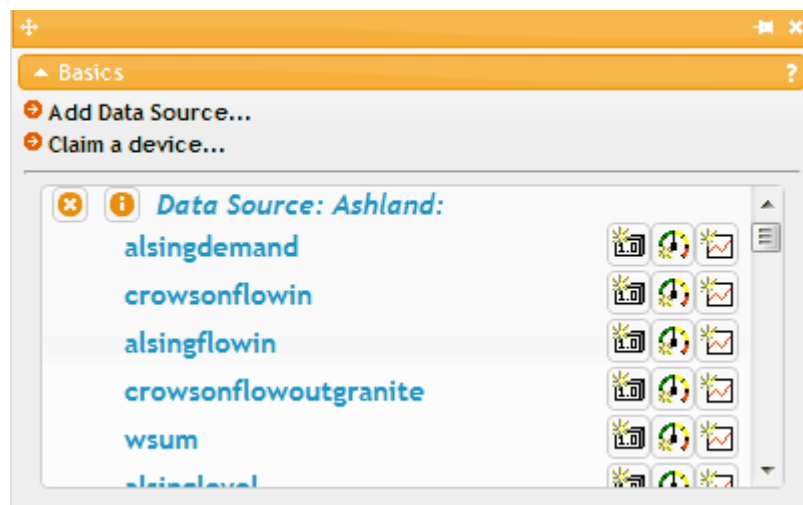
Note that some control functionality, such as the click events, and sliders, do not work in edit mode.

Menus

Basics

Data Sources and Tags

Overview



A data source is a remote piece of software that connects your data acquisition hardware or software to our servers. A data source could be a piece of data acquisition hardware with built in push capability, or a data acquisition

software package. You can have as many data sources as you need to collect data from multiple locations, subject to your account's tag limit. To begin sending data to out servers you need to create a new data source. This can be done two different ways depending on the type of data source.

Some software data sources, such as the OPC Quick Client, allow you to create and link to our servers all from the data source itself. You can even create your account from within that application. You typically only need to provide the data source a name, which should start with a letter and contain only letters, numbers or the underscore. Note that it is case sensitive!


For some data source types, such as DAQFactory, you need to provide it with a data source ID, so you need to click "Add Data Source..." to create a new data source, then open its properties to get the ID. This ID is entered into your remote hardware or software to tell it where to send its data. You will also likely need to specify which data you want sent and how much to keep. How this is done depends on the data source. Please see the data source section towards the end of the manual for help with some common data sources.

Other data source types, especially hardware such as the DAQBridge, have a claim mechanism. You simply plug your device into the network, click "Claim a device..." and enter the unique identifier on the device and a name for the data source, then cycle the power.

A tag is an I/O point on a data source. Tags are created by the data source itself. Once you link your data source and it starts sending tag data, you will see the tags listed in the window. You can quickly get a visual of the tag by dragging one of the three icons to the right of the tag name to the page.

Add Data Source...: adds a new data source for software tools that require a data source ID. You will be prompted for a name. The name must start with a letter and contain only letters, numbers or the underscore, and is case sensitive. The name is used both for the list in this menu, and in expressions you'll use in your controls to allow you to specify which data you wish to display.

Claim a device...: creates a data source for a hardware device that you just plugged in. You will be prompted for a name for the data source and the unique identifier for the hardware. Consult your hardware manual to see where this number might be. You typically need to cycle power on the hardware once claimed to prove to us that you actually own the hardware.

Data Source / Tag List: this lists all your data sources and all the tags in each data source. Click on the  next to a data source to open its properties. The properties window includes the data source ID you may need for your remote hardware or software. Please see the next section for more information about the [data source properties](#). To the left of each connector is an X. Clicking on this will delete the connector and all stored data. Your confirmation will be requested before deleting.

Below each data source is listed any tags that are being sent data from your data source. Until you setup your remote hardware or software to send data, no tags will appear. Once tags appear, you can click and drag any of the three icons to create visualization controls for that particular tag. The first icon creates a simple numeric display. The second creates an angular gauge. The third creates an historical trend.

Data Source Properties

The screenshot shows the 'Data Source Properties' window for a data source named 'Ashland'. At the top, it displays 'Data Source: Ashland' with a 'Refresh' button and 'Total Usage: 274177 data points'. Below this, the 'Data Source ID' is 'AA3236D2-A1F7-4319-A4E8-DC341BD997C5' and a plan summary states 'Your plan: 100 unique tags, 1000000 historical data points stored.'.

Tag Name	Usage (data points)	Capacity (data points)	Time of Last Data Point	Data Sets
<input type="checkbox"/> alsingdemand	10000 (100%)	10000		Change
<input type="checkbox"/> crowsonflowin	10000 (100%)	10000		Change
<input type="checkbox"/> alsingflowin	10000 (100%)	10000		Change
<input type="checkbox"/> crowsonflowoutgranite	10000 (100%)	10000		Change
<input type="checkbox"/> wsum	1294 (12%)	10000		Change
<input type="checkbox"/> alsinglevel	10000 (100%)	10000		Change
<input type="checkbox"/> w3	1294 (12%)	10000		Change
<input type="checkbox"/> maintank1pressure	2 (0%)	10000		Change
<input type="checkbox"/> crowsonvalveoutgranite	10000 (100%)	10000		Change
<input type="checkbox"/> meter2	216 (2%)	10000		Change
<input type="checkbox"/> i3	1294 (12%)	10000		Change
<input type="checkbox"/> w2	1294 (12%)	10000		Change
<input type="checkbox"/> alsingtempin	10000 (100%)	10000		Change
<input type="checkbox"/> crowsontempin	10000 (100%)	10000		Change
<input type="checkbox"/> crowsontemp	10000 (100%)	10000		Change
<input type="checkbox"/> i1	1294 (12%)	10000		Change
<input type="checkbox"/> maintank1temperature	1 (0%)	10000		Change

At the bottom, there is a 'Select:' dropdown with 'All' and 'None' options, followed by an 'Add selected tags to:' dropdown currently set to 'PowerLevels'. To the right of this are 'Clear data' and 'Delete' buttons. Further right is a 'Download all data as:' section with radio buttons for 'CSV', 'XML', and 'JSON'.

Once you have some data coming in, the data source properties window allows you to do a number of other things:

Clear data: this will clear the data stored for a particular tag(s) without actually deleting the tag(s). You must select the desired tags using the check boxes listed to the left of each tag.

Delete: this will completely delete a particular tag(s), removing all data. The delete tag(s) will no longer count towards your total allowed tag count for your account.

Change: next to each tag is listed the total number of data points being kept for the tag. You can change this value for each tag by clicking on the Change button. Note that the total data points for all tags across all your data sources must be less than the account limit displayed at the top right below the total usage. Usage and total usage is the current number of data points you have for a particular tag and in total across all tags and all data sources.

Add selected tags to: this will add the selected tags to the data set specified to the right of the button. You will need to create your datasets from the [Data Sets](#) menu. Any tag that is assigned to a data set will list that data set in the far right column. You can click on the data set name to jump to its properties, or click on the X to the left of the data set name to remove the tag from that data set.

Download data as: this will do a complete data dump of your connector, including all tags, in one of the three listed formats. This includes all data. No data alignment is performed, nor can you specify a date range.

The rest of the window provides various useful information about your tags. The total usage figure at the top right is across all data sources.

Alerts and Messages

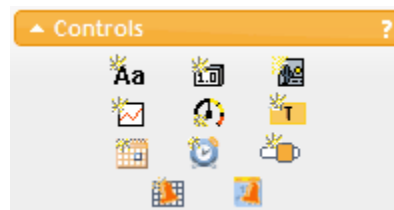
Overview

This panel is used to display messages and errors that may crop up during development. Because of the nature of development in real-time systems, we use this window instead of creating annoying popups. During development you may want to pin this menu so it is always in view. Any errors in your script discovered during execution will display here along with the time of the error (compiler errors are usually displayed in the script editing windows). The current browser time is also displayed for your reference.

Visuals

Controls

Overview



The application allows you to design your own pages to display your data in many different ways. Your pages are assembled from Controls that you drag from the Controls menu onto one of your pages. To change the properties of a control that is on a page, double click on it or click on it to select it then click on the wrench icon. This will open the properties window for the control. The properties vary per control. Some properties, such as Font and Text Color are pretty self explanatory. Others require more explanation, and we'll go into detail on each control shortly. Once a control is on a page, you can click and drag it to move it around the page, or click and drag an edge to resize the control. Additional tools for arranging your controls is available from the [edit menu](#).

Most changes to controls on a page only affect your local display and are not permanently saved. To save changes to the server so the page will reappear with the new settings, you must click on the save button at the top right corner of the screen, next to the logout button.



Static Text: displays text that does not change. This can be one word or complete paragraphs. You can also use this control without any text to create colored areas or panels.



Value Display: displays the result of an expression, which could be a tag or variable value, or a complex formula.



Symbol: displays a JPEG, GIF, or PNG image. DAQConnect includes a library of images you can use, or you can upload images of your own to create your own library.



Trend Graph: displays your data or variables in a graph over time



Angular Gauge: displays a gauge that rotates a pointer depending on a value



Text Button: displays a button with a text label with different styles for hover, pressed and disabled



Date Edit: an edit control that allows you to select a date.



Time Edit: an edit control that allows you to select a time.



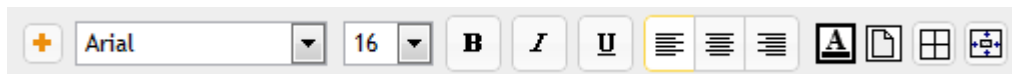
Slider: a horizontal or vertical bar with one or two handles that can be moved along the bar to select a value.

Control ID

Every control that you create on a page gets a unique ID. This ID is used when you want to access the control from script, either to retrieve values or set parameters. When you create a new control, an ID is automatically generated for it, however, these ID's are just numbered sequentially and thus not particularly user friendly. For controls that you want to access from script, we recommend renaming the ID to something more appropriate, which you can do from any of the control's properties window. The new ID must be unique, and also must start with a letter and contain only letters, numbers or the underscore.

Conditional properties

Many properties of controls have the option of being conditional properties. Conditional properties are properties that can change depending on conditions you specify. For example, you can make the background of a text control change to red when a value goes over 100. You can recognize conditional properties because there is a button with a plus next to them. Styling is a common conditional property that demonstrates this:



Until you click on the +, the parameters listed will be used in all cases. If you want different parameters to apply under different conditions, click on the +. The area will expand:



Now, in addition to the properties, there is an error where you can enter a condition, that when true (evaluates to a non-zero value), the specified parameters are used. Along the bottom is navigation that allows you to move among the options. The final one is always the fall back properties that are applied if none of the conditions pass:



From here you can also click + to add more conditions.

For some controls, such as the graph, the same user interface is used to add multiple traces. The difference, however, is that there isn't a condition specified. All parameters are used to create different traces.

The Controls

Static Text



The static text control simply draws a text label. By adjusting the size of the control you can make the control wrap long text inside the box. The default background color is transparent, thus letting controls under this control show through. You can change this and other styling, and even make the styling change based on conditions. The

parameters of this control are pretty self explanatory.

Hint: a static text control with no text is an empty block. If left with a transparent background, it makes a good clickable "hot-spot" on the screen. If given color, you have a panel, especially if you add a border.

Click:

This is script you can enter that will execute when the control is clicked in run mode. Please see the chapter on [script](#) for more details.

Property names for script access (intermediate):

you can access the properties of this control as described in the [script section](#). Here are the properties for this control:

text: the text displayed

mainStyle: the style used. Please see the section on [scripted changing of styles](#) for more information.

Value Display



The value control allows you to display a data point value in numeric or, possibly, text form. Its very similar to the static text control, except that what is displayed is determined by an expression. An expression is basically a formula, the result, of which, is displayed by the control. Expressions can be as simple as a tag name, or even just a number, or they can be more complicated, combining multiple tags, various formula and mathematical operators and numbers. They use Javascript syntax. For example: `Math.sin(tag * 3) / othertag`

The details of expressions are discussed later. For now, if you just want to display a value coming in from a data source, click on the down arrow to the right of the edit box and select the data source and the tag you wish to view.

Caption: this is a text label that is displayed in front of the value.

Expression: a formula in Javascript syntax, the result of which is displayed by the control.

Units: a text label that is displayed after the value.

Precision: determines how many digits past the decimal place are displayed. This only applies if the result of the expression is a number and not a string.

Property names for script access (intermediate):

you can access the properties of this control as described in the [script section](#). Here are the properties for this control:

caption

units

precision

expression: this is a string representing the expression

mainStyle: the style used. Please see the section on [scripted changing of styles](#) for more information.

Symbol



The symbol control allows you to display an image uploaded from your system. You can have the control display a different image depending on the state of your system.

The image can be a .jpg, .gif, or .png file. If it is a .gif or .png, DAQConnect will use any transparency in the image. Animated .gif images will be animated. To use the symbol control you have to upload your images, basically creating a personal library of images for use in the pages. You can do this from the Symbols panel of the Visuals menu. You will be prompted for the local file name which you can browse for, and a name for the symbol.

The properties of a symbol control are simple. By default, only one symbol is displayed. Click on the symbol to select a different one. If you want to make different symbols appear for different conditions, click on the + to add

new conditions as described in the Conditional properties section.

Property names for script access (intermediate):

The symbol control does not have any properties you can access programmatically, though you can [move and resize](#) the image using normal jQuery commands as described in the section on scripting. Use conditions to change the symbol displayed.

Trend Graph



The trend graph allows you to view a graph of your data over time. You can view a real time graph that scrolls as new data comes in, or scroll the graph back in time to view historical data.

Traces:

The graph supports multiple traces. Each trace has the same set of properties:

Expression: an expression, the result of which is graphed as the trace. Note that the result must be an array with time associated with it (usually from a tag), and since JavaScript doesn't directly support array math, if you want to do anything other than trend a tag, you will need to use the [math. functions](#) described later in this guide. Note also that `tagName()` returns an array of historical values based on the X scaling of the graph. In all other controls, `tagName()` returns a scalar with the most recent reading.

Line Width: the width of the line in pixels.

Label: if you assign a label to a trace, a legend will appear for the traces with labels.

Scaling:

If you double click on a graph while in run mode, a scaling box will appear. This allows you to adjust the scaling parameters and jump to any particular time. These settings are also available in the properties of the graph under the Scaling tab.

The Y scaling is pretty straight forward. You can select Auto for either max or min and the graph will set the Y scaling appropriately. If Auto is not checked, then the value specified will be used for the Y scaling.

For X scaling there are two options: if you have X Time Width checked, then the graph will always show the most recent data point at the right of the graph and will scroll as new data arrives. The amount of time displayed along the bottom axis is determined by the value to the right of the check box (in seconds).

If X Time Width is not checked, then the X Min and X Max settings are used to determine the scaling. Simply specify a start and end time to display. You can jump to any point in time that exists in your history. If you jump back in time it may take a little time for the graph to get data from our server to render. This is usually one update cycle. If the X axis scaling results in a lot data points coming from our servers, the data will be spread out over several updates and you will see chunks of data drawn as the new data arrives. Once data has been downloaded, it remains in memory until you quit, refresh your browser, or hasn't been accessed for a while. This means once the data is downloaded you can quickly zoom in and out without the initial delay. This applies across pages if you switch between pages. Refreshing the browser clears the data downloaded. Also, to avoid using too much memory on your system, data in memory that hasn't been accessed in a while is released and has to be re-downloaded when a new need for it arises.

Property names for script access (intermediate):

you can access the properties of this control as described in the [script section](#). Here are the properties for this control:

Series:

All series properties start with "series.x." where x is the series # starting with 0. If you only have one trace, it will be 0:

series.0.expression: a string containing the full tag name. Even though this is called "expression" it can only be a tag name

series.0.color : trace color. Use standard browser format, for example #FF0000 for red.

series.0.lineWidth


```
series.0.label
```

"series" is a property that contains an array. If you want to add new traces programmatically, you will have to retrieve the series property, then add to it manually, then set the property with the result.

Scaling:

```
xaxis.timeWidth
xaxis.useTimeWidth: set to true or false
xaxis.userMin
xaxis.userMax
```

```
yaxis.userMax
yaxis.userMin
yaxis.autoMin
yaxis.autoMax
```

Programmatically manipulating series (intermediate):

There are a number of functions available to allow you to manipulate series in a trend graph from script. This would allow you to, for example, create buttons to select which traces are displayed. As mentioned, all control functions start with "control." then the ID of the control, then (, then the command and parameters, then). To manipulate series there are 5 functions:

```
getSeries(index)
setSeries(index, newSeries)
addSeries(newSeries)
removeSeries(index)
removeAllSeries()
```

So, for example, to get the first series, you'd do:

```
control.myGraphId("getSeries", 0);
```

index in all cases can either be the series number, starting from 0 or the exact expression (tag name), so you could do:

```
control.myGraphId("getSeries", "myTag()");
```

The two remove functions are pretty self explanatory. Get will retrieve the details of a series. If you pass -1 as the index, you'll get the default series. You should usually use getSeries to get a template object to modify and pass back to setSeries or addSeries, though technically you don't have to. You can call set/add with partial details and the rest is filled in. For example, you can create a new series with all the defaults, display myTag by doing:

```
control.myGraphId("addSeries", {expression : "myTag()"});
```

However, its often helpful to do the full template, so it'd go something like:

```
var template = control.myGraphId("getSeries", -1);
template.expression = "myTag()";
template.color = "#FF0000";
control.myGraphId("addSeries", template);
```

It adds a copy of the template, so you can then modify the template again and add a new series:

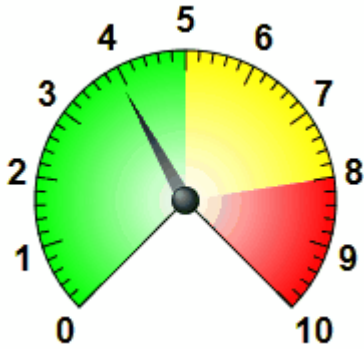
```
var template = control.myGraphId("getSeries", -1);
template.expression = "myTag()";
template.color = "#FF0000";
control.myGraphId("addSeries", template);
template.expression = "myOtherTag()";
template.color = "#00FF00";
control.myGraphId("addSeries", template);
```

Angular Gauge



An angular gauge is like the speedometer on a car. It displays a value by rotating a pointer around a central

axis:



Gauges are quite customizable and all sorts of gauges can be created. Many of the measurements used in customizing angular gauges is done in % units. This is percent of the maximum radius. The maximum radius is determined by the overall control size, changed like any other control by dragging the corners of the control. The maximum radius is simply 1/2 the minimum of the width and height of the control. By using percent, the proportions of the gauge will scale as the control is scaled. The exception to using percent is in line widths, which is specified in pixels, and font sizes.

Main:

Expression: the result determines the rotation of the pointer on the gauge. If the result is past the min/max of the gauge, it will simply be pegged.

Min / Max: the minimum and maximum values displayed on the gauge. If the pointer expression goes past these limits, it will simply "peg" at the limit.

Start Degrees: the angle where the minimum value will be displayed. 0 degrees is to the right of center, with positive degrees going clockwise around.

Range Degrees: the total sweep angle of the gauge. This can be 360 degrees. It can be more than 360 degrees even, though labels will not display past 360 degrees. This can also be negative, which causes the gauge to go counterclockwise instead of clockwise.

Style: with the exception of background color, this is for the text labels

Ticks:

Major ticks are split around the range of the gauge and also determine where labels are displayed. Minor ticks are displayed between each major tick.

Major Tick Count: the total number of ticks displayed. Note that since one is displayed at the beginning and end, you end up with 1 less area. So, for example, if your range is 0 to 10 and you want to display a tick every 1 unit, you would specify 11 ticks. If your range is ≥ 360 or ≤ -360 then any labels at or past the 360 mark are not displayed.

Minor Tick Count: the number of ticks displayed between each major tick.

Position: how far from the center the tick is displayed

Labels:

A label is placed for every major tick if shown. Position determines how far from the center the label is displayed. Note that unlike all the other elements of the gauge which are cropped to the overall size of the control, labels can appear outside the control's boundaries.

Area Colors:

Area colors determine the background color of the gauge area between the inner and outer arcs. You can have varying colors depicting certain ranges of values. You can have as many areas as you need, but each area should end at a value greater than the previous value.

Up to: since area color sizes are determined by an expression, you can actually make the area color change size based on a tag or other variable. Do not use a pointer (by leaving its expression blank) and this might make an interesting gauge type.

Property names for script access (intermediate):

you can access the properties of this control as described in the [script section](#). Here are the properties for this control:

pointers.0.expression: a string containing the expression for the pointer

rangeMin

rangeMax

startDegrees

rangeDegrees

mainStyle: the style used for the text and background. Please see the section on [scripted changing of styles](#) for more information.

showMajor

majorCount

majorPosition

majorLength

majorStrokeWidth

majorColor

showMinor

minorCount

minorPosition

minorLength

minorStrokeWidth

minorColor

showLabels

labelPrecision

labelPosition

Area Colors: "bkColors" is a property that contains an array. If you want to add new colors programmatically, you will have to retrieve the bkColors property, then add to it manually, then set the property with the result.

Otherwise, like series in trend graphs, you can access an already created color by specifying the index, starting with 0. So for the first color, it would be bkColors.0.:

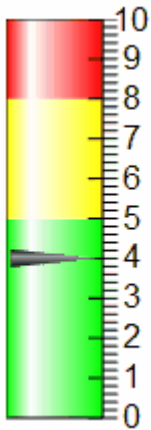
bkColors.0.color

bkColors.0.expression

Linear Gauge



A linear gauge is almost identical to the angular gauge, except instead of the pointer rotating, it moves up and down or side to side:



The properties of a linear gauge are pretty much identical to the angular gauge control except those that obviously don't apply. Measurements are done in %, just like the angular gauge, except they are a percentage of the width or height. Here's a summary of the ones that are specific to the linear gauge:

Main:

Horizontal: determines whether the gauge moves up and down, or side to side.

Text Button



The text button control creates an on screen square button with a text label. Although other controls, such as the static text, value display, and symbol can act like buttons, the text button control allows you to add special styling for when the mouse is over the button (hover), and when the button is pressed.

Like the static text control, the text label can have multiple lines. However, unlike static text (and most other controls), the size of the button is largely determined by the size of the text inside of it along with the specified margins. Resizing the control by dragging will not do much.

Style / Hover Style / Pressed Style

There are three groups of styles that are applied to the button depending on its state. The normal "Style" settings are when the button is active, but the mouse is not over the button, nor is the button pressed. "Hover Style" is applied when the mouse is over an active button, but the button is not pressed. "Pressed Style" is applied when the button is pressed (only when enabled).

The style options match up with the styles available for the static text control, though font type and size are the same for all three styles, as determined by the main style specification.

OnClick:

This is script you can enter that will execute when the button is clicked in run mode. Please see the chapter on [script](#) for more details.

Property names for script access (intermediate):

you can access the properties of this control as described in the [script section](#). Here are the properties for this control:

text: the text displayed

normalStyle: Please see the section on [scripted changing of styles](#) for more information.

hoverStyle

pressedStyle

Date Edit



The date edit control is an edit box with a date picker attached. You can enter dates into the control or select from a popup calendar by clicking on the calendar icon. You can then access that value from script. The properties for this control are largely about styling, which works essentially the same for all controls.

Property names for script access (intermediate):

you can access the properties of this control as described in the [script section](#). Here are the properties for this control:

currentDate: the currently entered date. Note that this in Javascript time format, namely milliseconds since 1970 and is for midnight on that date.

mainStyle: the style used. Please see the section on [scripted changing of styles](#) for more information.

Time Edit



The time edit control is similar to the date edit control. It is an edit box with a time picker attached. You can enter times into the control or select from a picker by clicking on the clock icon. Although the picker is limited to 15 minute increments, any value can be entered into the edit box itself. You can then access that value from script. The properties for this control are largely about styling, which works essentially the same for all controls.

Property names for script access (intermediate):

you can access the properties of this control as described in the [script section](#). Here are the properties for this control:

currentTime: the currently entered time. Note that this in milliseconds since midnight.

mainStyle: the style used. Please see the section on [scripted changing of styles](#) for more information.

Slider



The slider control is handle that rides on a track that you can drag to select a value from a range of values.

You can have either a horizontal or vertical slider. This is automatically set based on the dimensions of the control. Initially it will be a horizontal slider. To create a vertical slider, resize the control so the width is less than the height.

Min: the minimum value (left or bottom) of the slider

Max: the maximum value (right or top) of the slider

Step size: as you drag the slider, it will be limited to values that are an even multiple of the step size. Ideally, the step size should be divisible by the full range of the slider.

Change: script that runs when either handle is dragged or released. Because it is called as the handle is dragged, it is called many times so you should make sure your script is short so it executes fast, otherwise the control will be sluggish.

Property names for script access (intermediate):

you can access the properties of this control as described in the [script section](#). Here are the properties for this control:

stepSize

value: the currently selected value

min

max

Alarm Status

Status:	Name:	Fire time:	Reset time:	Ack time:
Fired	lowDemand	07/20/10 20:24:02		

The alarm status control displays all the current alarms marked "Display". The alarms are sorted and highlighted based on their state with fired alarms at the top. There are no properties for this control.

Alarm History

The alarm history control displays all the state changes for your alarms over the last 24 hours in a table. There are no properties for this control.

Custom Control



The Custom control is an advanced control that allows you to script your own custom control from within the DAQConnect editor. The rendering of the script is totally controlled by your script. The control is entirely made up of events. There are three:

Load: this event script is called when the control first loads. On the tab for editing the Load event there is a refresh button. If you Apply your changes then click the refresh button, it will execute your Load script. This is designed for testing purposes so you don't have to save your changes and refresh the browser every time you change your load event script.

Refresh: this event script is called whenever the page refreshes

Resize: this event script is called when the control is resized by dragging

All these events are called in the scope of the DOM `<div>` element that contains the control. Therefore, any additional rendering should be added to this element. When the control is first created, it is filled with a default element so it actually renders and is visible. Therefore, you will likely want to `empty()` the element in your load event first.

We strongly recommend using jQuery in your scripts. jQuery is already loaded and available for you to use.

An example will probably help:

Load:

```
$(this).empty();
$(this).append("<div>The time:</div>");
$(this).append("<div class = 'val'>---</div>");
```

Refresh:

```
$(this).find(".val").text(dateTime.systemTime());
```

The script above will create a control that displays the current system time in milliseconds since 1970 along with a caption. You could achieve basically the same thing (except having them on separate lines) using the variable value control, but this is just a simple example. There are more compact ways of achieving the same thing as well, but this gets the point.

In the first line of Load, we clear out any existing content: `$(this).empty()`

In the next two lines we add two `<div>` elements to our control. The second div element we're giving a class of

"val" so that we can access it from the Refresh event.

In Refresh, we find the element with the class of "val" and change its text to the current system time.

Technically we could empty and recreate the content with every refresh by moving the Load script into the refresh event, but its much more efficient to create content that doesn't change in Load, and then only update the required content in Refresh.

This sample creates a combo box with a list of pages (the sel.append() lines). When you select from the combo, you are switched to that page. Truthfully, the same template could be used to do a wide variety of things by changing the script in the sel.change() event. This script goes in the Load event of the Custom Control:

```
$(this).empty();
var sel = $("<select>");
sel.append($("<option value = 'tester'>My test page</option>"));
sel.append($("<option value = 'calibration'>Calibration Page</option>"));
sel.append($("<option value = 'OtherPage'>My Other page</option>"));
sel.change(function() {
    var newPage = $(this).find("option:selected");
    if (newPage.length > 0) {
        page.setCurrentPage(newPage.attr("value"));
    }
});
$(this).append(sel);
```

This is all done with jQuery. To explain:

```
$(this).empty();
```

clears the contents of the control. Always a good thing to do as the control will have default content so it is visible before you add script.

```
var sel = $("<select>");
```

creates our combo box. HTML uses the <select> element for combo boxes. We store the combo in a variable so we can work with it. We haven't put it on the page yet.

```
sel.append(...)
```

These lines add the various options that will appear in the combo. In HTML this is done by adding <option> tags inside a <select> element. The value = 'xxx' part is the name of the page we switch to when selected. The part between <option> and </option> is what is displayed in the combo. Change these two values and replicate as many sel.append() lines as you need options.

```
sel.change(...)
```

This creates an event that gets called whenever something is selected in the combo. Inside is what is called an anonymous function containing that event script. In that script:

```
var newPage = $(this).find("option:selected");
```

gets the element that is currently selected and puts it in the newPage variable.

```
if (newPage.length > 0)
```

makes sure that something is actually selected.

```
page.setCurrentPage(newPage.attr("value"));
```

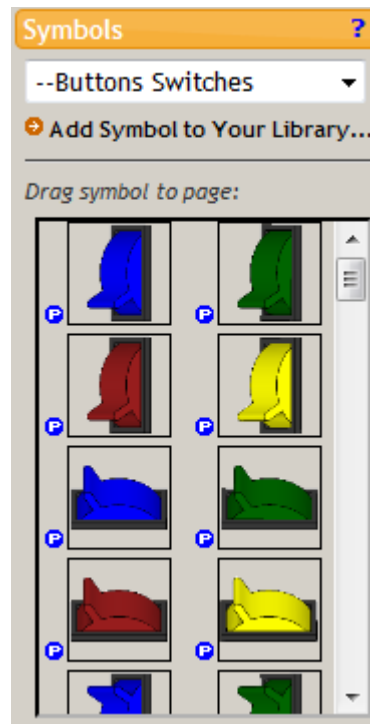
sets the current page based on the contents of the "value" attribute of the selected option.

Finally, we add the select combo box to our control:

```
$(this).append(sel);
```

Symbols

Overview



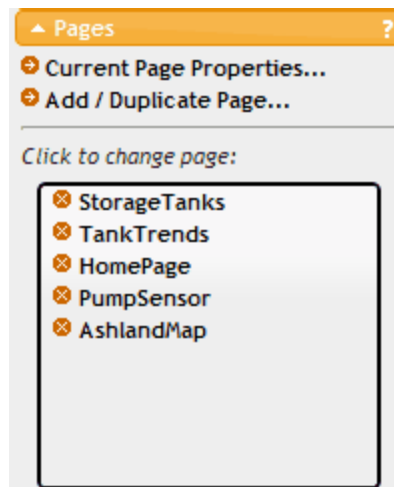
The symbol panel gives you quick and easy access to both the public symbol library and your own symbols. From here you can click and drag a symbol onto your page and a symbol control is automatically created with the symbol you dragged. The symbol control is automatically sized to the size of the image selected, but of course can be resized. Symbols can also be accessed from within the properties of the symbol control. This is the only way, in fact, to select multiple symbols for a single control that display.

At the top is a drop down menu allowing you to select from the public symbol categories, or your own symbol library.

Add Symbol to Your Library: click on this link to upload a symbol to your own symbol library. You can access all the symbols you upload from any of the pages you create. Symbols can be any JPEG, GIF, or PNG image. GIF and PNG images can be transparent. GIFs can be animated GIFs.

Pages

Overview



A page is a group of controls that are displayed together on the screen and provide the primary way to view your data. You can select which page you would like to view by clicking on the page in the list. You can also create screen controls that will change the page as well as described in [Page Variables](#).

Current Page Properties... : this will open a dialog for editing the page properties of the currently displayed page. If you are displaying multiple overlaid pages, the properties of the primary (first listed) page will be displayed. A description of the properties is in the [next section](#).

Add / Duplicate Page...: this allows you to create a new page. You can create as many pages as you need. You will be prompted for a page name which must start with a letter and contain only letters, numbers or the underscore. There are two buttons in the window: "create new page", which will create a new blank page, and "duplicate current page(s)" which will create a new page, but copy all the currently displayed controls to that page.

Page Properties

Pages have properties like controls:

Background: by default a page has a white background. You can change this color, or even have the color change automatically based on an expression. Please see the section on Conditional properties for more information.

Scroll page?: normally a page only shows what will fit within the browser window. If this is checked, then the page "canvas" can be larger than the window and scroll bars will be displayed. The size of the canvas is determined by the next two parameters:

Page Width / Page Height: determines the size of the page canvas when Scroll page? is checked.

External?: if checked, then the page can be viewed without logging into DAQConnect. This allows you to embed pages on other websites, or give public access to some of your pages. Please see the section on [Embedding pages](#) for more information.

ID: the page ID can be used in the URL to jump to a particular page, and in embedded pages. Please see the section on [Embedding pages](#) for more information.

Load: event script that is called when the page is first loaded. Pages are only loaded once they are first accessed. This means that OnLoad may not be called when you first enter the application.

Show: event script that is called when a page is shown. Note that if you are using overlaid pages as described below, and this page remains visible while an overlaid page changes, both the OnShow and OnHide events for this page will still be called.

Refresh: event script that is called before the page is redrawn. The page is redrawn typically once a second, though other events can trigger a refresh. Since this event occurs often, make sure your script is fast.

Hide: event script called when the page is switched.

Page Variables

Pages also have internal variables and functions. All of them start with "page." Note it is case sensitive!

page.setCurrentPage(newpage): call this function to change pages. Pass in a string containing a comma delimited list of the pages you wish to display. For example, to change to page2:

```
page.setCurrentPage("page2")
```

Normally the string will just contain a single page name, but you can specify multiple pages and create overlaid pages. Overlaid pages allow you to create a single page with common content (such as a menu, or overview) and overlay other pages on this common content. To overlay two pages, just set this variable to a comma delimited list of the two pages:

```
page.setCurrentPage("page1, page2")
```

When you have overlaid pages and switch to edit mode, there are a couple important points: You can drag any existing control and it will remain on its original page, however, if you add a new control, either by dragging it onto the page, or duplicating an existing control (no matter what page it was originally on), the new control will always be on the first page listed (i.e. page1 in the example above).

Because of this, if you are creating a page with common content, we recommend listing it last so that you can create and edit the other pages while viewing the common content.

Embedding pages

If a page is marked External from the page properties, anyone can access the page without having an account. There are two ways this can be done:

Direct access:

You can jump directly to a particular page by putting the page ID in the URL. For example:

```
www.daqconnect.com/daq/page.do?id=20BAC465-94B5-4A0E-866A-9F64DC3826C2
```

where 20BAC465-94B5-4A0E-866A-9F64DC3826C2 is the page ID. Replace "serverAddress" with the URL for your particular server which you can see in the address bar of your browser. These vary depending on which server your account is placed on. If a page is external, anyone can access the page this way. So, you could create a link from your regular website to jump directly to a page for public viewing. Anyone accessing it will not be able to edit the page, nor switch pages unless you provide controls to change pages.

Embedded page:

You can also embed a page or part of a page on another website. To do so, you need to add just a couple lines to your web page:

In your <head> block, preferably after all your stylesheets, put:

```
<script type="text/javascript" src="http://www.daqconnect.com/daq/dc.do?id=ABCDAAAB9-2BB1-44EC-8C54-7E71514A1F7B">
```

Replace the id (ABCDAAAB9-2BB1-44EC-8C54-7E71514A1F7B) with the id for your page.

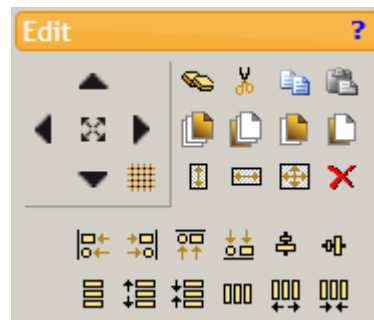
In your <body> block, create a <div> wherever you want the page. Size and style the <div> as you desire. Note that you MUST provide a size for the div. It won't automatically size because the controls don't affect the flow of the html. You will need to give the div the unique id 'dcul-mainContent'. Leave the div empty unless you want content underneath. The div looks like:

```
<div id="dcui-mainContent" class = "hp_dc" style = "width: 800px; height: 800px;"></div>
```

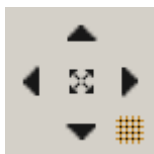
Note that you can only embed one page per web page. The embedded pages, however, are fully functioning in run mode, and so, for example, Click events of controls will function inside an embedded page.



Edit

Overview



Once you have controls on your pages, you will likely want to be able to manipulate them to perfect your screens. First you need to select the control. To do this, simply click on the control while in edit mode. To select more than one control, hold down the Ctrl key while clicking on each control you want to select. Once selected, you can move controls by clicking on the selected control(s)'s header and dragging them. You can resize a control by clicking along the edges and dragging. The Edit menu gives you more options:



These buttons allow you to move and resize your control(s) in small steps.  determines the grid size that controls are snapped to and the number of pixels the nudge/resize arrows will adjust the controls. If the resize button:  is selected, the arrows will resize the selected control(s) based on the current grid size. If it is not selected, the arrows will move the control(s).



Duplicates the currently selected control(s), shifting the new control(s) down by the height of the control.



Cuts the currently selected control(s) and places them on the application clipboard. For security reasons, the controls are not put on your computer's clipboard, so are only available within this browser window.



Copies the currently selected control(s) and places them on the application clipboard.



Pastes the contents of the application clipboard to the current page.



These four buttons allow you to arrange the order of your controls on the canvas. Controls in front can cover controls behind.



Deletes the currently selected control(s).



These three buttons will make all the selected controls the same size as the first selected control of the group.



These six buttons allow you to align your controls to the first selected control of the group.

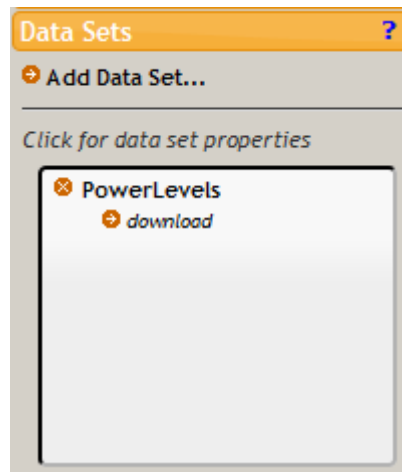


These six buttons allow you to space your controls. The ones without arrows will space the controls evenly, while the ones with arrows will increase or decrease that spacing.

Data

Data Sets

Overview



A data set is a group of tags that you can download to your local machine for archiving or further analysis. When you trigger the download you will be prompted for a date range and time interval. Data is then placed on the desired time interval and returned as either a CSV or JSON file. CSV, or "Comma Delimited Values" files are almost universal in their ability to be imported into other programs and is probably your preferred format. JSON is a web standard format. To download data, you will first need to create a data set, then go to the data source properties windows for your desired tags and select the tags for your data sets as described in the [data source properties](#) section. You can include tags from multiple data sources in a single data set.

Add Data Set: click to add a new data set to your account. You will be prompted for a data set name, which must start with a letter and contain only letters, numbers or the underscore. Once you have created a data set you can add tags to it by going to the data source properties for the desired tags.

Data set list: this is a list of data sets you have created. Click on the data set name to display the data set properties window. Click on the X to delete a data set. Click on download to download the data set. A window will appear requesting the date range, time interval and data format.

Data Set Properties

Data Set Properties

Data set: WaterUsage, ID: B1D112F3-6351-491B-9507-7E0BE9EE62A7

Select: [All](#) [None](#) [Delete Tag\(s\) from Set](#) [Save Changes to Ordering](#)

Drag tags to rearrange the order they show up in the data set then click 'Save' to save the changes.

- ☐ Remote - alsingdemand
- ☐ Remote - alsingflowin
- ☐ Remote - alsingflowout
- ☐ Remote - alsinglevel
- ☐ Remote - crowsondemand
- ☐ Remote - crowsonflowin
- ☐ Remote - crowsonflowout
- ☐ Remote - crowsonflowoutalsing
- ☐ Remote - crowsonflowoutgranite
- ☐ Remote - crowsonlevel

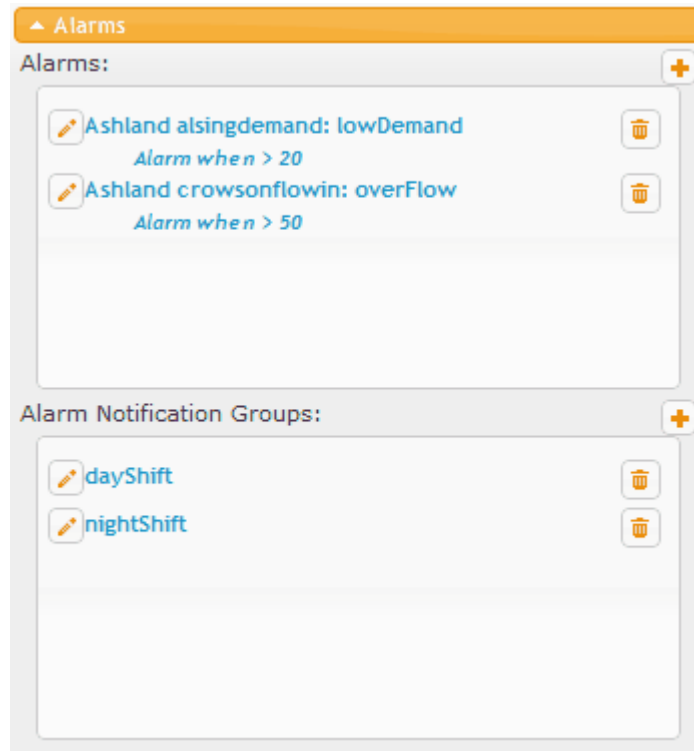
The data set properties window allows you to rearrange the ordering of your tags within the data set and quickly remove tags from the data set. You can also remove tags from the data set from the data source properties window. To rearrange the ordering, simply click and drag the desired tag to its new location. When complete, click the "Save Changes to Ordering" button.

Alarms

Overview

With our alarm feature, you can have the system monitor your tag values and if they go over, under or equal a particular value, email you an alert. You can also display currently active alarms on your pages using the [alarm status](#) and [alarm history](#) controls. You can acknowledge them from the page as well from a button. Each tag can

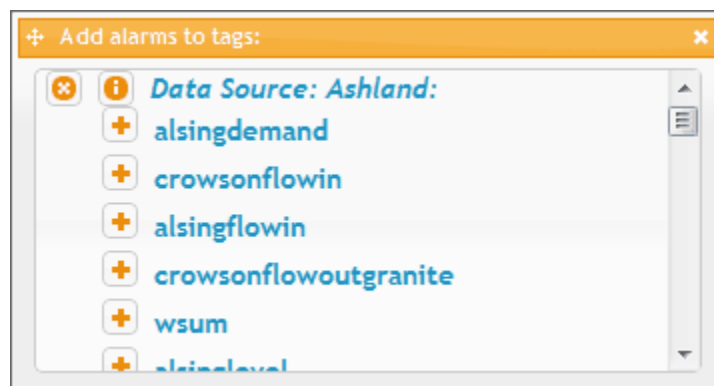
have multiple alarms. You can then create notification groups which are a group of email addresses that get notified when certain alarms occur or are reset. Notifications to a particular alarm group can be limited to certain times of day, allowing you to notify just the on-shift people.



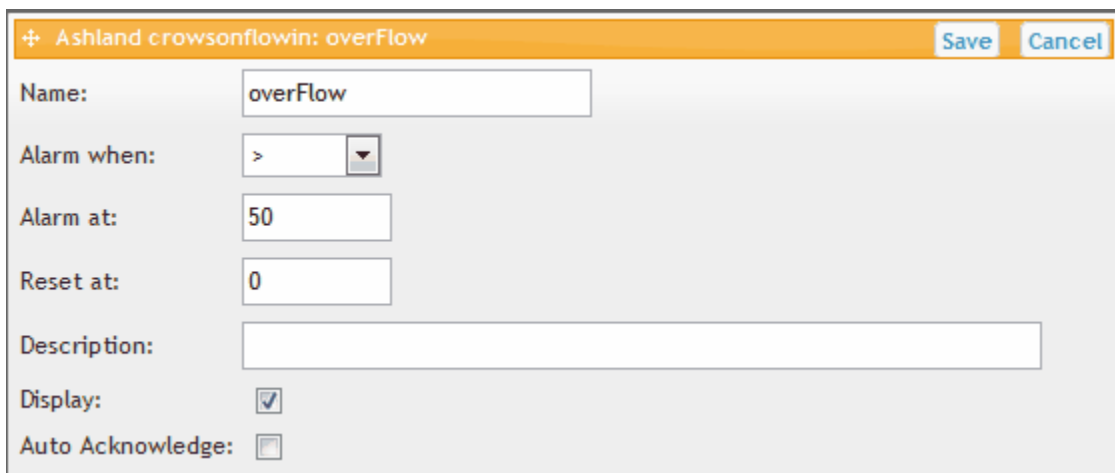
The main alarm panel lists existing alarms and alarm groups, allowing you to edit them and delete them. To add a new alarm or alarm group, click on the + button to the right.

Editing Alarms

To create a new alarm, click on the + to the right side of the alarm panel. This will display a list of tags similar to the list in the Basics menu:



To add an alarm to a particular tag, click on the + next to the desired tag. You can add as many alarms as you need for any tag. This brings up the alarm edit window. You can get to this window for an existing alarm by clicking the pencil icon next to the alarm listed in the alarm panel.



Name: the short name for the alarm. It is used in the alarm display controls, so it should be somewhat descriptive, however not long as the columns in the alarm display controls are not very wide.

Alarm when: select a desired operator that is used with your tag value and the Alarm at condition. So, in the screenshot above, the alarm would occur when the tag crowsonflowin is greater than 50. The alarm would reset when the tag goes below, or is equal to 0.

Alarm at: the value that the tag value is compared to that determines when the alarm fires. Once the alarm fires, it will not fire again until the reset condition has been achieved.

Reset at: the value that the tag value is compared to that determines when the alarm resets. This only applies when the alarm is in the fired state.

Description: a long description of the alarm. Put whatever you need here.

Display: if checked, then the alarm appears in the alarm status and history controls. If not checked, it does not, but the alarm still will send notifications.

Auto Acknowledge: if checked, then an alarm that has reset is automatically acknowledged. If not checked, then you have to manually acknowledge alarms. Until an alarm is acknowledged, it will remain towards the top and highlighted in the alarm status control.

Data Age Greater Than

There is one special Alarm type called "Data age when greater than" that you can select for the Alarm when property. If this is selected then the alarm will fire when the server doesn't receive any new data for that data point in the specified number of seconds. This is a useful alarm to ensure that your remote data source connections are still active. One important point: this is only evaluated every 30 seconds so your Alarm at and Reset at values must be greater than or equal to 30.

Editing Alarm Groups

An alarm group is a set of contacts (email addresses) that will get notified if one of the alarms in the group fires or resets during the active times of the group. To add or edit an alarm group, click on the + or pencil icon. This will display the alarm edit window:

The screenshot shows a configuration window for an alarm group named 'dayShift'. At the top, there is a title bar with a plus icon, the name 'dayShift', and 'Save' and 'Cancel' buttons. Below the title bar, there is a text field for 'Name:' containing 'dayShift'. Underneath is a section for 'Alarm Group Members:' with a plus icon to add more. This section contains a single member named 'me' with a checkmark, a trash icon, and an info icon. Below this is an 'Alarms:' section containing two items: 'Tag alsingdemand: lowDemand' and 'Tag crowsonflowin: overFlow', both with checkmarks and info icons. At the bottom, there is a 'Notification Type:' section with 'Immediate' selected and 'Active Times' as an option.

In this window are two list boxes. The top box displays all the alarm contacts you have created. Click on the + to add new contacts, the trash can to delete a contact, and the info button to edit an existing contact. A contact is simply a name and an email address. If you have a single person with multiple emails, simply create multiple contacts giving each a slightly different name, like Joe-Home, Joe-Mobile, etc. The bottom box displays a list of all the alarms you have in your account.

First, give you group a name. Then select by clicking on all the contacts and all the alarms you want in the group. If you accidentally click on a contact or alarm that you don't want in the group, just click on it again to unselect it. Finally, click on the Active Times button to open the selection window for when this group can receive notifications. The default is 24x7:

Active Times

Time zone offset (hours from UTC):
(UTC -8:00) Pacific Time (US & Canada)

☒ Sun: 00:00 to 24:00 Invert? ☐

☐

☒ Mon: 00:00 to 24:00 Invert? ☐

☐

☒ Tue: 00:00 to 24:00 Invert? ☐

☐

☒ Wed: 00:00 to 24:00 Invert? ☐

☐

☒ Thu: 00:00 to 24:00 Invert? ☐

☐

☒ Fri: 00:00 to 24:00 Invert? ☐

☐

☒ Sat: 00:00 to 24:00 Invert? ☐

☐

Simply drag the sliders to select different time ranges. To deactivate an entire day, uncheck the box next to the days name. Use Invert? to flip when the group is active during that day. So if you have 9am to 5pm selected for Sunday and click Invert, it will make it so the alarm is active from midnight to 9am then from 5pm to midnight.

Utilities

Utilities

Overview

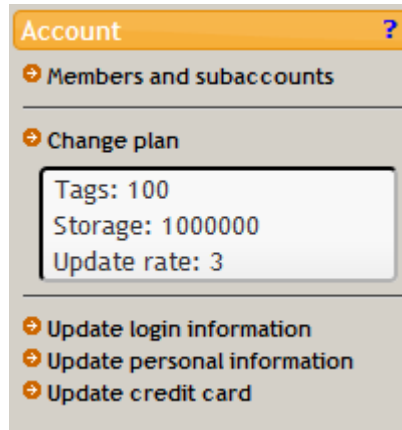
This panel includes several useful utilities:

Download pages: this will download all your pages and associated symbols to a zip file. You can use this for backup, or to copy your pages from one account to another.

Upload pages: this will upload pages previously downloaded into the current account, adding to existing pages and symbols.

Account

Overview



This menu allows you to manage your DAQConnect account.

Members and subaccounts: this will open a new window allowing you to manage any additional members and subaccounts associated with your account. A member of your account is another person with a different account login who can access your account. Members can either be Managers or Restricted members. Managers can make edits to your pages, add symbols, data sources and data sets, but has no access to billing information. Restricted members only have access to view pages. When you create a Restricted member, you will select the page that is displayed when they login. Since Restricted members have no access to the menu, you will need to provide screen controls to change between pages.

Subaccounts provide a way for system integrators, and those requiring multiple accounts billed under a single entity, to create new accounts that are billed to the main, root account. You can then create members of the subaccount that either can do page design, or that simply view pages you create for them. None of the members of the subaccount will have access to billing or plan information, allowing you to package the cost with other services, or simply consolidate your billing. Once you have created a subaccount, you can access it either by using the login associated with the subaccount, or by clicking on the subaccount from the Members and subaccounts window. If you switch to a subaccount from the Members and subaccounts window, you can return to the master account by clicking Logout. This will log you out of the subaccount and return you to the master account.

Change plan: this allows you to change the limits on your plan. You will need to arrange payment before the change is completed.

Update login information: this allows you to change your login password.

Update personal information: this allows you to update your mailing address.

Control capability

IMPORTANT NOTE: the architecture of our system keeps your SCADA system secure by having data push out the SCADA system so that the SCADA system can remain securely behind firewalls. Once you add control capability, you are opening up the SCADA system to possible hackers, though the only things they'd have access to are things you have control capability enabled for. For this reason, you should only enable control capability on items where random settings will not cause any injury or property damage. This can be done reasonably effectively by adding constraints in the SCADA system and of course the proper hardware safety systems. However, when designing a system with control capability, you should always assume that someone with malicious intent may get access and send random values to your keys. Our system is secure, however, the weakest link is you, your users and your passwords. Even the most secure password can be compromised by a key capture device on a public terminal, or a virus on your system. The best defense is to assume the worse and keep the mission critical items on local control only.

In addition to viewing data arriving from your data sources, you can send commands back to the data source to

control outputs, acknowledge alarms, or whatever else you may setup. The capabilities are largely determined by the data source itself. The application sends this commands with a simple line of script, typically placed in the Click event of a control:

```
system.sendCommand("dataSource.key","value")
```

dataSource.key is the name of the parameter / command you want to send to the data source. The dataSource portion should be the name of the data source to receive the command. The key is any name. It could be a tag name if you are controlling an output tag, or it could be the name of an alarm to acknowledge, or anything else. Our application doesn't do anything with it other than pass it to the data source, so its up to the data source to interpret the key.

value is the value to go along with the command. So, it could be the value to set the output channel to, or anything else.

NOTE: the dataSource.key pair MUST always be in quotes. The value can be in quotes, or if you are just sending a number, it can also be without quotes. For advanced users, you can of course use string variables instead of these string constants.

Expressions



Expressions

Overview

Most controls use expressions to determine what is displayed. They are simply Javascript formula, and as such you can refer to any Javascript book or the Internet for additional help. If you've used Excel or other programs that take formulas, expressions should be pretty easy for you. If not, it's really not that much harder than using a calculator.

In addition to all the standard Javascript functionality, we have added a number of functions and variables you can use to make expressions and script easier for you.

Advanced: expressions and script all run inside a sandbox, partially to make it easier to work with, and partially to protect the application from accidental mis-coding. Because of this, you can't use most of the functions and variables shown from any Javascript debugging environment because that environment is running outside the sandbox. However, the sandbox is a global object which you can access called `$.dc.sandbox`. So, for example, if you wanted to access the math functions, you could: `$.dc.sandbox.math.sin(x)`, or to access global variables: `$.dc.sandbox.global.myVariable`.

Note that tags are not part of the sandbox. This, again, is to protect the application. If you want to access tag data from outside our environment, you should use `$.dc.tagValue.dataSourceName.tagName`. However, this is giving you direct access to the functions so if you accidentally assign values to these variables, you will break the environment and have to refresh.

Referencing tags

If all you want to do is display the value of a tag, you can simply click on the down arrow to the right of the expression box and select the data source and the corresponding tag and it will be placed in the expression box for you. If you need to do extra math, or are using script, you'll need to enter the tag names yourself. Tags are always specified in the following format:

```
dataSourceName.tagName()
```

where `dataSourceName` is the name of the data source the tag is on, and `tagName` is the name of the tag. So, for example:

```
remote.alsingDemand()
```

Using this format returns the most recent reading for the given tag, unless you are specifying a trace in a trend graph, in which case you get all the historical values that are within the X axis range in an array.

Sometimes, however, you want the last 5 readings, perhaps to take the average of those readings. In this case, simply list the indexes of the desired range in parenthesis after the tag name:

```
remote.alsingDemand(0,4)
```

0 is always the most recent data point, so 0,4 returns the last 5 readings. You can also select by time, so to get the data from the last hour you can put:

```
remote.alsingDemand(dateTime.systemTime(), dateTime.systemTime() - 3600000)
```

`dateTime.systemTime()` is a function that returns the current system time in Javascript format, which is milliseconds since 1970. The only problem with `dateTime.systemTime()` is that it returns the time of the local browser, which may not be in sync with the data stream. Instead you probably want to reference the time of the data point. To do this, add `.time` after the tag name. So, the time of the most recent data point is, for example:

```
remote.alsingDemand.time()
```

or the last 5 data points:

```
remote.alsingDemand.time(0,4)
```

General Math Functions and Operators

Since expressions and script are all Javascript, you can use any of the standard operators and Math functions available when working with scalars (i.e. the most recent value of a tag). Most are pretty standard, like +, -, *, /, >, < etc. A few non-standard ones:

== for comparison, not =, which is assignment

Math.pow() for power, not ^ used in some other languages

Most math functions, like sin and cos, are part of the Javascript Math object, so its Math.sin() not just sin().

The custom math object:

Often you'll want to work with arrays of data since historical data is stored in arrays. Javascript doesn't offer math functions that work on arrays, so we've provided them for you along with some extra useful functions. These are all contained in the math object (note lowercase). You can use math. wherever you'd use the standard Math. and the functions will work on arrays, so math.sin() instead of Math.sin. In fact if you just get used to using "math" instead of "Math" it will work on both scalars and arrays. We've also added functions for the standard operators:

```
plus()
minus()
times()
divide()
not()
lessThan()
greaterThan()
greaterThanEqual()
lessThanEqual()
equal()
notEqual()
bitNot()
bitAnd()
bitOr()
xor()
shiftRight()
shiftLeft()
and()
or()
mod()
```

Extra handy functions:

random(x): returns a random number, or if x is specified, an array of random numbers with length = x

min(x, y) / max(x, y): if you pass two arguments, it works the same as Math.max(). If you only pass x, it assumes x is an array and gives you the min or max of that array.

sum(x): returns the sum of all the elements in x

mean(x): returns the mean of all elements in x

delta(x): takes an array, and returns an array that contains the difference between each subsequent array element. This is most useful when your tag is a counter, such as the output on many energy meters, and you want the difference between consecutive counts.

Date and Time functions

Date and time are always important in data acquisition applications. As such we give you a few handy functions. All of them start with "dateTime."

`dateTime.dateFormat`: a variable that can contain either "mdy" or "dmy". The default is "mdy". Changing it to "dmy" will change how date values are displayed across the application.

`dateTime.systemTime()` : returns the current browser system time in Javascript time format, milliseconds since 1970. It is basically the equivalent of: `(new Date).getTime()` in Javascript.

`dateTime.printDate(val)`: prints the specified date value in either m/d/y or d/m/y format.

`dateTime.printTime(val, ms, sec)`: prints the specified date value's time portion. By default, seconds and milliseconds are displayed. Set ms and/or sec to false to turn them off.

`dateTime.printDateTime(val, ms)`: prints both the date and time for the specified value. If ms is set to false, milliseconds are not displayed.

Advanced: you can also access the `dateTime` object from global Javascript using `$.dc.dateTime`. `dateTime` in the sandbox is simply a reference to the global object.

Script



Script

Script Overview

Most controls, as well as pages and other items allow you to enter script for various events, such as when a page loads, or when you click on a control. With this script you can set variables, and do calculations and comparisons. This script is standard browser Javascript, and as such offers a lot of flexibility. Here we'll talk about some of the unique features available from script and some common tasks. In general, though, you should refer to your favorite Javascript text book or the Internet on how to do basic Javascript scripting.

Because browser Javascript is not multithreaded, scripts should be kept relatively short to avoid making your browser sluggish.

Variables

Like regular Javascript, you can create local variables to your script using the var statement:

```
var x = 3
```

You can then reference them by name, "x". These variables will go out of scope and thus disappear when the script completes. To create a global variable that is accessible between scripts and script executions, use "global":

```
global.x = 3
```

Reference global variables using global. as well: "global.x"

Advanced: because all script is run in a sandbox, it is impossible to create a true, system global variable. You can access them (for example: window.location), you just can't create them. You can access script global variables from outside the sandbox by accessing the sandbox: \$.dc.sandbox.global.x for example. "global" is simply an empty object declared in \$.dc.sandbox and doing global.x = 3 simply adds a member variable to the global object.

Accessing on-screen controls from script

All on-screen controls that you place on pages can be accessed programmatically from script. This allows you to retrieve values, for example, from a slider, or set properties. This is done by typing in "control." followed by the ID of the control. This is a function, so you'll pass in values to the function to retrieve, make changes, and call any member functions of the control. For example, if you have a variable value control with an ID of "myVal", you'd put:

```
control.myVal("option","caption","new caption")
```

to change the caption to "new caption". All properties are accessed by passing "option" as the first parameter, and the property name as the second parameter. If a third parameter is specified, then the property is changed to that value. If not, then the property is retrieved instead, so:

```
control.myVal("option","caption")
```

would return "new caption".

Please refer to the sections on the controls to see what properties are available and what their names are. Remember, Javascript is case sensitive!

Advanced: all controls are actually jQuery UI objects and thus follow the same pattern, with some exceptions. control.controlID() is almost the same as \$("#controlID").controlType() except that we allow you to change just parts of properties that are objects without having to set the entire object. The same goes for arrays. You can access controls through \$() notation, however you'd have to know our the full UI type for the control and our additions won't apply and you'll have to fully specify objects and arrays. If you are going to access properties from outside, we recommend going through the sandbox: \$.dc.sandbox.control.controlID() unless you are just going to use standard jQuery functions on it. That said, if you want to use normal jQuery functions like css() to do things like move the control, you will have to use the jQuery \$() notation as described in the next section.

Moving and resizing from script

Because all on screen controls are jQuery UI elements with the ID set, you can use standard jQuery commands on them. This means you can easily move and resize them from script. For example, to move a control with an ID of "myVal" to the top left by doing:

```
$("#myVal").css({left : 0, top : 0})
```

or change its width and height to 100:

```
$("#myVal").css({width : 100, height : 100})
```

Changing styles from script

Most objects have some sort of style object. To programmatically change the style, you should pass an object with all the parameters you want changed. For example, to change the "mainStyle" property so that the font size is 22, you would do:

```
control.myVal("option", "mainStyle", {fontSize : "22px"})
```

You can set multiple parts in a single call. For example to also set it to bold:

```
control.myVal("option", "mainStyle", {fontSize : "22px", fontWeight : "bold"})
```

The parameters are normal Javascript names for CSS properties and include:

```
fontFamily
fontSize
fontColor
backgroundColor
textAlign
fontWeight
fontStyle
textDecoration
paddingLeft
paddingRight
paddingTop
paddingBottom
borderStyle
borderWidth
borderColor
```

You can actually use any of the others, however, the standard properties window for the control may reset some of them, so typically you should either programmatically change style always, or never.

Note that this does not work with conditional styling.

Downloading data from script

To download data from a button press, simply call the downloadData function:

```
$.dc.dataSource.downloadData(data source name, options)
```

Options can be a whole bunch of stuff, but if you leave it empty and just do, for example:

```
$.dc.dataSource.downloadData("Belleville")
```

it will prompt the user for many of the items, including which tags to download.

Options is a javascript object, with these defaults:

```
{
    filename : dataSourceName + "_data.csv",
    columns : [],
    tags : [],
    tagsToDisplay : null,
    start : $.dc.member.getCurrentServerTime() - 86400000,
    end : $.dc.member.getCurrentServerTime(),
    format : "csv",
    interval : 60000,
    ignoreEmpty : true,
    promptMember : true
};
```

Probably a little confusing. First an example, let's say you want to display a different default file name and interval:

```
$.dc.dataSource.downloadData("Belleville", {filename : "myfile.csv", interval : 120000})
```

Here's a description of the options:

filename: the default file name displayed, or the actual final file name if promptMember is false

columns: an array specifying exactly what to download. Should only be used if promptMember is false. Takes an array of objects in the form: {tag : "xxx", label : "yyy"}. Use this when you want to create a button that downloads the same set of tags every time and you don't want to ask the user. Using this is a nice replacement for a dataset

tags: an array of string specifying the tags to download. This is just a slimmer version of columns. Use one or the other. Using columns gives you control over labels. This uses the tag name as the label.

tagsToDisplay: an array of strings listing (case sensitive) the tags you'd like the user to see when the popup arrives. Leave null to list all tags in the data source

start/end: the start and end of the time range to download, in ms. Defaults to the last day.

format: presently always "csv"

interval: the spacing of the data, in milliseconds. Defaults to 60 seconds.

ignoreEmpty: if true, then a row is only created if there is data in one of the columns. If false, then a row is created at every interval between start and end

promptMember: if true, then a popup asking for the tags, range, interval, etc is displayed. Otherwise it uses the above settings. Make sure tags or columns is specified if this is false. Make sure they aren't specified if true.

Remote Connectors



Remote Connectors

RESTful interface

For those who wish to send data to DAQConnect from their own code and not use the Windows DLL we offer a simple RESTful interface that allows posting data to DAQConnect using simple HTTP GET (or POST) requests. This makes it easy to send data from just about any source, no matter what the operating system or programming language.

The URL for the RESTful is:

`https://www.daqconnect.com/daq/postData.do`

You can use POST or GET. GET has a length limit (typically 1024 characters), POST does not. Although not as secure, you can use `http://` as well.

There are two ways of sending data:

Data for a single tag

The URL parameters for a single data point are:

1. `c=<Connector UUID>`
2. `t=<Tag Name>`
3. `d=<time>:<value>` (this parameter can be repeated as many times as necessary)

NOTE: Time should always be in UTC (coordinated universal time). This allows your DAQConnect pages to display data in the local time zone.

An example HTTP GET that sends a single data point would be:

`https://www.daqconnect.com/daq/postData.do?c=12345678-1234-1234-1234-123456789012&t=testTag&d=1257449020.855:103.941`

To send multiple data points for a single tag the HTTP GET would look like:

`https://www.daqconnect.com/daq/postData.do?c=12345678-1234-1234-1234-123456789012&t=testTag&d=1257449020.855:103.941&d=1257449021.855:104.95`

An example of an HTTP POST payload for the previous request would look like:

```
c=12345678-1234-1234-1234-123456789012
&t=testTag
&d=1257449020.855:103.941&d=1257449021.855:104.95
```

Data for multiple tags

To send data for multiple tags add these URL parameters

1. `c=<Connector UUID>`
2. `t=<Tag Name>:<time>:<value>` (this can be repeated as many times as necessary)

NOTE: Time should always be in UTC (coordinated universal time). This allows your DAQConnect pages to display data in the local time zone.

An example HTTP GET that sends two tags:

`https://www.daqconnect.com/daq/postData.do?c=12345678-1234-1234-1234-123456789012&t=testTag:1257449020.855:103.941&t=testTag:1257449021.855:104.95`

Return payload

A simple JSON packet is returned that contains the status of the operation: `{ "status" : result }`. Where result is:

0 : Data has been received and stored

2 : The customer account is expired. Please log in and update your account.

3 : The connector UUID is invalid. It may have been deleted from the account or mistyped in the URL.

5 : Sending data faster than the account is permitted to send data.

6 : No more room to store any data in separate tags. Delete old tags or update your account to add more tag storage.

7 : No more room to store extra data. Delete old tags or update your account to add more storage.

HTTP Content-Type Header

The HTTP content type header for a POST request must be: application/x-www-form-urlencoded

Adding Control Capability

If you want to allow control capability through DAQConnect, that is, having DAQConnect send messages to your device, you should use a slightly different API. The data sent is exactly the same, except the URL is different. It becomes:

<https://www.daqconnect.com/daq/postDataControl.do>

The response is what is different. It looks something like this:

```
{
  "status" : status-code,
  "controls" : {
    "data" : []
  }
}
```

If there are no control values for the connector then the "controls" object will not exist, for example:

```
{ "status" : status-code }
```

If there are control values then the message will look like this:

```
{
  "status" : status-code,
  "controls" : {
    "data" : [ [ key1, value1 ], [ key2, value2 ] ]
  }
}
```

"key" is a string containing whatever key was specified using the Set() command in DAQConnect. Note that the connector information is stripped from the key, so "remote.mykey" becomes just "mykey". The connector part is used to determine who gets the response. "Value" is also a string, but could contain a numeric value, i.e. "3.412", and is simply the value parameter of the Set() command.

Windows DLL

With the Windows DLL you can communicate with the DAQConnect servers from your own application, or from any application that can link to a Windows DLL. This section gives a brief description of the DLL's API. It is assumed that you know how to link to a DLL.

To DAQConnector DLL distribution contains three files:

DAQConnector.dll: the actual DLL that is loaded at runtime and does all the work.

DAQConnector.lib: a lib file for those that want to link to the DLL at compile time.

DAQConnector.h: the header file that lists the prototypes for the API.

There are only a few functions needed to communicate with DAQConnect:

```
short DAQConn_Initialize(const char *ConnectorID)
```

Call this function once with the Connector ID you are using. The Connector ID is the one listed in your My Account page. This function returns 0 if it is successful and non-zero if there is an error. You can get errors using the DAQConn_GetErrors() function described below.

```
short DAQConn_AddValue(const char *TagName, double val, double time, long maxstorage)
```

This is the primary function for sending data to DAQConnect. It should only be called after you have successfully initialized the connection. You should call this function once for each data point you want sent. TagName is the name of the Tag, val is the value to be sent. time is the timestamp (in seconds since 1970, UTC time). maxstorage is the number of historical data points to keep for this tag. You should always pass the same value for a particular tag otherwise any accumulated historical data is lost. You can pass -1 to have DAQConnect automatically configure the history based on the account plan.

This function does not actually send the data to DAQConnect. Instead it queues the data until the next interval. That interval is determined by your plan. An internal thread, created when you successfully initialize a connection, handles sending the data. The function will only fail if you try and send data that is spaced too closely together in time.

NOTE: if you lose your Internet connection or otherwise have troubles sending data to DAQConnect, the data will queue up until a connection can be reestablished. There is a hard limit of 10,000 data points per packet across all tags to ensure you don't run out of RAM. If more than 10,000 data points queue up while waiting for a connection, the packet will be cleared and all accumulated data will be lost. New data will then reaccumulate until the connection is established or the process repeats.

```
short DAQConn_GetErrors(char *Message, long MaxLen)
```

Since the sending of data runs asynchronously, you must call this function to retrieve any errors that have occurred. Errors are essentially stored in a FIFO buffer internally. Each call to DAQConn_GetErrors() returns the oldest error and clears it from the FIFO. If an error is returned, the function will return 1. You should then repetitively call this function until it returns 0. Message should be a preallocate buffer. MaxLen is the length of this buffer. Make sure and save an extra byte for the \0 character.

```
short DAQConn_SetStatusCallback(tStatusCallback fCallback, void *pData)
```

For advanced users wanting to provide more detailed debugging information, you can have the DAQConnector DLL perform a callback function for various status events. fCallback is a function pointer for a function with the tStatusCallback form shown in the header. pData is a void pointer to whatever data you want. This data is then also passed to the callback function when it is called. Note that since your callback will most often be called by a secondary thread, you MUST ensure that your callback function is fast and doesn't call any Windows API calls, including updating screen controls.

Control capability:

If you would like DAQConnect to be able to send messages to your application such as controlling outputs, you will need to create another callback function and then tell the DLL about it using this function:

```
short DAQConn_SetControlCallback(tControlCallback fCallback, void *pData)
```

fCallback is the callback function which should have this prototype: void foo(const char *key, const char *value, void *pData). pData is a void pointer to whatever data you want which is passed back to your function. In your callback function, key is a string containing whatever key was specified using the Set() command in DAQConnect. Note that the connector information is stripped from the key, so "remote.mykey" becomes just "mykey". The connector part is used to determine who gets the response. "value" is also a string, but could contain a numeric value, i.e. "3.412", and is simply the value parameter of the Set() command. Like the Status Callback, your function will be called by a secondary thread so you must ensure that it is fast and doesn't call any Windows API calls.

DAQFactory

DAQFactory is a data acquisition application that can connect to a wide variety of hardware devices, including PLCs and RTUs, collect the data and send it to DAQConnect. DAQFactory Express is available for free from www.azeotech.com. Simply download the DAQFactory trial and when the trial expires, it will revert to a free version that you can continue to use. The full paid versions of DAQFactory offer many more features such as automated control, advanced screens, alarming, and much more.

DAQFactory release 5.80 and newer supports direct, easy connectivity to the DAQConnect service to allow you to view your data live from a web browser without having to worry about your firewall or router settings, static IPs or any thing else related to IT really. You don't need an external website either, as would be required using FTP. In 5.86 the connection method between DAQFactory and DAQConnect changed to make things easier. Older documents will still work, however, the user interface has changed. This documentation applies to the new user interface:

Sending data to DAQConnect:

You can easily start connecting to DAQConnect by going to **Real-time Web -> Connect...** in the DAQFactory main menu. The first time you do this in a session, you will be prompted to login or create an account. If you've already created a DAQConnect account, click login, otherwise, you can create a free DAQConnect account right from this window. If you haven't created any data sources yet, after logging in or creating an account you will go directly to the page for creating new data sources. A data source is a group of channels within DAQConnect. Typically you'll create a data source for each location sending data to DAQConnect, but you could also use separate data sources for separate applications running on a single computer. Give your data source a name, which like all DAQFactory names must start with a letter and contain only letters, numbers or the underscore. Once the data source is created, your DAQFactory document will be linked to it.

Note: if you are using a proxy to get to the Internet or a dedicated instance of DAQConnect (i.e. you access DAQConnect through a URL other than www.daqconnect.com, please see the Settings section below before you you do Connect...

To then send data to DAQConnect, you simply mark the desired channels for sending:

1) Go to the channel table by clicking **CHANNELS** in the workspace. Towards the right of the table you'll see three columns related to DAQConnect:

DAQConn?, determines if the channel data is being sent to DAQConnect. Check this box to mark the channel.

DC Hst:, determines how many data points are kept on the DAQConnect server for historical viewing. This number is limited by the plan you select. You can enter -1 to have DAQConnect automatically apply an even amount to all of your tags based on your plan, or you can specify exact amounts for each channel allowing you to have more data retained for certain channels.

DC Intvl: determines how often the channel data is sent up to DAQConnect. This is in data points. So, if you are taking data once a second and you put a value of 10 in this column, every 10th data point is sent. This is to allow you to take data at a faster rate than it is sent to DAQConnect. Please note that DAQConnect may limit the spacing of data points being sent to it. If you need to send high speed data such as streamed data, you should contact DAQConnect to arrange for a special plan.

Once you click **Apply** in the channel table, DAQFactory will start sending the data to DAQConnect. You can go to DAQConnect, and in the Basics menu see your tags appear. You can then start creating your screens within DAQConnect to view your data from any browser.

To change the data source you are sending data to, you can once again go to **Real-time Web -> Connect....** You will be taken to a screen listing all your data sources on your account, highlighting the one you are currently

connected to. You can then select a different data source.

Settings:

From Real-time Web -> Settings... you can adjust a couple settings for your DAQConnect connection:

Display Detailed Status: checking this box will display detailed status information about your DAQConnect connection in the Command / Alert window. This is quite useful when you are first establishing a connection. This setting is not saved with the document as its designed primarily for debugging.

Host: this setting should only be changed from www.daqconnect.com if you are using a proxy server or a dedicated DAQConnect instance. If using a proxy, set this value to the IP of the proxy. If using a dedicated instance (and no proxy), set this to the URL of your dedicated instance.

Host Name: this setting should only be changed from www.daqconnect.com if you are using a dedicated DAQConnect instance. If so, you should set this to the URL of the dedicated instance.

Control from DAQConnect:

DAQConnect can also send commands back to your DAQFactory application. To ensure that the commands are processed the way YOU want them to be, there is no automatic command processing. Instead, when a command comes in from DAQConnect, the system event "OnDCSet" is triggered, passing in the two values from DAQConnect: the key and the value.

To allow a remote user of DAQConnect to change values in your DAQFactory installation:

1) in DAQConnect, you will use the `system.sendCommand()` function (or other possible output functions) in an Click or other event. These functions will take a key and a value. The key is a string and often contains the name of the connector with a dot after it, such as "MyConnector.MyKey". The value can either be a string or number, and may be part of the function. Please review the DAQConnect help for more information.

2) in DAQFactory, create a Sequence called "OnDCSet". This sequence will be run whenever a new command comes in from DAQConnect. Commands from DAQConnect only come in when DAQFactory sends data to DAQConnect. So, if your channels are updating every 60 seconds, you can only possibly get commands from DAQConnect every 60 seconds. In these situations, you might consider using a Test channel or similar to ping the server more rapidly.

3) When the sequence is called, two local variables are created: "key" and "val". What these mean is largely up to your design. If, for example, you were directly controlling a digital output, "key" might be the name of a channel, and val either 0 or 1. In this case, you might use this script:

```
if (key == "mydigout")
    mydigout = val
endif
```

If you wanted to do more channels, you'd just repeat, or use a switch/case instead.

Now some important details:

- key is always lowercase, no matter how you specify it in DAQConnect, and will NOT include the data source name. This is stripped by DAQConnect.
- val is a string, but will automatically get converted to a number when setting numeric output channels. For other uses, you may need to use `strtodouble()` to convert it to a number

- in the above example we did no constraining or safety checks. Unless the output being controlled doesn't connect to anything dangerous, you should always do a safety check or ensure proper safeties are in place in hardware. Remember that the output could be controlled remotely, and you should assume it could change at any time, even if you think you are the only one with access.
- here's an example of a constrained output:

```
private nVal = strtodouble(val) // convert to a number
if (key == "myout")
    if ((nVal > 20) && (nVal < 40))
        myout = nVal
    endif
endif
```

Note that we converted the string "val" to a numeric "nVal" before doing comparisons. > and < and other operators will work on strings, but work in a different way ("2" is actually > "10" for example, even though 2 < 10)

- the OnDCSet event should run fast. Do not put loops or delay()'s in this sequence. If you need to do a loop or delay, do it in another sequence started with beginseq()

NOTE: if you lose your Internet connection or otherwise have troubles sending data to DAQConnect, the data will queue up until a connection can be reestablished. There is a hard limit of 1,000 data points per packet across all tags to ensure you don't run out of RAM. If more than 1,000 data points queue up while waiting for a connection, the oldest data is lost.
