\

PC Based Control and Traceability

Configuration Manual for
version 3.6.4.7 of PlantWatch.

www.PlantWatch.com

Developed by HTEinc

"Your Technical Solutions Provider"
www.hte.net•(248) 371-1918•Fax (248) 371-2185

Table of Contents

Installation 5
Getting Started 5
- Available Tools for Learning 5
- System Usage 6

Overview 6
- Component Overview 6
- Devices 8
- Logic Charts 10
- Logic Chart Execution 11
- Logic Chart Triggering 11
- Variables 11
- Graphic User Interface 12
- OPC 13
- TCP/IP 13
- PLC communications 14
- IO 14
- Part Numbers 14
- Database Browser 14

Tutorial 15
Devices 40
- Adding a New Device 41
- Interval 41
- Socket 42
- COMPort 43
- Message Determination Method (MDM) in SOCKETs and COMPorts 44
- Overview 44
- Variable 47
- ReadFromFile 47

Logic Charts 49
- Logic Chart Value Types 49
- Logic Chart Cell Types 50
- If Cell 51
- If Then Cell 54
- Birth Cell 57
- ConsumeMaterial Cell 58
- CollectDataPoint Cell 61
- WriteToDevice Cell 63
- WriteToOPC Cell 64
- WriteToFile Cell 65
- Trigger Exe Cell 66
- Split Cell 67
- Math Cell 68
- File Manager 70

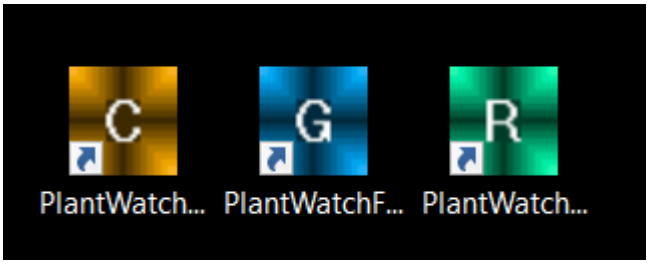
DBBrowser Cell 72
Substring Cell 88
Local Type Variables 90
OPC Type Variables 91
Array Type Variables 94
Graphical User Interface 98
Overview 98
Adding a Screen to the application 100
Adding a Animation to the current Screen 101
Change Screen 103
Output Text 105
Input Text 107
Set Variable 108
Images 110
Do Logic 114
Variable List 116
Drop Down List 118
Security Configuration 119
OPC 122
What is OPC? 122
OPC within PlantWatch Overview 122
Servers and Groups 123
Configuring OPC Servers 123
Getting OPC Data into PW. 126
Getting OPC Data out of PW. 128
IO 132
Hardware 132
Incoming Data 133
Outgoing Data 133
Configuration 133
Part Number Setup 136
Parts and Sub Parts 137
Part Attributes 140
Appendix A 141

Installation

Run the installation file provided with the Plantwatch installer Zip file.

The install should complete in less than 10 seconds

Your desktop should now have four new icons as shown below



That completes the installation.

Please read the *Getting Started* section of the manual before proceeding further.

Getting Started

Available Tools for Learning

There are several tools available that can be used to learn about Plantwatch.
(All of these tools are located in c:\HTE\Documentation)

This manual in general but specifically the following sections:

- Installation
- Getting started
- Overview
- Tutorial

The configuration example HowToExample_BarcodeToFile.doc. This is a example of how to configure Plantwatch to connect to a bar code scanner and place the data coming from it into a file.

The three videos Barcode1Of3_Connect.mp4, Barcode2Of3_Variables.mp4 and Barcode3Of3_Display.mp4. These videos show the process of configuring Plantwatch to connect to a bar code scanner and present the data to the screen.

It is highly recommended that you first read the *Overview* section of this manual.

System Usage

Read this section to gain an understanding of how to use Plantwatch for the first time.

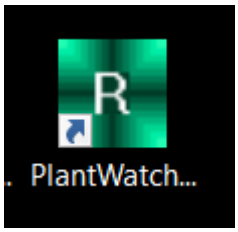
Plantwatch is a software product that is configured and then started.

After installation you must open up the *Plantwatch Configurator,* create an application and save it before starting Plantwatch. This will be an application that does nothing, but you must still create it.

The default password is 123.

Follow the instructions in the *Tutorial* section of this manual to learn how to create a Plantwatch application.

After creating an application, you can start Plantwatch.
Click on the *Plantwatch Runtime* icon to start Plantwatch.



After a moment you will see the Plantwatch screen.

Overview

Component Overview

PlantWatch is configured with the PlantWatch Configuration Tool and Graphics Editor.

Then PlantWatch Runtime is started.

It has 8 important areas that need to be configured.

Devices – *Devices* perform two functions, getting data into PlantWatch and triggering the execution of *Logic Charts*. Data can flow to and from PlantWatch thru a device, such as a bar code reader thru RS232.

When new data flows into the system thru a *Device*, the *Device* is triggered and all logic charts associated with that *Device* are executed.

It is also possible to write data out thru a *Device*.

Logic Charts – A *Logic Chart* is a collection of cells made up of action cells and logic cells. Each Logic Chart is associated with a specific device. When that specific *Device* is triggered all of the logic charts associated with the Device are executed. When a logic chart is executed it will process the configured cells starting at the top left and will execute cells until it reaches a point where there are no more cells to execute.

There are two types of cells, action and logic, within a *Logic Chart.*
An example of an action cell is *WriteToOPC*. Using the *WriteToOPC* cell you could write a value to a register in a PLC.

An example of a logic cell is *IF*. Using an *IF* cell you can control the execution path within the *Logic Chart.* If the statement defined within the *IF* cell is true, the cells following the *IF* will be executed. If the statement defined within the *IF* cell is false, the cells following the *IF* will not be executed.

Variables – *Variables* are used to bring OPC and IO values into PlantWatch as well as holding values used in logic charts or screens.

Graphical User Interface – Used for graphical presentation and user interaction. It allows you to have many screens with each screen having buttons, images, input text and display text.

OPC – Provides a data connection to most PLC's and all OPC compliant components.

IO – Allows you to connect directly to plant floor devices such as proximity switches and motor starters. PlantWatch's *IO* system is made up of *Racks* which have *Slots* that can be populated with a variety of different *Cards, d*igital or analog. *Racks* can have up to 8 *Slots*. There are *Cards* for 24VDC inputs, 24VDC outputs, 110 VAC inputs, 110 VAC outputs, Relay Outputs, 0 to 10 VDC analog Outputs and many more. All Inputs cards data is mapped into PlantWatch *Variables.* To write a value to an *Output Card* the WriteToIO cell type is used within a Logic Chart.

Traceability – The definition of part numbers, their parent child relationship and attributes.

Within PlantWatch there are actions specific to traceability such as the *Birth* cell. When invoked the *Birth* cell will track the creation of a unique item by inserting data into the associated SQL database. To validate that the data is correct, the definition of part numbers and part attributes are setup in advance of collecting the data.

Database Browser – Functions that allow bi directional communication to SQL databases.

Barcode Printers – Functions that allow printing barcode labels.

Devices

A device is a connection to a source of information. An example could be a TCPIP Socket network connection. When new information arrives thru the network the device causes all logic charts associated to the device to be executed. As the logic Charts are executed they are passed the value coming in to the device.

Examples of supported source of information types:

Barcode scanner thru COMPort device. (RS232)
PLC using the OPC device.
Text files using the ReadFromFile device
Vision Systems using the TCPIP device. (Ethernet)
Voltages thru the IO system
SQL: databases thru the Database Browser

Each device type has unique attributes but all of them result in a single value. A device will monitor it's data source and if it changes, it will trigger all Logic Charts associated with that device, and pass the newly acquired value into the logic chart(s) associated with the device. Most device types can also be written to.

Device types:

COMPort – RS232
Socket – TCP over Ethernet
ReadFromFile – ASCII text files
Variable – Watches one variable for any change
Interval – Periods of time such as 100 milliseconds.

As you create a Device you must choose what type of device it is. After creating the device you must configure it. Note that the device's configuration will be based on the

type of device it is. For example, a Comport device will have baud rate, parity, stop bits etc. An Ethernet device will have an IP address and Port number.

COMPort

Comport devices allow communication to RS232 data. Data can be received or sent.

Configuration for the COMPort includes:

Port ID – The com port number. For com3 you would enter 3.
Port Configuration – Baud rate parity etc. You could enter "9600,8,n,1" for the settings.

An additional type of configuration is called the Packet Determination Method, or PDM. This applies to several devices and is covered later in the manual.

Socket

Socket devices allow communication to TCPIP data, over ethernet. Data can be received or sent.

Configuration for the Socket includes:

Remote IP Address – The TCPIP address of the server you wish to interact with. You could enter 196.1.1.33.
Port ID – The port number of the server you wish to interact with. You could enter 4096.

An additional type of configuration is called the Packet Determination Method, or PDM. This applies to several devices and is covered later in the manual.

ReadFromFile

ReadFromFile devices allow communication via file transfer. PlantWatch can watch for the existence of a text file, read it and get the information it contains. The data from the file will be passed into all logic charts associated with the device. Data can only be received. (To create text files, or write into existing text files, you could use a Write to File action within a logic chart.)

Configuration for the ReadFromFile includes:

File Spec – The path and file name.
Search method – One of three types
File Exists, no matter the content of the file
Data Within the file matches a preset value
There is some data in the file, no matter the contents
Scan Rate – How often you look for the file.

Delete File After Read – Should the file be deleted after it has been read.

An additional type of configuration called the Search Method allows you to define a valid piece of data.

Variable

Variable devices execute their logic charts each time a specified variable's value changes. Variable devices allow you to link logic charts together. One logic chart can set a variable to a value. If the variable is specified within a device, all of it's associated logic charts will run.

Configuration for the Variable Device includes

Selected Variable – The variable who's value causes this device to run.

Interval

Interval devices do not communicate to anything. They provide a method of invoking logic charts at timed intervals.

Configuration for the Interval includes

Seconds – Select the number of seconds from:

.1
1
5
15
30
60

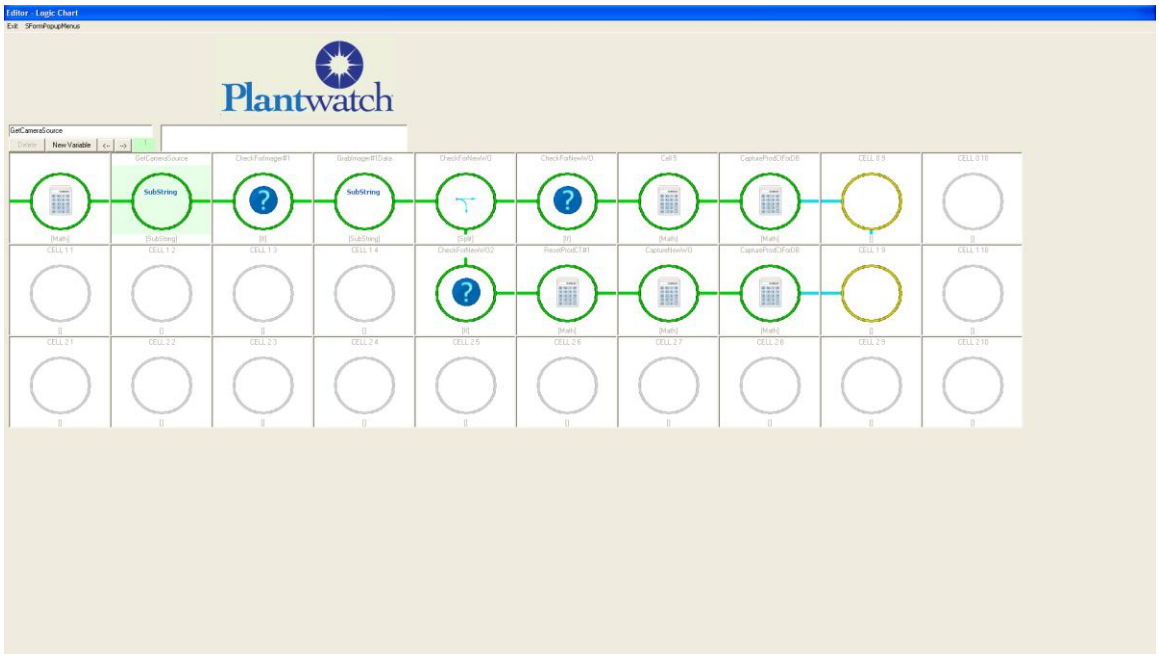
Logic Charts

Logic Charts allow you to combine work and logic in a simple flow chart. The chart has cells that flow from top left to bottom right. Cells are assigned a type from a collection that includes both logic and work cell types.

A *If* cell is an example of a logic cell type. When an *If* cell is encountered as the Logic Chart is executed, it's work will be evaluated and only if the logic is true will cells to the right of the *If* cell execute.

A *WriteToFil*e cell is an example of a work cell type. When *a WriteToFile* cell is encountered as the logic chart is executed, its work will be done and then the rest of the *Logic Chart* will be executed.

A *Split* is an example of a logic cell type. It allows flow to the line under the split.



Logic Chart Execution

When a *Logic Chart* executes it always starts with the top left cell. It will execute the top left cell, and then continue to the right until it encounters a cell that is not configured or a *If* cell that fails.

After executing the top line it will execute the middle and bottom lines in a similar fashion.

Logic Chart Triggering

Logic Charts are triggered by the device it is associated with. Each time a device's value changes, all Logic Charts that are associated with that device will be executed. Consider a Socket device. (This device allows two way communication with any other Ethernet based program or hardware). If any data comes into the computer thru the specified Ethernet Socket *Device*, all logic charts associated with the Socket device will be executed.

In addition to using *Variable* and *Constants*, *Logic Charts* can also use the value of it's associated *Device*.

Variables

A variable is a named storage location that can be used for real time data storage and used within any Logic Chart Cell.

There are three types of Variables, Local, OPC and Array.

Local type variables are used for storing values within Plantwatch for later reference by a logic chart cell or graphic screen.

OPC type variables are used to bring in data from PLCs via OPC servers. OPC type variables are associated with an OPC item and are constantly updated with the items value.

A single Array variable can hold many values which can be presented to a user via a list box on a graphics screen or used within a Logic Chart cell.

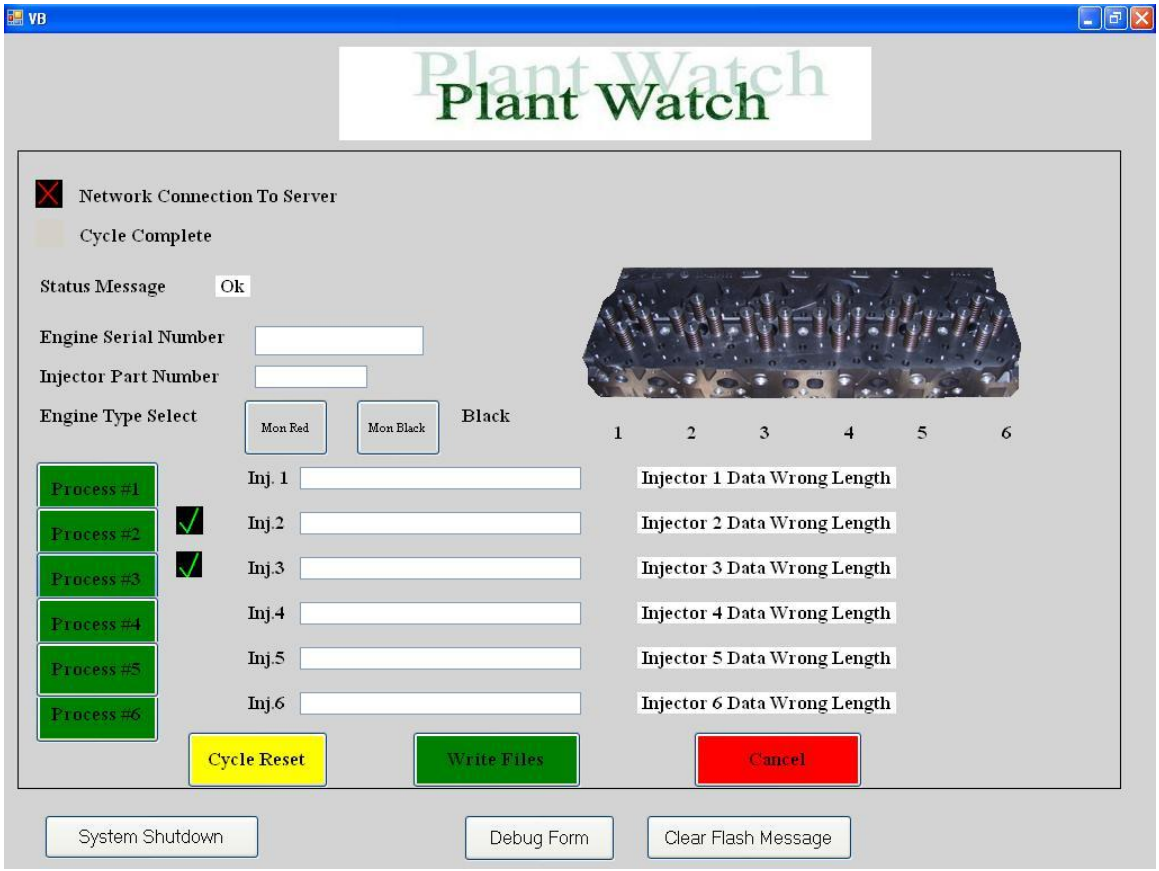
Graphic User Interface

PlantWatch has Graphics to provide you with editable screens that can be used to interact with the user. Graphics allows you to have many unique screens. Screens can have:

Input Texts – For data entry
Output Texts – For data display
Buttons – To change screens, set values and trigger logic charts
Pictures – To display graphically what is happening
List Boxes – To allow a user to select one from a collection of data

Security stops users from causing any action unless they know the password.

This is an example of a screen created with PlantWatch



Example of a screen created with PlantWatch's Graphics

OPC

OPC is a communication tool allowing PlantWatch to communicate to most PLCs. It routes all communication to a supported OPC server for the specific PLC vender. For example to communicate to Allan Bradley PLC's you could use Rockwell software RS Linx.

OPC communication for inbound data comes thru a variable defined as type OPC. OPC communication for outbound data goes thru an OPC Write cell in a logic chart. Note that OPC is covered in detail in it's own section within this document.

TCP/IP

Native mode TCP/IP to connect to Ethernet devices including Mod bus

PLC communications

PLC's supported

AB Control Logix
Siemens S7
GE

IO

PlantWatch's IO allows you to connect directly to plant floor devices such as proximity switches and motor starters. You can read inputs or write to outputs.

PlantWatch's *IO* system is made up of *Racks* which have *Slots* that can be populated with a variety of different *Cards*, *d*igital or analog. *Racks* can have up to 8 *Slots*. There are *Cards* for 24VDC inputs, 24VDC outputs, 110 VAC inputs, 110 VAC outputs, Relay Outputs, 0 to 10 VDC analog Outputs and many more. All Input card data is mapped into PlantWatch *Variables.* To write a value to an *Output Card* the WriteToIO cell type is used within a Logic Chart.

Each rack is assigned a node address and is connected via Ethernet to the computer

Part Numbers

Within PlantWatch there are actions specific to traceability such as the *Birth* cell. When invoked the *Birth* cell will track the creation of a unique item by inserting data into the associated SQL database. To validate that the data is correct, the definition of part numbers and part attributes are setup in advance of collecting the data.

To *Birth* a part it must first be declared within *Part Number Setup* as a valid part number. Once a part number is created it can be configured to have attributes and to be consumed by another part.

An example of an attribute could be the diameter of a shaft. If we had a part number that represented an axle, an attribute may be desired where the actual measured diameter of the axle shaft could be stored.

Parts often are consumed by other parts. For example, as a car is built it will consume 4 wheels, four tires, one motor and one transmission. These allowed consumptions must be declared in advance of collection of the data.

Database Browser

Database Browser is a function that allows bi direction communication to SQL Databases. Database Browser can insert, select, modify or delete data in a SQL database.

Database Browser can connect to a database thru two different communication styles.

ODBC
Native SQL

Database Browser supports Plantwatch Array type of variables. This allows Database Browser to manage an array of records. Database Browser can select an unknown number of records or insert many records with just one insert.

Configuration of Database Browser is broken down into two parts. One part to define the connection to the database and then logic cells to define the exact transactions that need to occur.

There is a branch on the configuration tree call Databases where the connection to the database is defined.

To make a new Database Connection right click on Database Browser Databases and select New Database Browser Database Connection. Give it a name, enter a valid database connection string and save it. Now you can enter Database Browser cells in any logic charts to define the exact transactions that you need to occur. See the Database Browser cell documentation under Logic Chart for further information.

Tutorial

Overview

This tutorial will create a simple timer that counts from 1 to 7 in one second intervals. When it gets to 8 it resets to 1. It will show the current value on a user created screen.

The purpose of this tutorial is to familiarize the user with Plantwatch's configuration environment.

For a tutorial on how to connect a barcode scanner to Plantwatch, see file PlantWatchExamples.doc.

For this example we will need a one second interval. This will be obtained by defining a device of type *Interval*, and then by setting up the interval on the device to one second. A device set up as an interval device for one second will cause all of it's associated logic charts to be processed every one second.

Now we have a one second event.

After creating our device of type one second interval, we will create a Logic Chart associated to our new device. Each time our device fires, all logic charts associated with

it will be executed. We will configure this new logic chart to increment a variable up to 7 and then reset it.

Now we have an empty logic chart that is executed every one second.

For this example we will need a place to store the current value. We will define a *Local* type variable. We will call our variable SimpleTimerValue.

Now we will have an empty logic chart that is executed every one second, and a spot to store a value.

We still need to increment a number up to 7 and then reset it. We have a variable to store the value in, SimpleTimerValue, and a logic chart to do the work. Now we must configure the logic chart.

The logic we are going to create is:

1 Increment SimpleTimerValue
2 If SimpleTimerValue is greater than 7 set SimpleTimerValue equal to 1

We will use a Math cell followed by an If cell followed by a Math cell to create the above logic.

The first Math cell will add one to SimpleTimerValue.

The If cell that follows will test the value of SimpleTimerValue against the constant 7. If SimpleTimerValue is greater than 7, execution will continue to the third cell.

The third cell is a math cell that will always set the value of SimpleTimerValue to 1, causing a reset.

All of the rest of the cells are inactive and do nothing.

Now we have a logic chart that increments and resets SimpleTimerValue from 1 to 7, but we cannot see it so we will create a screen that displays the value of SimpleTimerValue.

Now we have a logic chart that increments and resets SimpleTimerValue from 1 to 7, and we can see it.

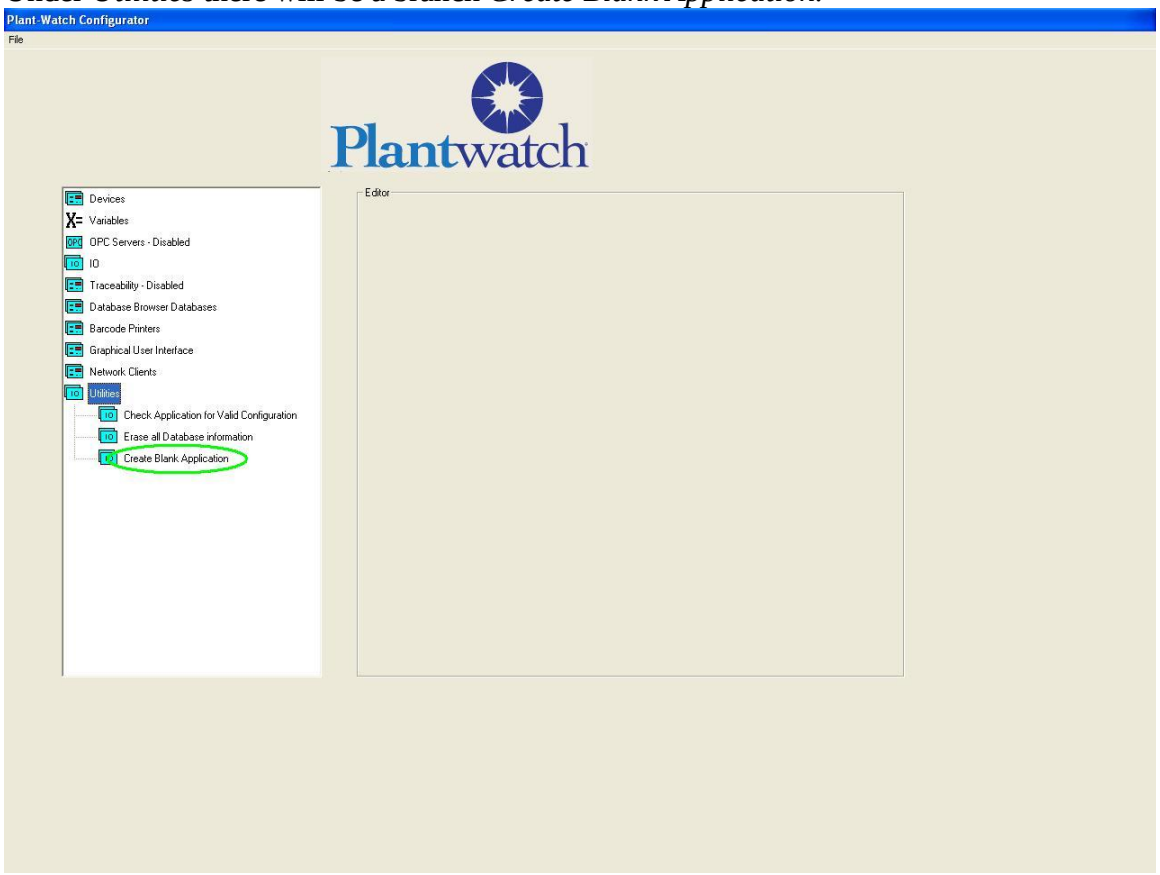
That is the functionality we will create.

Procedure

1. Clear all prior configuration

   Start the PlantWatch Configurator and then double click on the *Utilities* branch of the tree to expand it.

Under *Utilities* there will be a branch *Create Blank Application.*

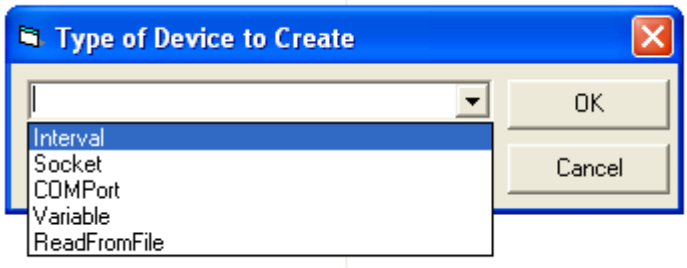


Click on *Create Blank Application.* You will be prompted for a password. The default password is 123.

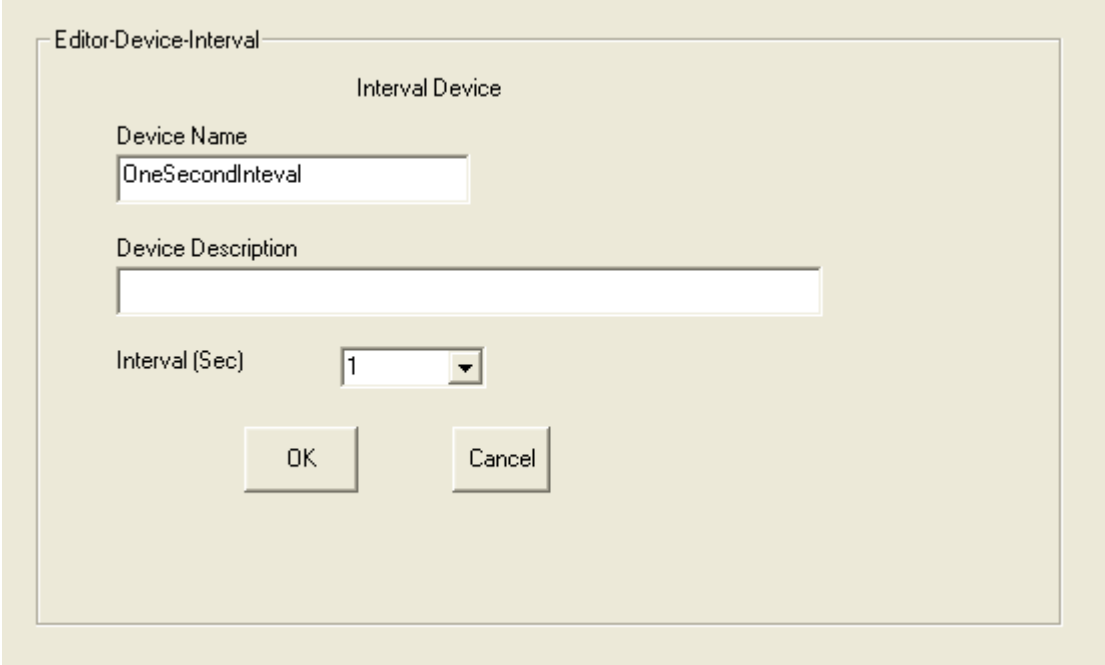
2. Create the one second interval *Device*

Right click on the word *Device* on the left panel, and then left click on *New Device*.

You will be presented with a dialog that will allow you to select the type of device you want to create.



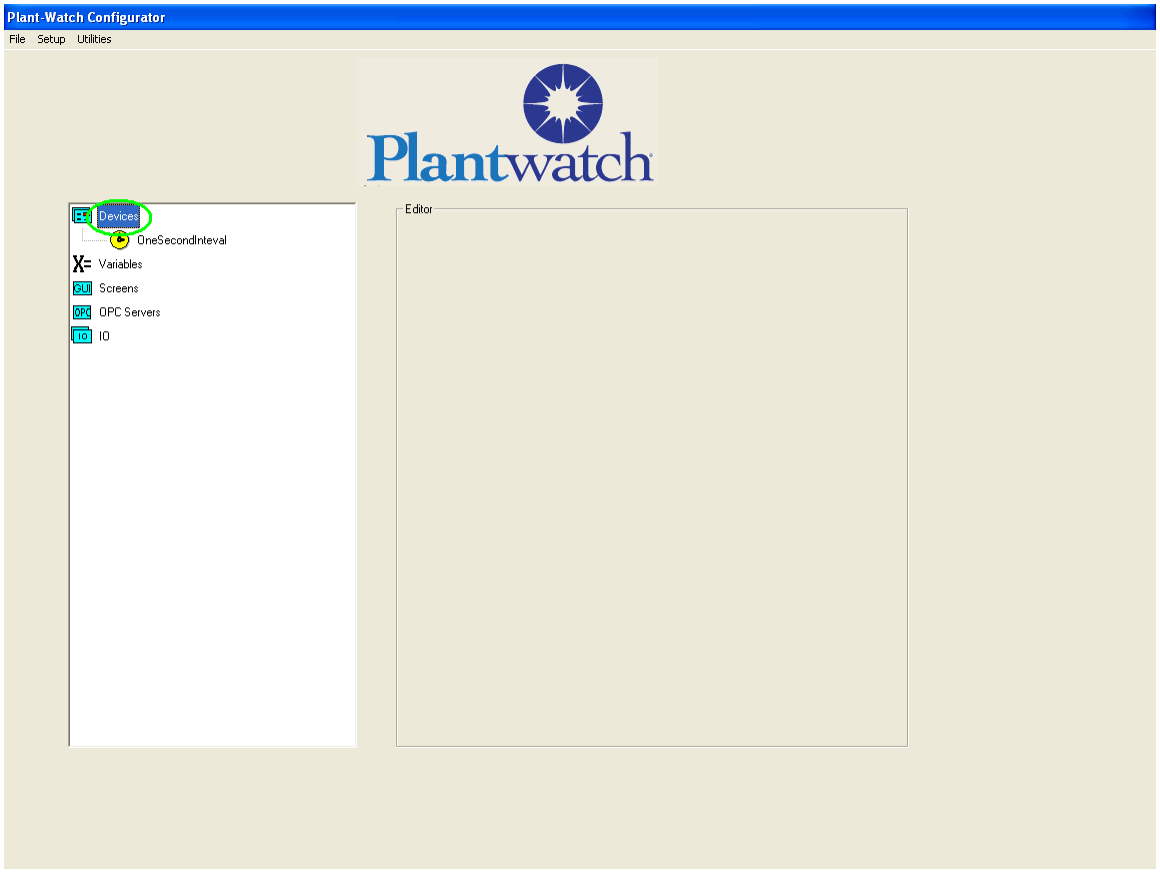
Select Interval from the list and then select *OK*. You will be presented with the configuration panel for an Interval type device.



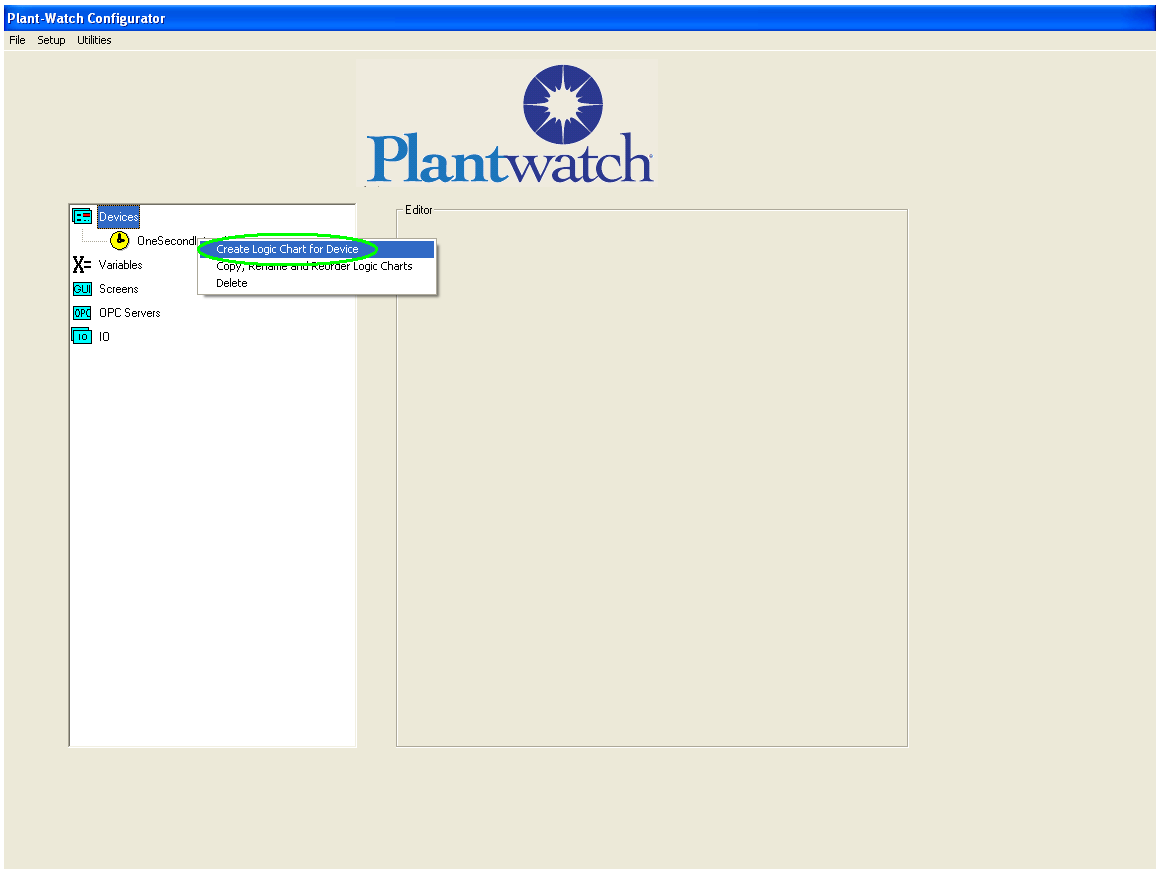
Enter OneSecondInterval for the device name. It defaults to one second so nothing needs to be done for it's interval setting. Then select *OK* to complete creating the device

3. Add Logic chart to new device.

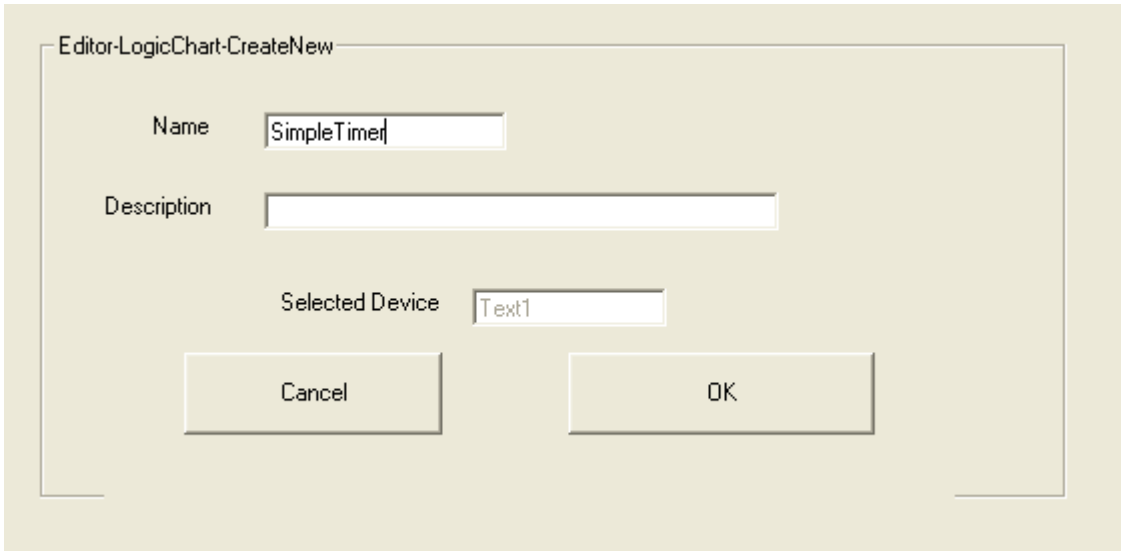
   Devices have two functions. Communicate to data sources such as RS232, and to trigger logic when the value of the data source changes. To trigger logic when the value of the data source changes you must create a Logic Chart associated to the Device. An interval device will trigger its associated logic charts each time the interval of time passes.



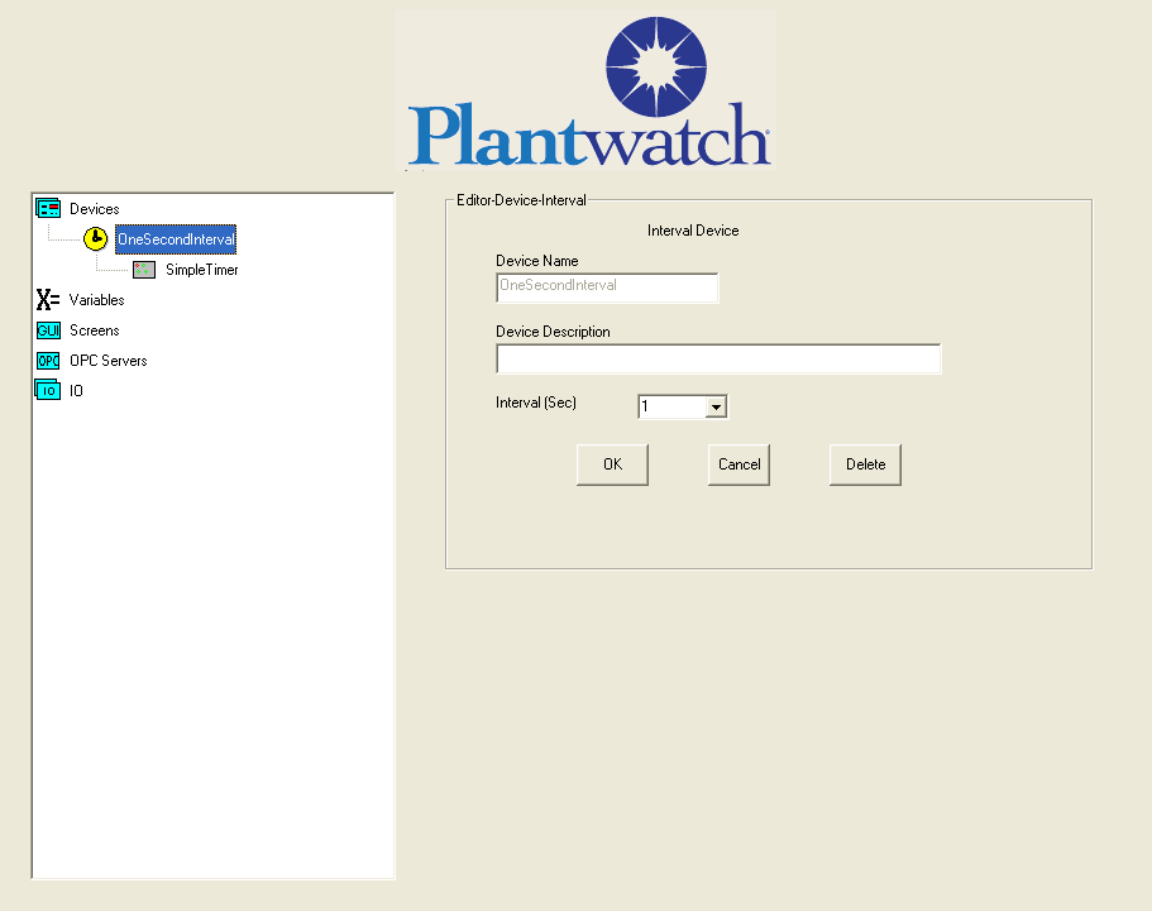
Double click on devices to expand out the tree to display the existing devices. After doing so you will see our new device.



Right click on our new device *OneSecondInterval* and select *Create Logic Chart For Device* . You will be presented with the dialog to name and create a new logic chart.

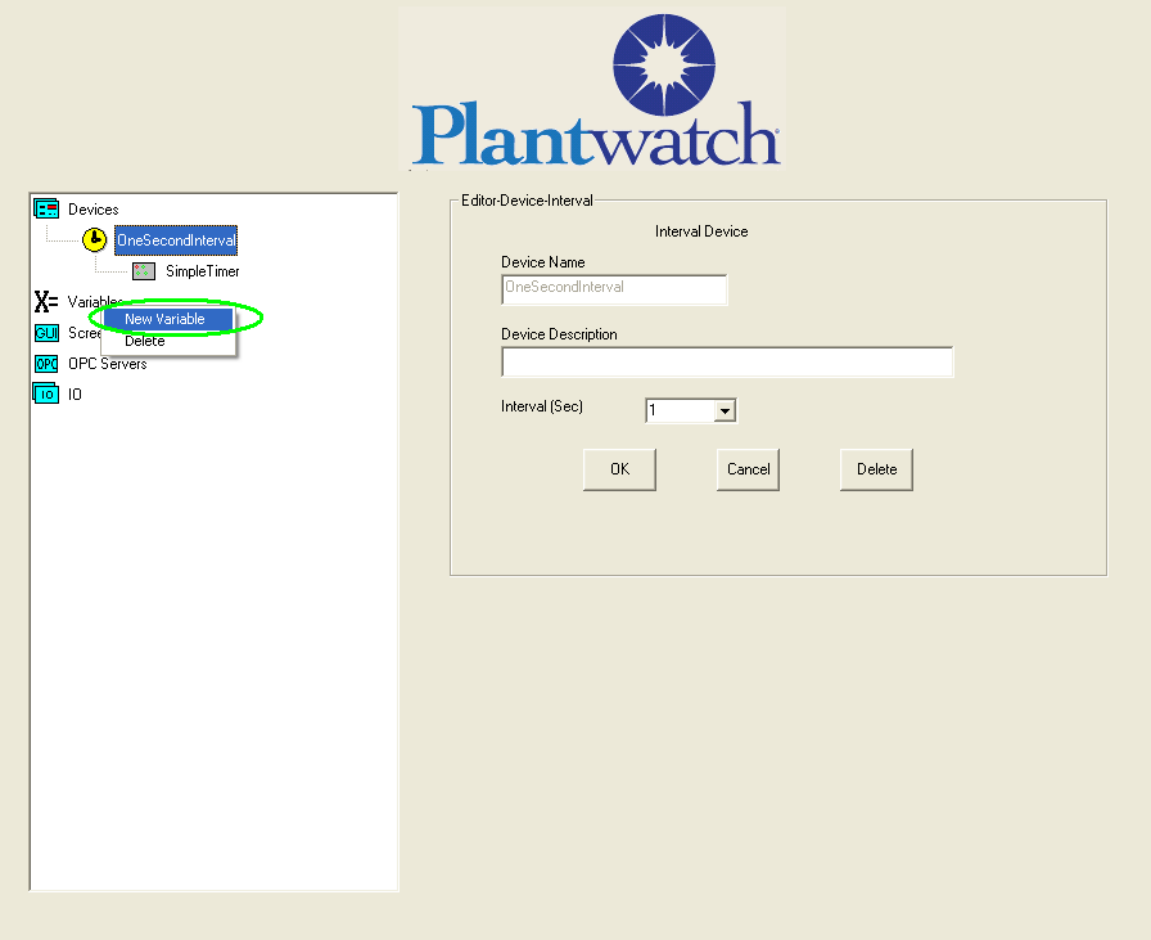


Name it SimpleTimer. And then select *OK* .

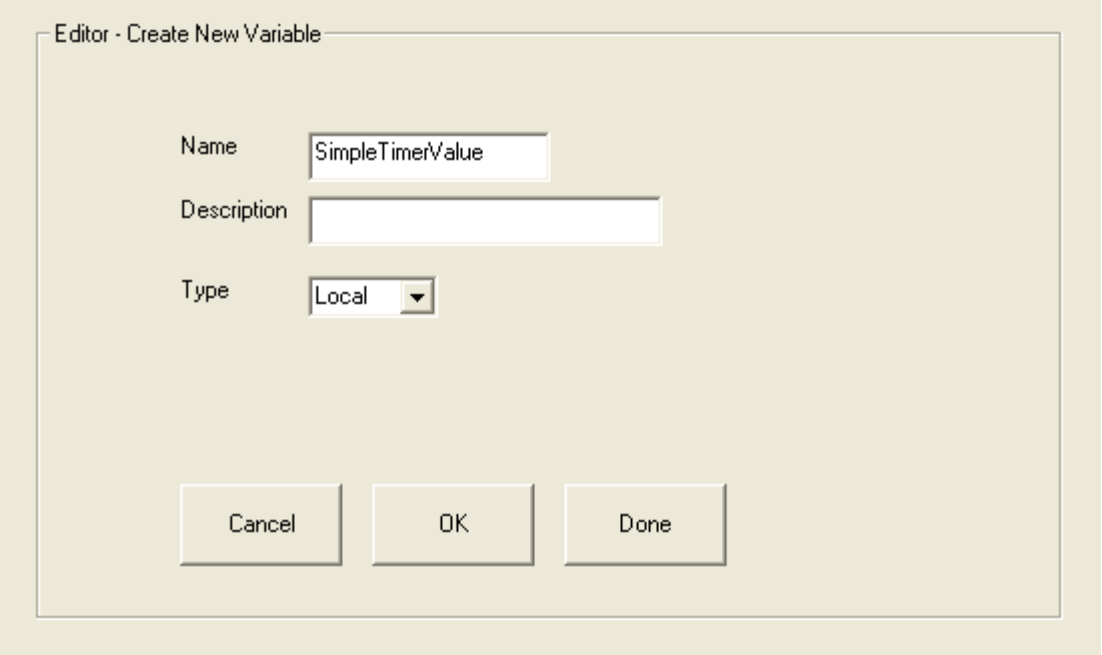


When you expand out devices you will see our device *OneSecondInterval*. You can now expand our device to see our new logic chart called *SimpleTimer*

4. Create a Variable to store our count in named SimpleTimerValue



Right click on *Variables* and then select *New Variable.* You will be presented with the create new variable dialog.



Enter in SimpleTimerValue for name and then select *Done*.

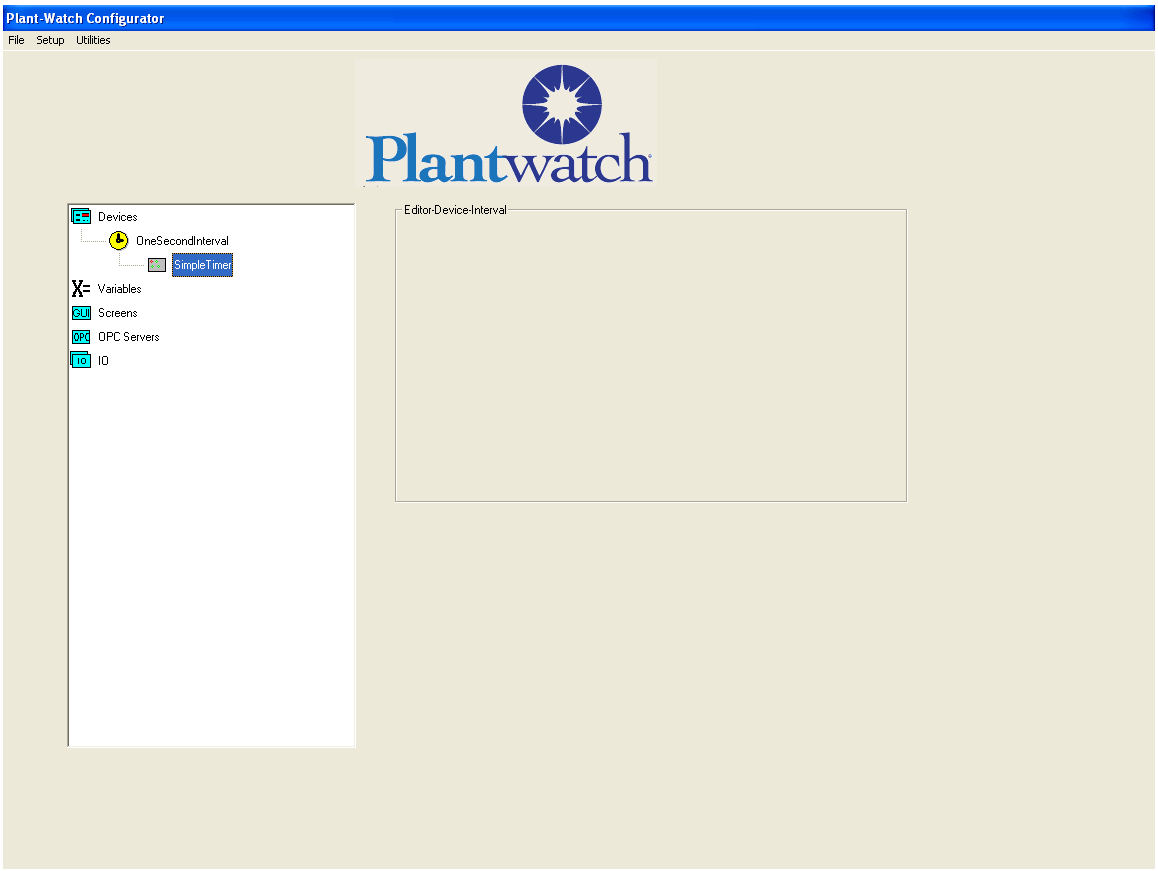
There are twp types of variables, Local and OPC. OPC variables allow communication to external devices. Local variables are simple storage location for values. For our timer function, we just want simple data storage so choose *Local* for type.

5. Configure our logic chart to increment and reset the variable SimpleTimerValue.

   The logic we are going to create is:

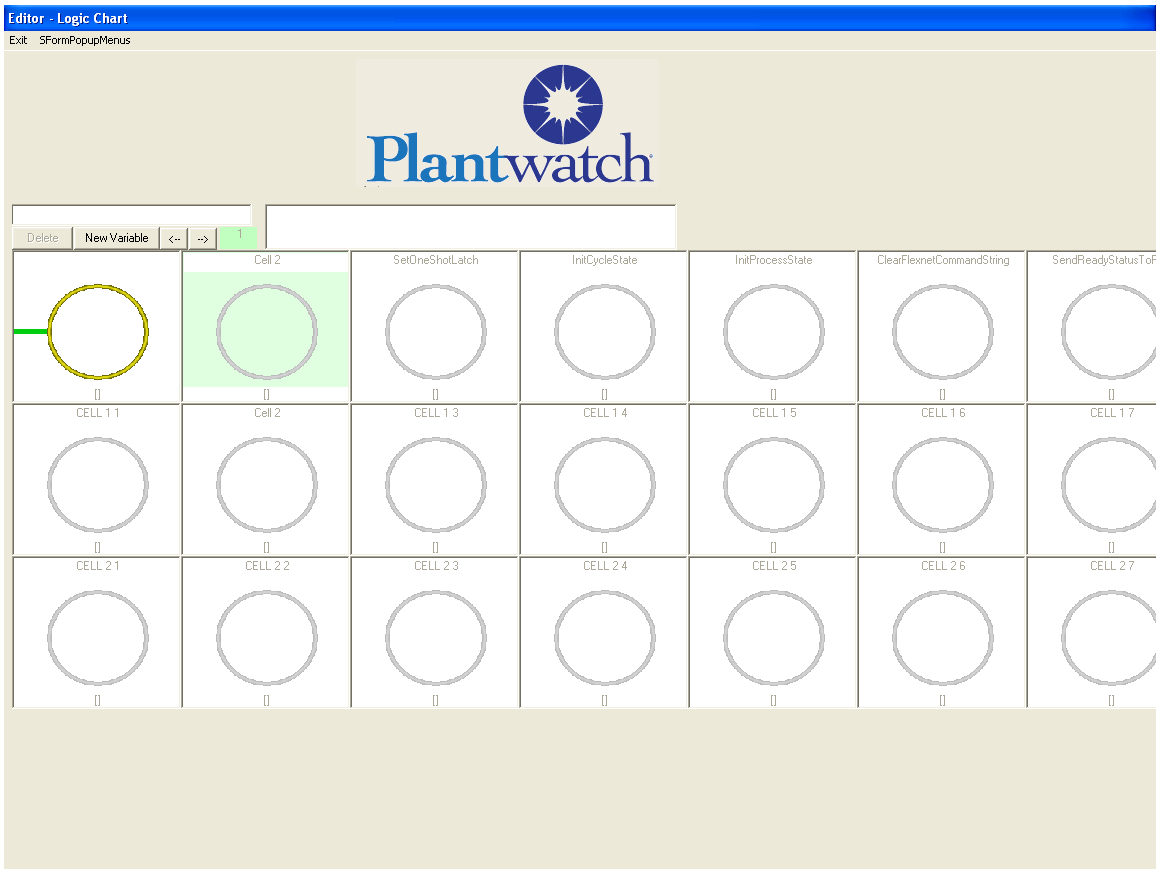
   Increment SimpleTimerValue
   If SimpleTimerValue is greater than 7 set SimpleTimerValue equal to 1

   We will use a Math cell followed by an If cell followed by a Math cell to create the above logic. The first Math cell will add one to SimpleTimerValue. The If cell that follows will test the value of SimpleTimerValue against the constant 7. If SimpleTimerValue is greater than 7, logic chart execution will continue to the third cell. The third cell is a math cell that will always set the value of SimpleTimerValue to 1, causing a reset.



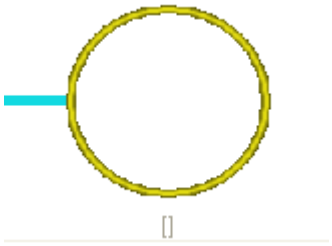
To Define our Logic Chart SimpleTimer, expand out devices by double clicking on Devices.

Expand out the device OneSecondInterval by double clicking on it and it will expose our logic chart SimpleTimer.



Double click on Logic Chart *SimpleTimer* and the logic editor will open. Also note that there is only one cell, the one located on the upper left, that has a path indicator attached to it's left side.

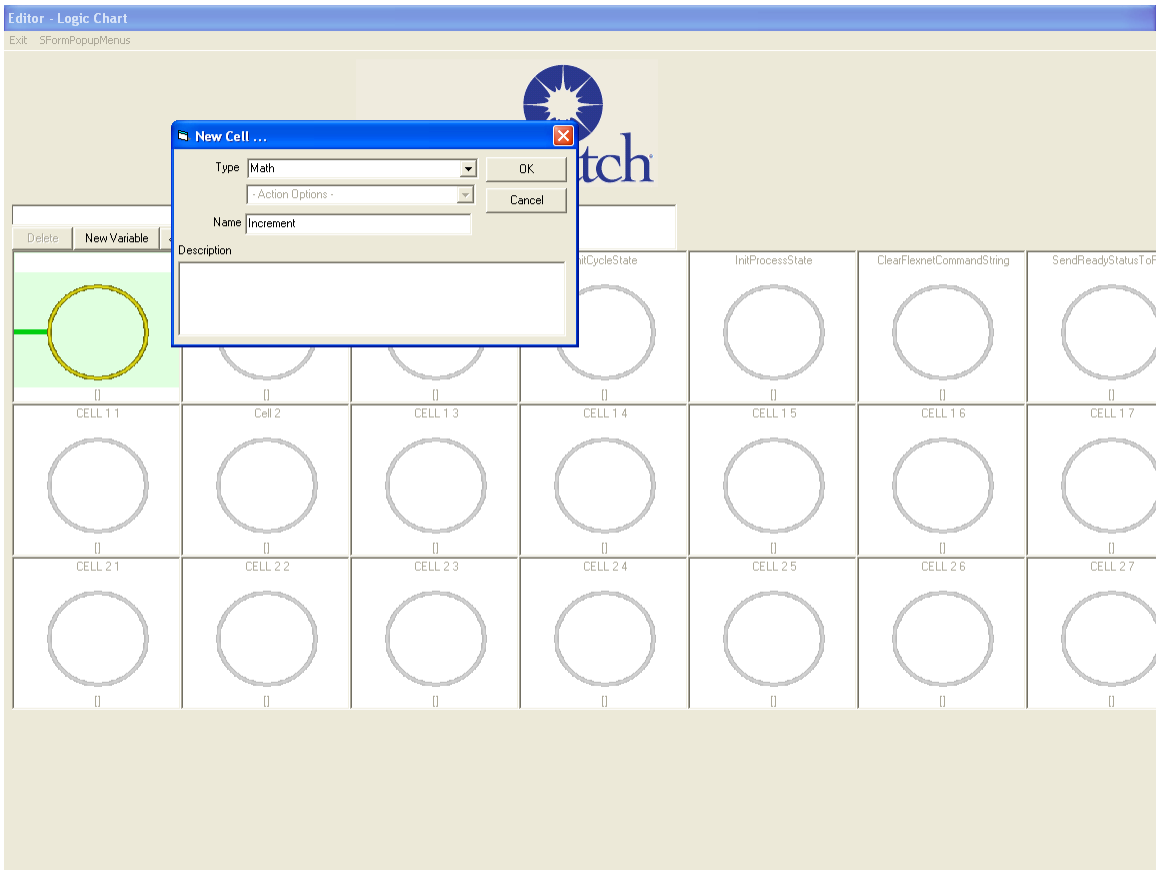
Only the first unused cell on each line is available for use. If that cell is activated, the next cell becomes available for use. When a cell is available for use it will have a path indicator attached to it's left side and it's color will not be gray.



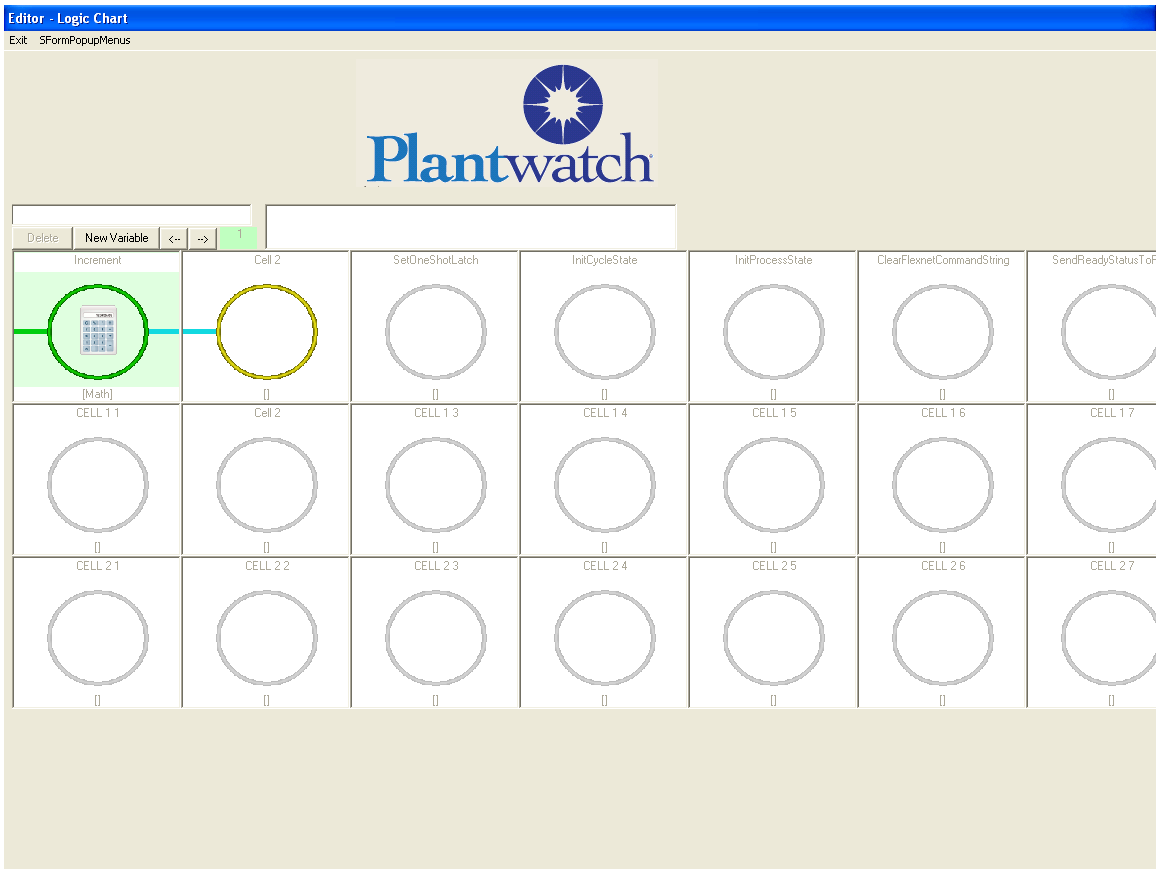
available for use

Not available for use

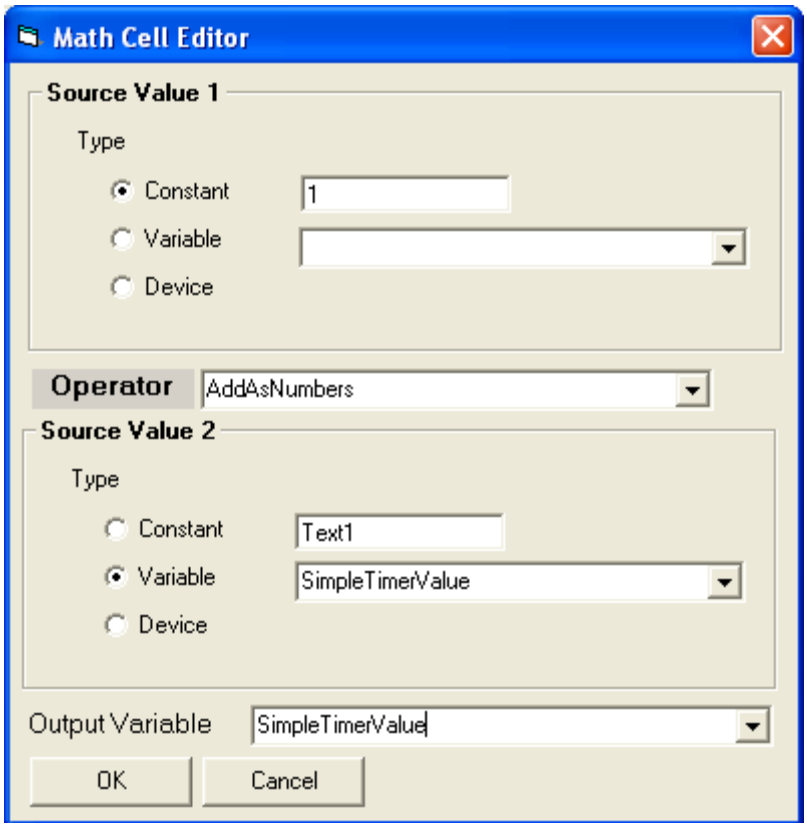
Try clicking on any cell except the upper left cell. Nothing will happen. Now click on the upper left cell, you will find that it is active and allows you to choose what type of cell it is to be.



Use the Type drop down to select Math, and name the cell *Increment.* Then select OK.



Now the upper left cell is displaying the Math cell icon, a calculator. Note that the second cell is available for use. Double click on the math cell to set it's configuration.

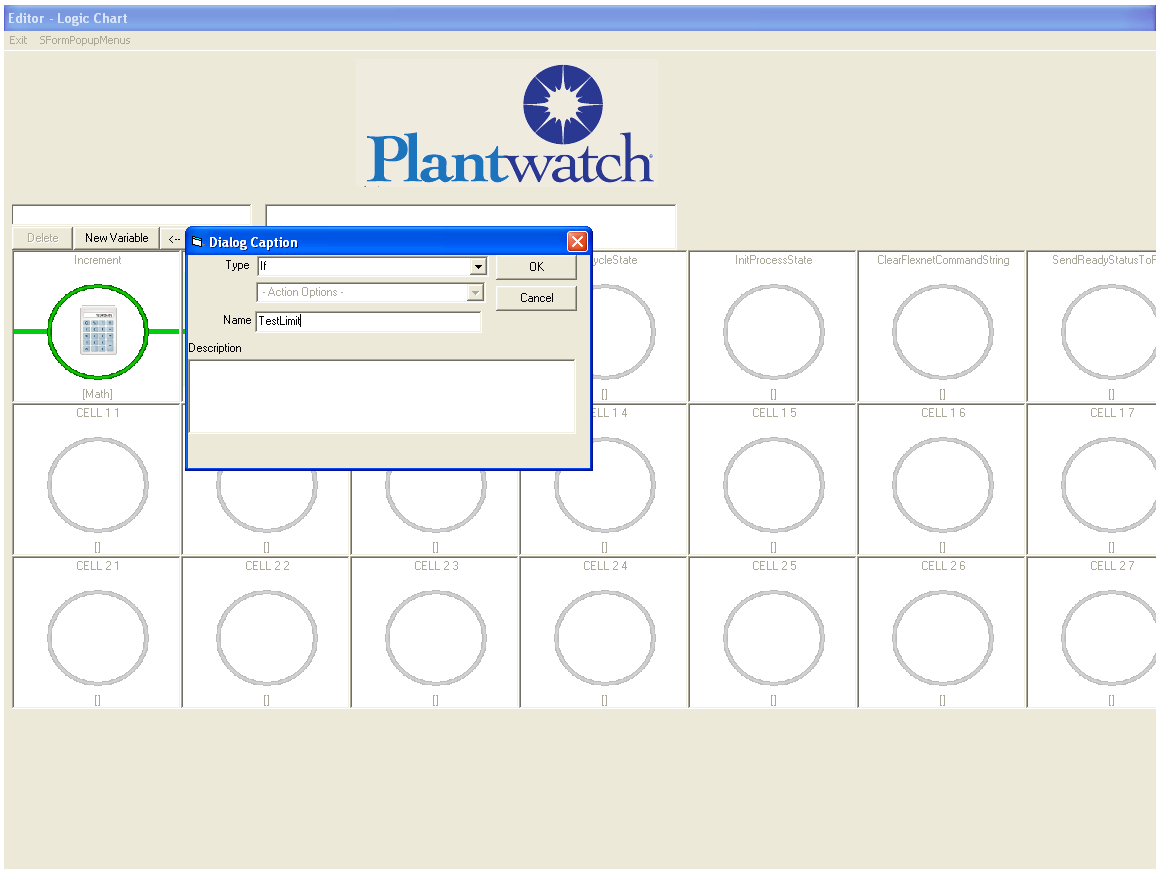


You will see the Math cell editor. We want our first cell to add one to the value in variable SimpleTimerValue. We will do this by:

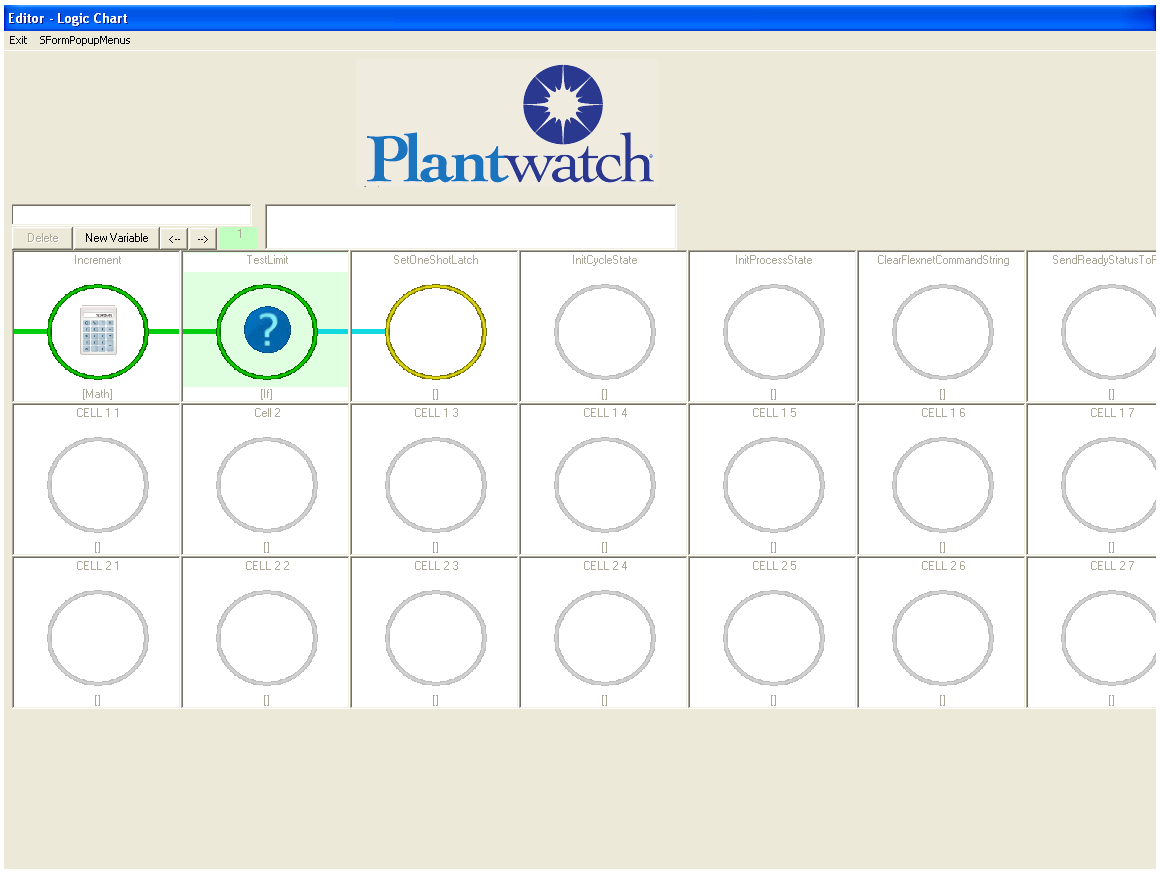
setting a constant value of one for the Source Value 1
selecting the AddAsNumbers operator
selecting the variable SimpleTimerValue for Source Value 2
selecting the variable SimpleTimerValue for Output Variable

This is telling the math cell to take the constant 1, add it as a number to the value in variable SimpleTimerValue and place the result into variable SimpleTimerValue.

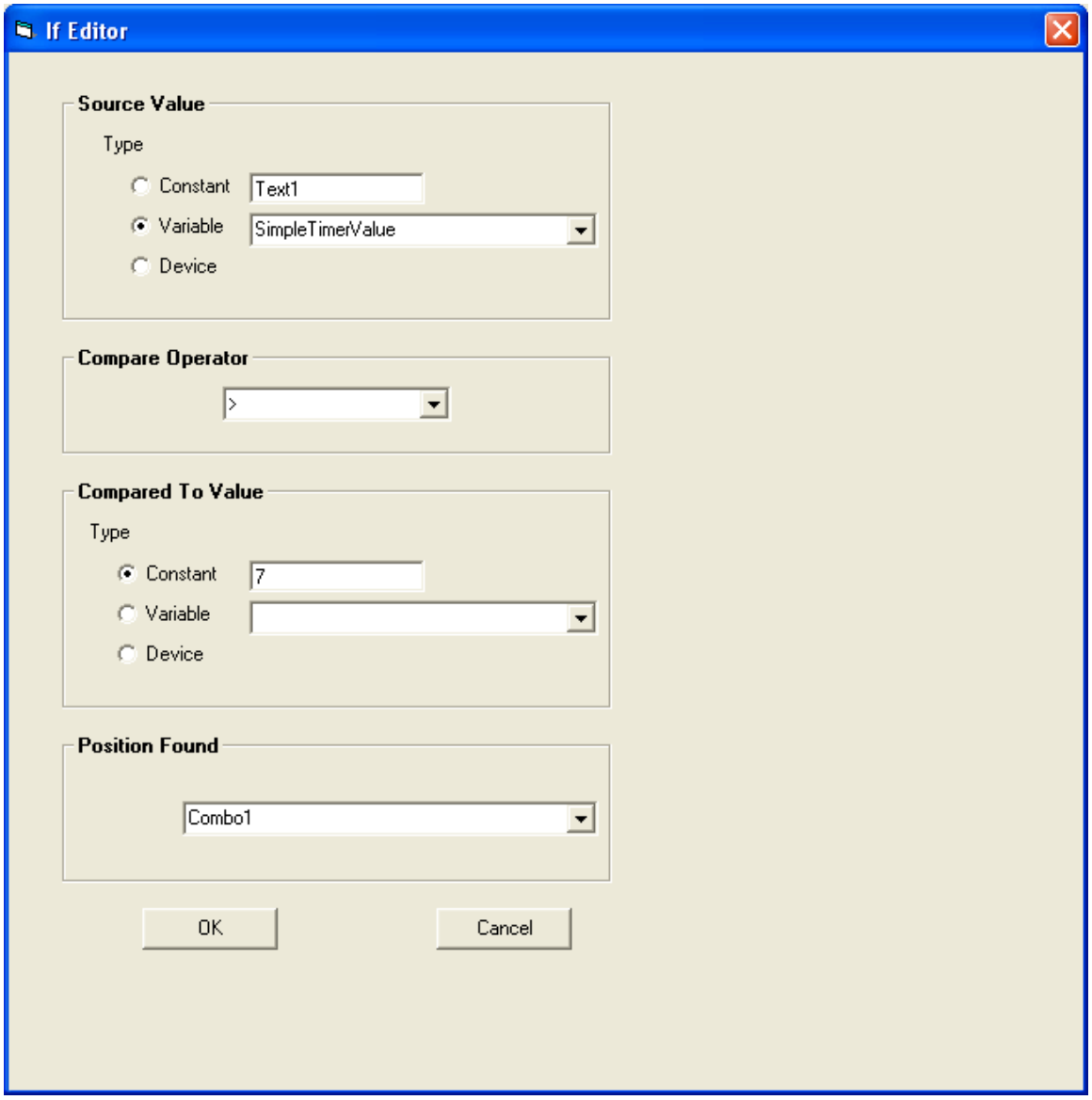
Click on OK to save and exit back to the logic chart.



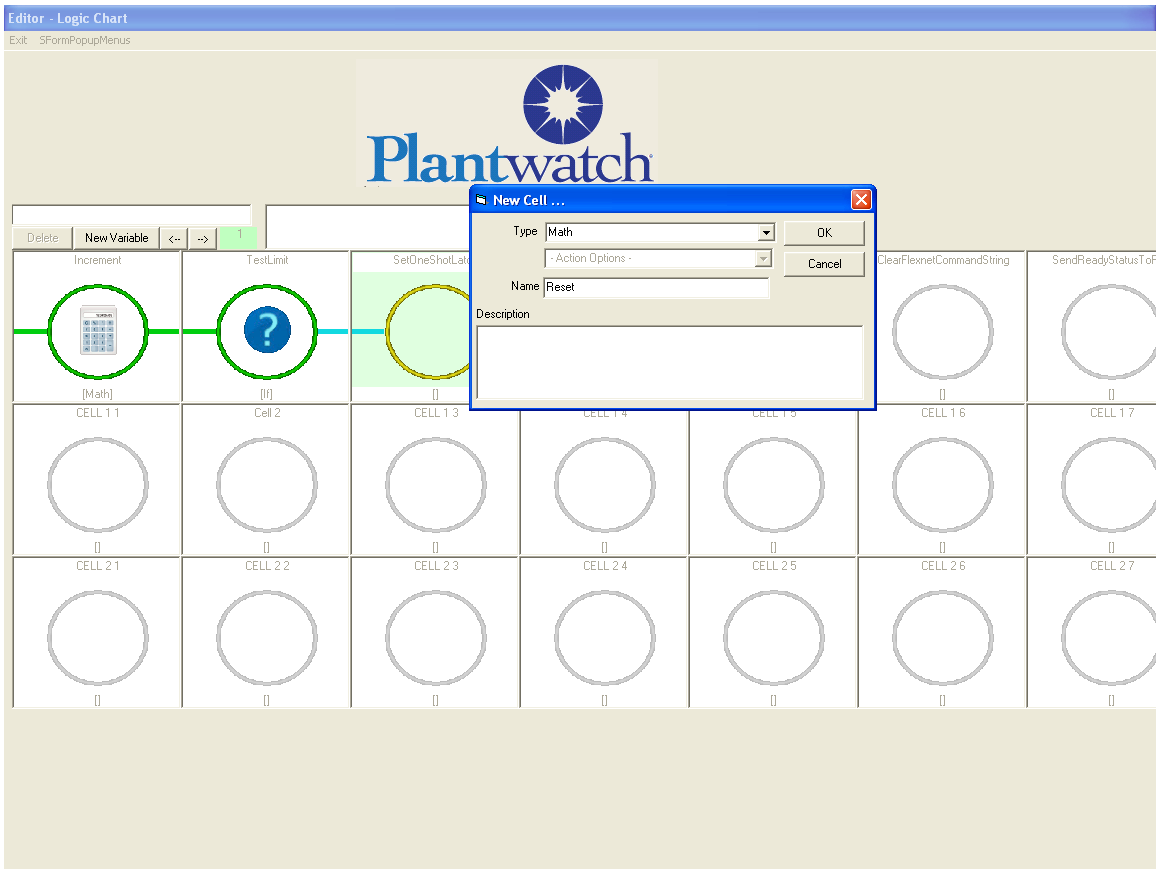
Double click on the second cell and set it's type to IF and it's name to TestLimit.



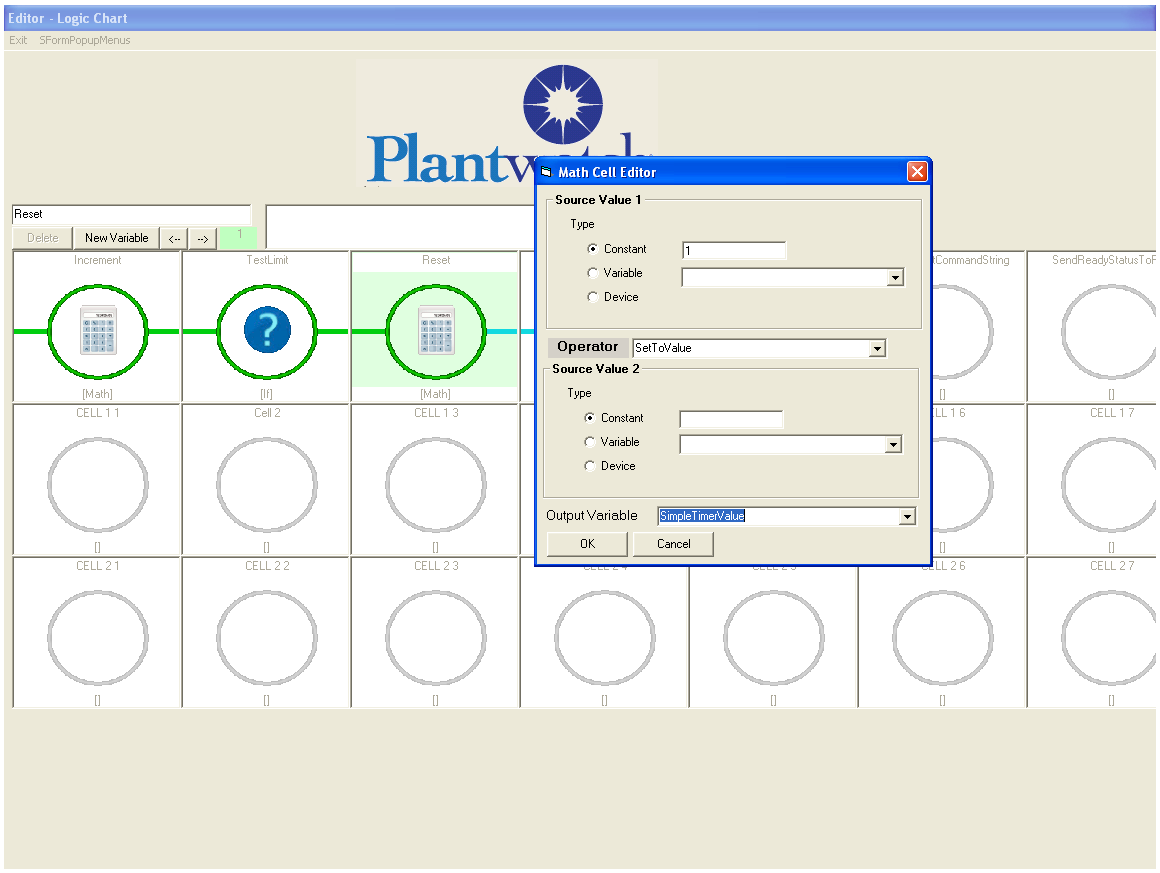
Now the second cell has the IF icon. Double click on the second cell to setup the IF.



We want it to only pass when we are to reset the timer back to 1, which is when SimpleTimerValue is at 7. To do this we will compare the current value of SimpleTimerValue to the constant 7. If it is greater than 7, the *If* will be true.



Save the settings and double click the third cell. Activate it as a Math cell with name Reset.



Double click on the third cell to configure the reset logic. We will use a math cell to set our counter, variable SimpleTimerValue, to one.

Set Source Value 1 to Constant Type with value of 1
Set Operator = SetToValue
Output Variable = SimpleTimerValue

Select *OK* to save your settings for the Math cell.

Select *Exit* from the top left of the window to exit the *Logic Chart Editor*.

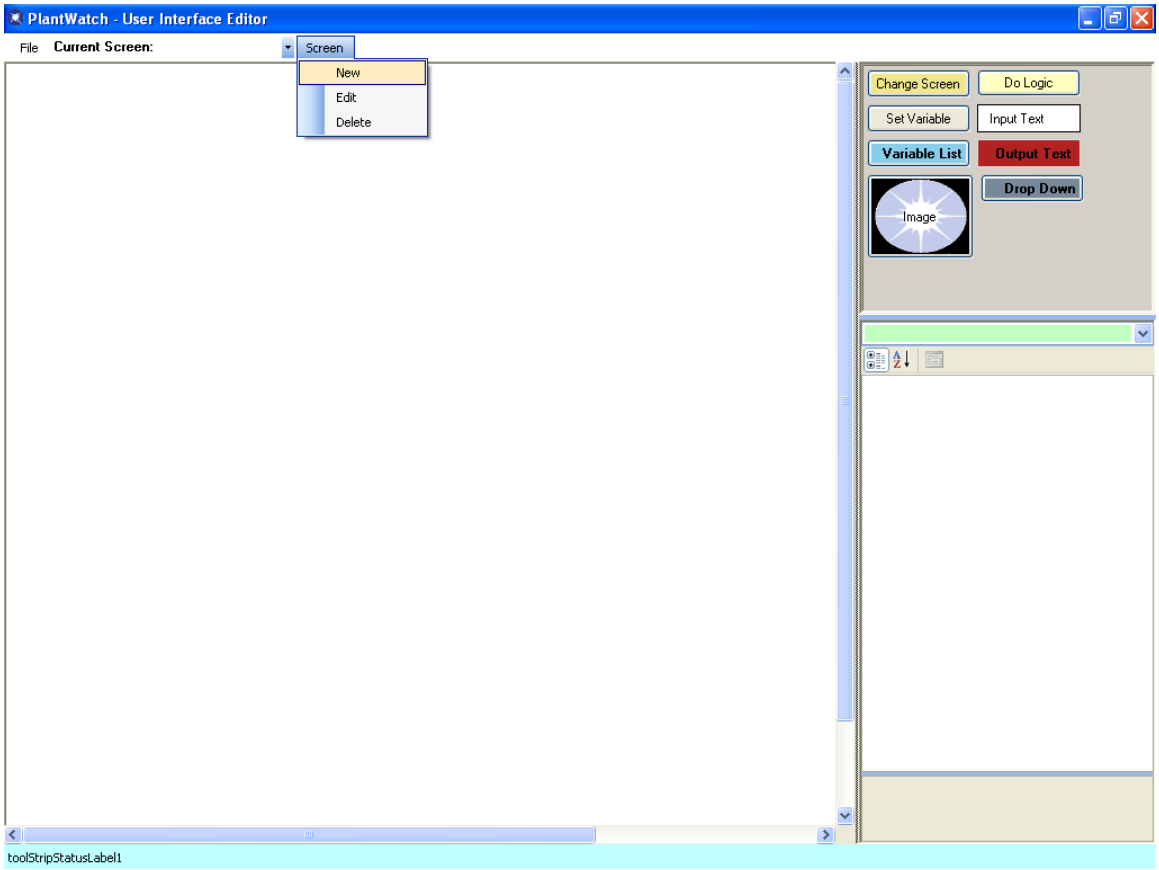
6. Save the current configuration
   From the Main screen of the Configurator select *File* and then select *Save*. Enter the password when prompted and then select *OK*. Default password is 123.

7. We will use Graphics to create a user screen to display the value contained in variable SimpleTimerValue.

PlantWatch Graphics support eight different types of animations including:

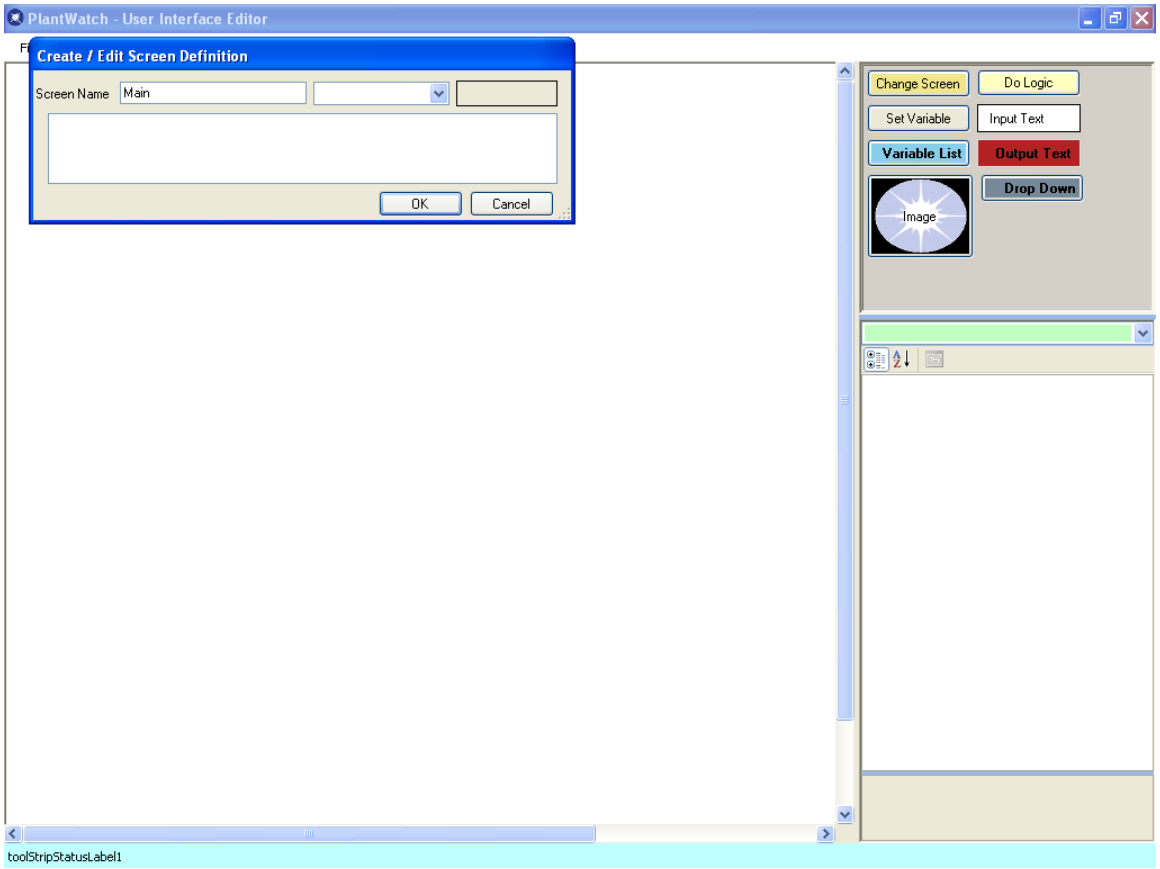
Buttons – To Cause actions
Output Text – To display fixed or variable content
Input Text – For data entry
Images – To display images as required

For our tutorial, we need to watch the variable SimpleTimerValue. This will require one *Output Text* animation associated with the variable SimpleTimerValue. To pretty things up a little, we will also add a label next to the *Output Text* animation. This will require one *Output Text* animation with a fixed content of 'Timer Value'. So for our screen we will have one *Output Text* animation presenting the value of *SimpleTimerValue* and the second *Output Text* animation presenting the constant text of 'Timer Value'.

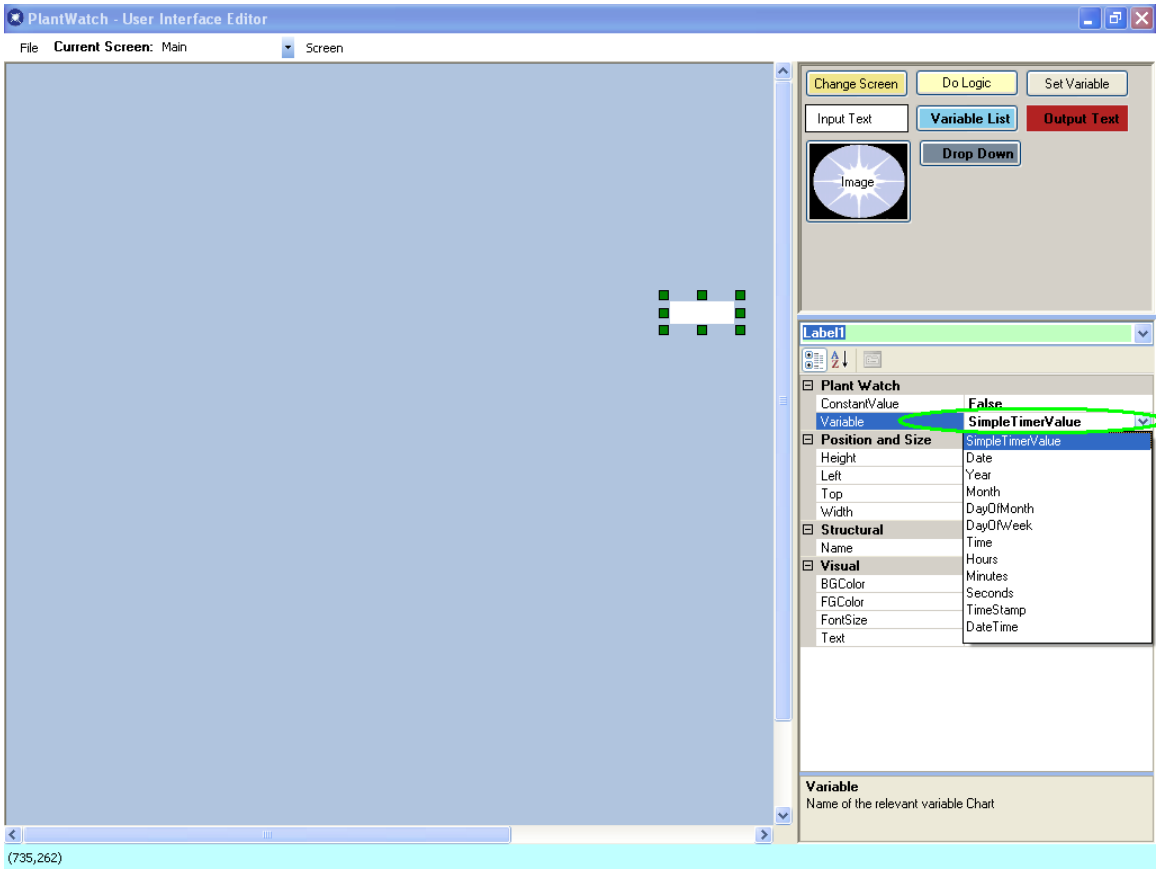


Open the PlantWatch Graphics Editor by selecting the desktop icon

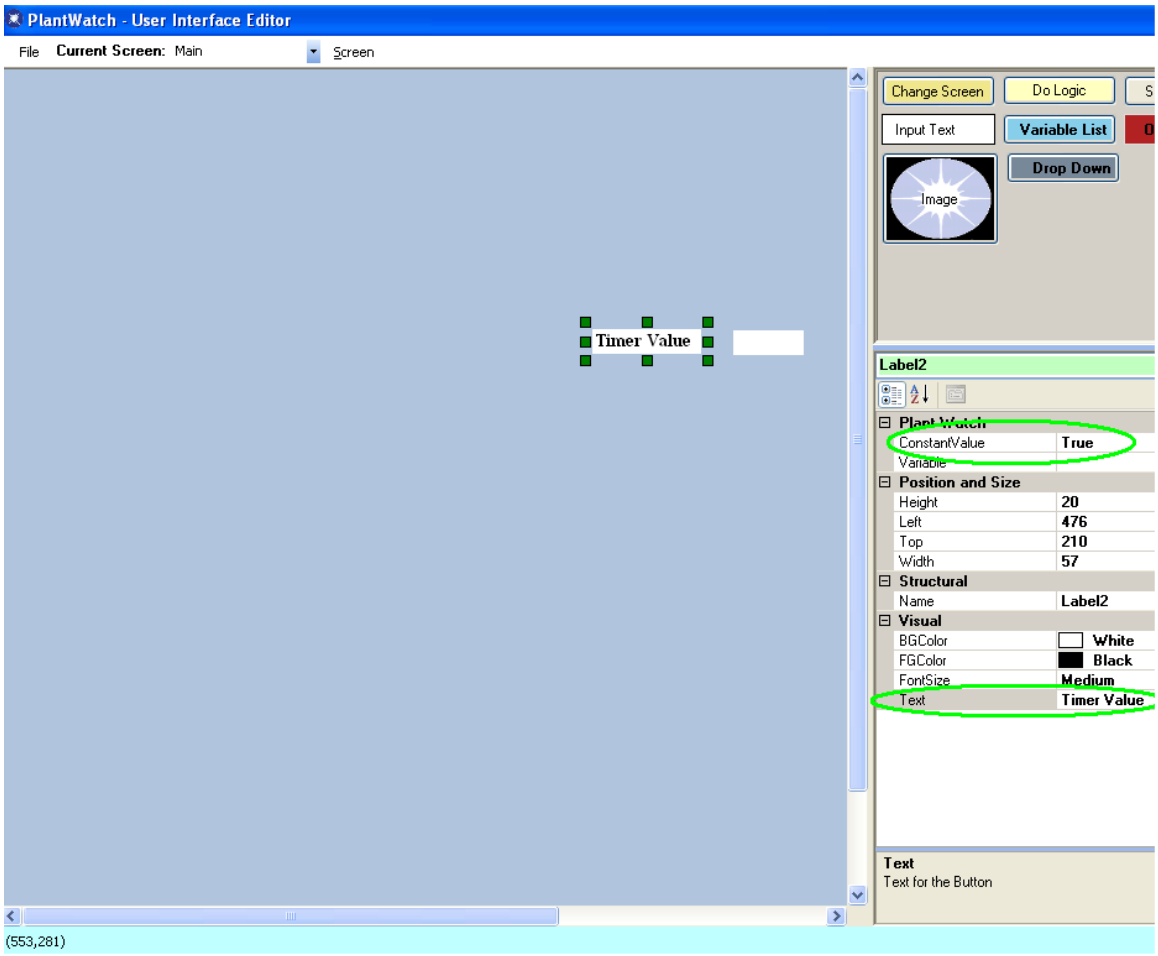
, select *Form* and then select *New*



Use Main for the Screen name and then select OK.



Drag an Output Text onto the form, and then select SimpleTimerValue for it's variable.



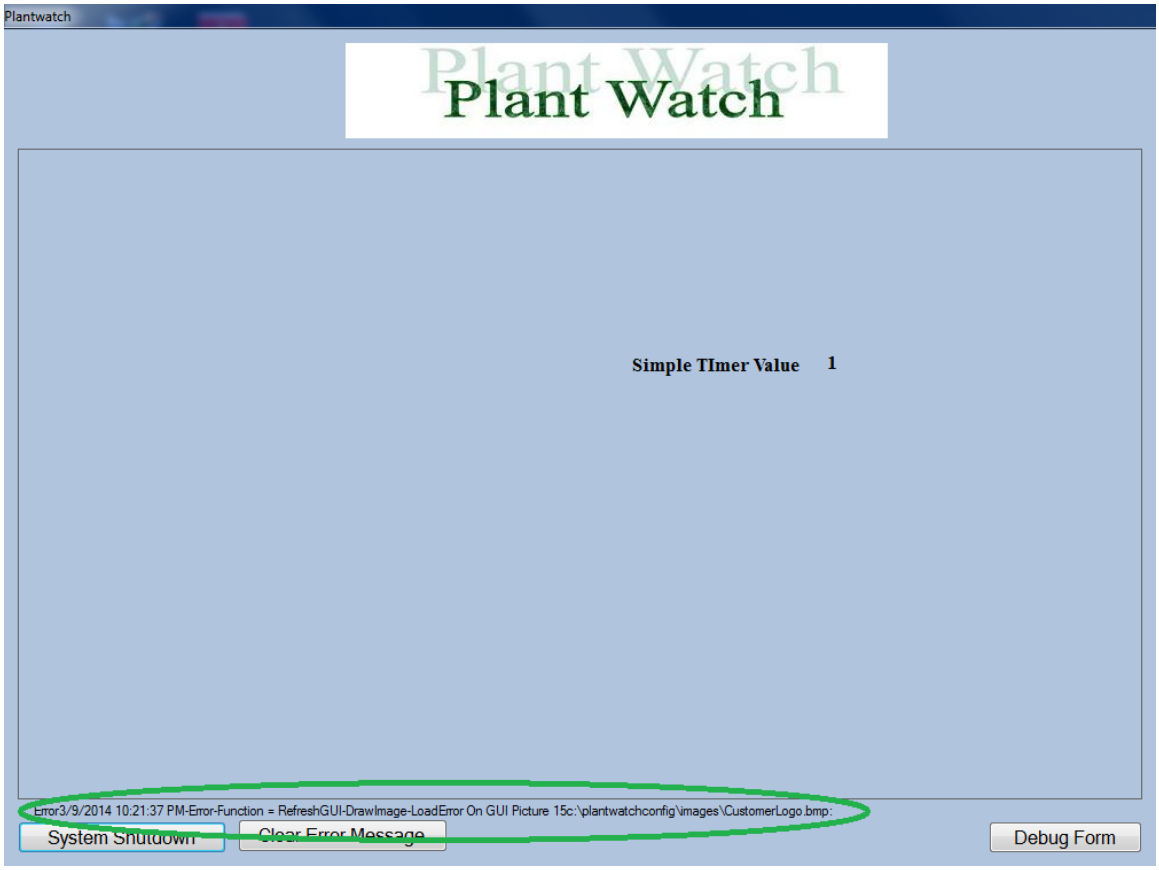
Drag another Output Text onto the screen. Set it's *Constant Value* property to True and it's *Text* property to 'Timer Value'

Save your new screen by selecting File and then Save. Now you can run the application.

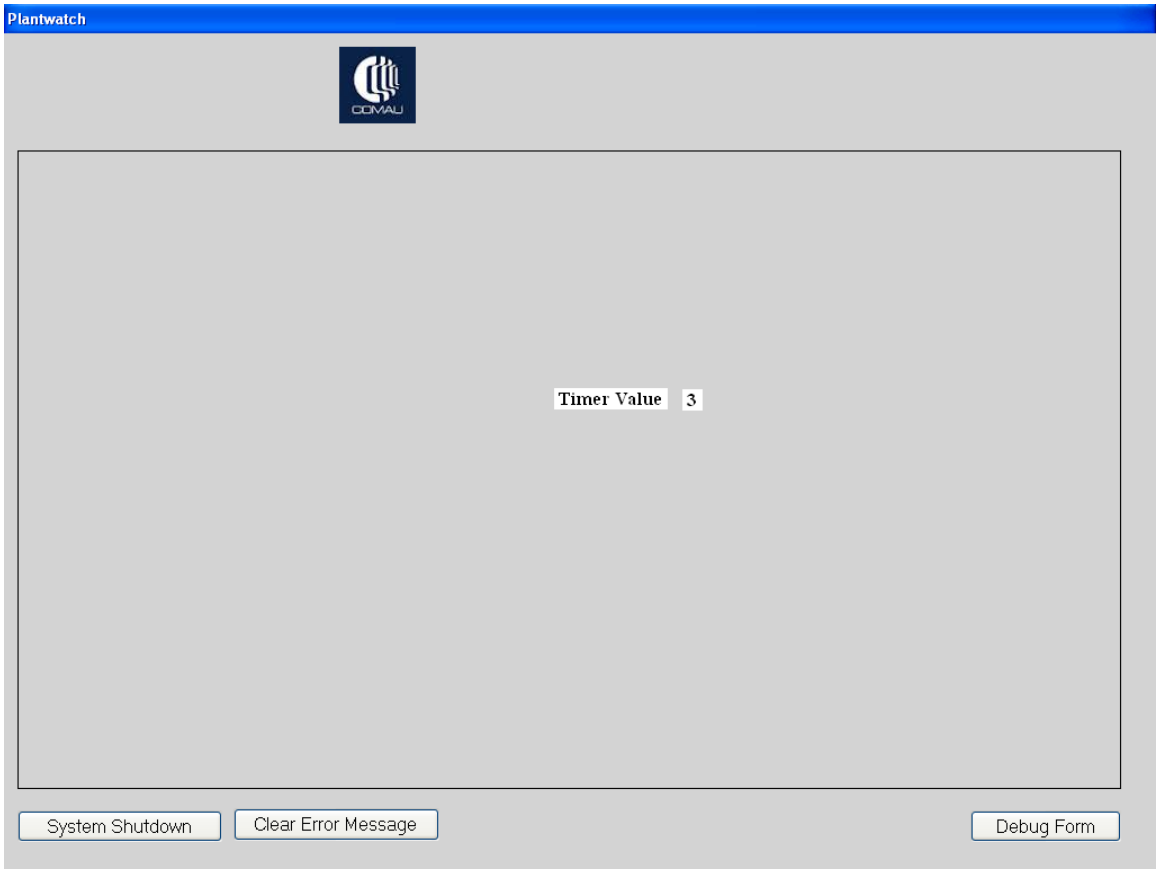
.

Now start up the runtime environment by double clicking on the Desktop icon labeled *PlantWatch Runtime*. After it starts you will see the value of SimpleTimerValue climbing up to 7 and then going back to 1.

You will also see an error message indicating that you have not set up your customer logo image.



PlantWatch expects that you want your image at the top of the screen. To put your logo there simple create a file named CustomerLogo.bmp and place it into the c:\PlantWatchConfig\Images folder and that image will be presented at the top of the screen like the Camau logo below.



To shut off *PlantWatch Runtime* select the *System Shutdown* button, enter the password and then select the *Shutdown* button. Default password is 123.

Devices

Devices are used within PW to bring in information from external devices and control the execution of Logic Charts. Within a Logic Chart the value of the Device is available to be used in any Logic Cell.

A good example of a source of data for PW is a RS232 port that is connected to a barcode scanner. This scanner may be used to indicate where a part is in the production process. For this type of data, RS232, a *ComPort* device is needed.

Based on the data type, one of 5 different types of Devices will be used.

Data Types include:

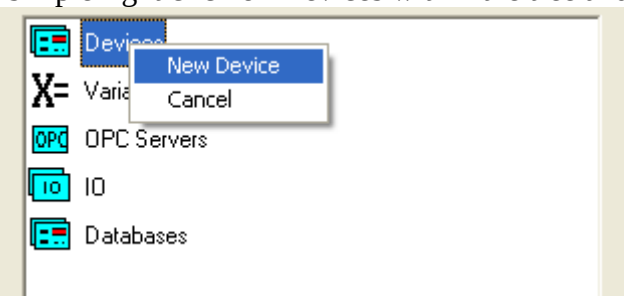
- Interval
- Socket
- ComPort

Variable
ReadFromFile

OPC is supported and will be covered later

Adding a New Device

To add a new device simple right click on Devices within the tree and select new Device



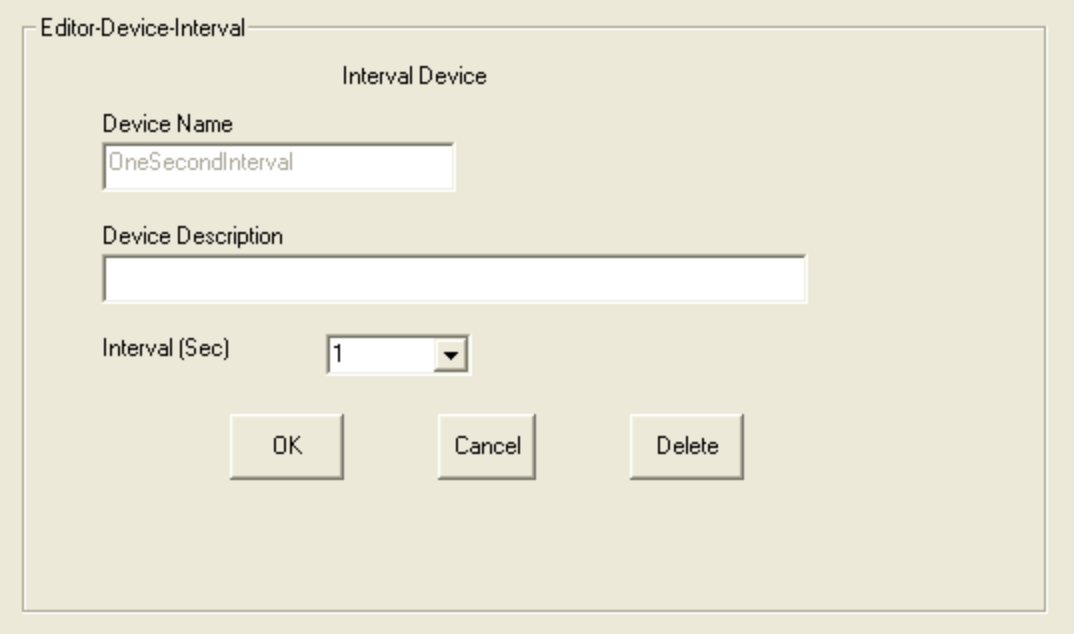
Adding a new device

Interval

One of the main functions of a device is to trigger Logic Charts. If there are any logic charts that need to run all of the time, an Interval type of device is used. The Interval type device will fire at the interval set in it's configuration. This results in an easy way to consistently trigger Logic Charts.

Configuration

For the Seconds entry, choose from intervals of .1, 1, 5, 15, 30 and 60 seconds



Socket

If the data is coming from a TCP socket host, you will use a Socket device. The PW Socket device will allow you to watch for and react to TCPIP data streams as the client.

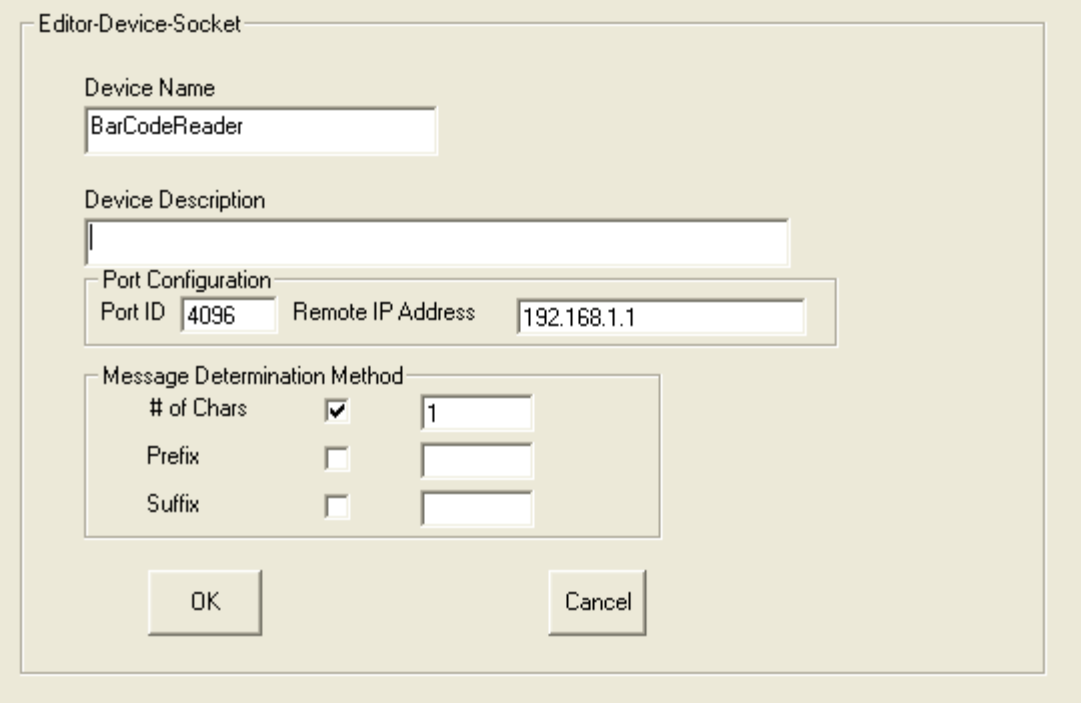
Configuration

Enter a name that makes sense for the *Device Name* field.

For the Remote IP Address entry, enter the IP of the network device you wish to communicate to.

For the Port ID entry, enter the chosen Port ID value. Both systems must use the same Port ID.

An additional type of configuration is called the Message Determination Method, or MDM. This applies to several devices and is covered later in the manual.



COMPort

To accept data coming from a serial port via RS232, you will use a COMPort device. The COMPort device will allow you to watch for and react to RS232 data streams.

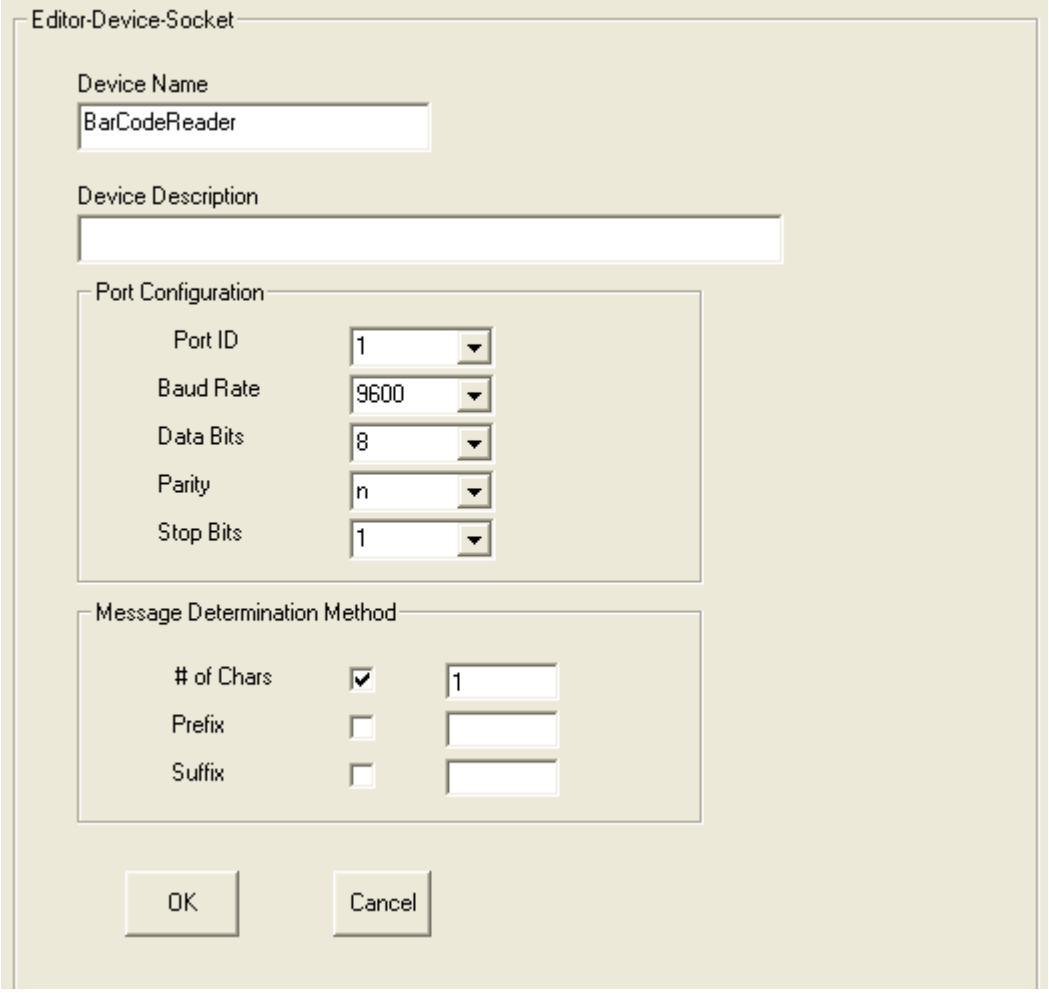
Configuration

Enter a name that makes sense for the *Device Name* field.

For Port ID enter the comm. port number of the port that the device is connected to.

For Port Configuration enter the com port number, baud rate, data bits, parity and stop bits.

An additional type of configuration is called the Message Determination Method, or MDM. This applies to several devices and is covered later in the manual.



Message Determination Method (MDM) in SOCKETs and COMPorts

Overview

The purpose of MDM is to provide a method to determine what makes a data message.

This is often necessary to make certain that data messages get processed as one message, not broken down into several smaller ones. Or to break one transmission down into several messages.

Within PlantWatch you can configure devices to use several methods to correctly assemble your messages.

- Look for at least 1 character, and process everything that is delivered
- Look for a specific number of characters
- Look for a specific starting character(s) and ending character(s)
- Look for a specific ending character

Explanation of why it is often needed:

When a RS232 barcode device reads a barcode and sends it's value into the computer thru the com port it does so one character at a time until all of it's data has been sent. This takes time. Depending on how quickly the com port is checked for data present, you could find the com port with only part of the data the barcode device was to send. In this case the next time you checked the com port for data present you would likely find the rest of the data the barcode device was to send.

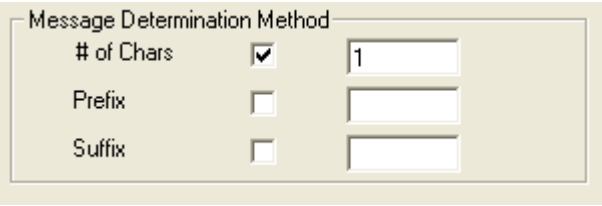
This could look like two separate barcode reads and if processed that way would result in corrupted data.

Another way a data message could be broke into two parts is if a large amount of data is sent. If the amount of data sent is greater than the incoming buffer of the communication layer, the communication layer will break it into manageable size chunks as it is processed. Unless properly handled, this will look like two transactions.

Another way a data message could fail is if the sending device sent two messages so close together in time that they looked like one transaction.

Usage

There are three options that can be selected. If no option is selected it will process all data as one transaction when ever data is present.



# of Chars (Number of Characters)

Causes the system to accumulate data in the incoming buffer until there are at least the configured number of characters. Any characters beyond the specified number will be left in the incoming buffer to be added to until there are enough characters to process as a transaction.

To configure select *# of Chars* and enter a number that represents the length of the data message that you will receive.

Prefix and Suffix

Causes the system to accumulate data from the incoming stream until the pattern for both the prefix and suffix is found, with the prefix found prior to the suffix.

Note that the system will discard the data in the incoming stream prior to the position where the pattern specified for the prefix is found.

To configure select Prefix and Suffix and then enter the character patterns expected.

Suffix Only

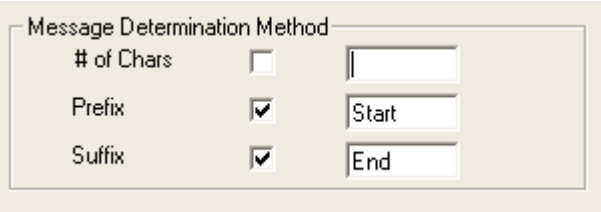
Causes the system to accumulate data in the incoming stream until the pattern specified for the Suffix is found. At that time all data up to the *Suffix* will be processed as one transmission.

To configure select Suffix and then enter the character pattern expected.

Non Printable ASCII Characters

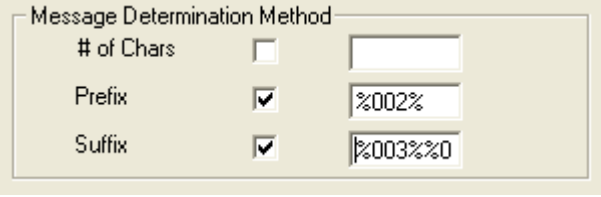
If the expected character pattern includes non printable ascii characters you can enter the expected character pattern using the decimal representation of the characters you need. To use the decimal representation, for each character needed place a percent character in from of and after the decimal number that represents that character.

Example 1:
The message always starts with 'Start'and ends with 'End'.



Configuration

Example 2:
The message always starts with ascii decimal 2, stx, and ends with ascii decimal 3, etx, followed by a carriage return, decimal 13.



Configuration

(Suffix content is %002%%013%)

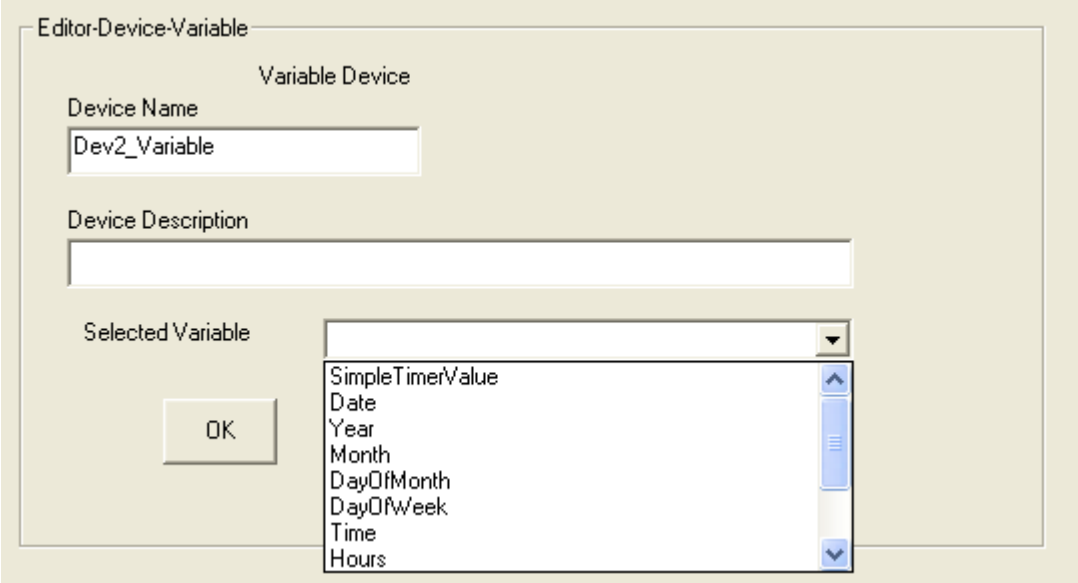
Variable

If you wish to react to a PW variable changing, you will use a *Variable* device. The PW Variable device will allow you to watch for and react to a change in the value of a PW Variable. For the device value it will use the value of the variable.

Configuration

Enter a name that makes sense for the *Device Name* field.

For Selected Variable choose the Variable to watch.



ReadFromFile

If you wish to react to the existence of a file on some drive, you will use a ReadFromFile. The device will look at a specified location on the hard drive for the existence of a specified file. The PW ReadFromFile device will allow you to watch for and react to the sudden existence of the file. When it does exist, the file will be read and the device will execute all of it's associated Logic Charts. For the device value, it will be the content of the first line in the file.

Configuration

Enter a name that makes sense for the *Device Name* field.

For File Spec enter the path and file name to watch for.

For Search Method choose one of the three

- Is Equal To
- Is Present
- Is Not Null

The search method Is Equal To will look for the file, if it is there it will read the first line of the file. If the first line equals the Search String, the device will fire.

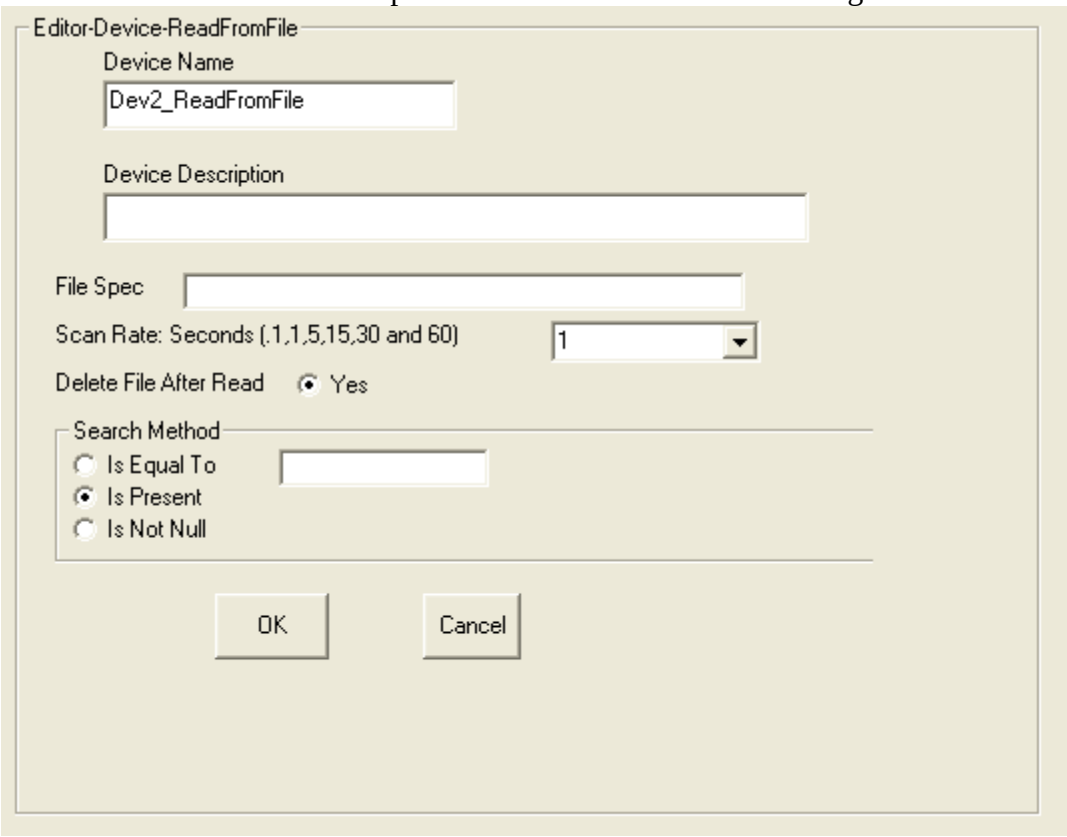
The search method Is Present will look for the file. If the file is there, regardless of content, the device will fire. For the device value it will use the first line of the text file.

The search method Is Not Null will look for the file. If the file is there, it will read the first line of the file. If there is anything in the first line of the file, the device will fire. For the device value it will use the first line of the text file.

If you choose the Search Method of Is Equal To you must enter in the string being searched for in the Search String configuration.

For Scan Rate choose 1,5,15,30 or 60 seconds as the rate it looks for the file.

Select the Delete After Read option to remove the file after looking at it.



Logic Charts

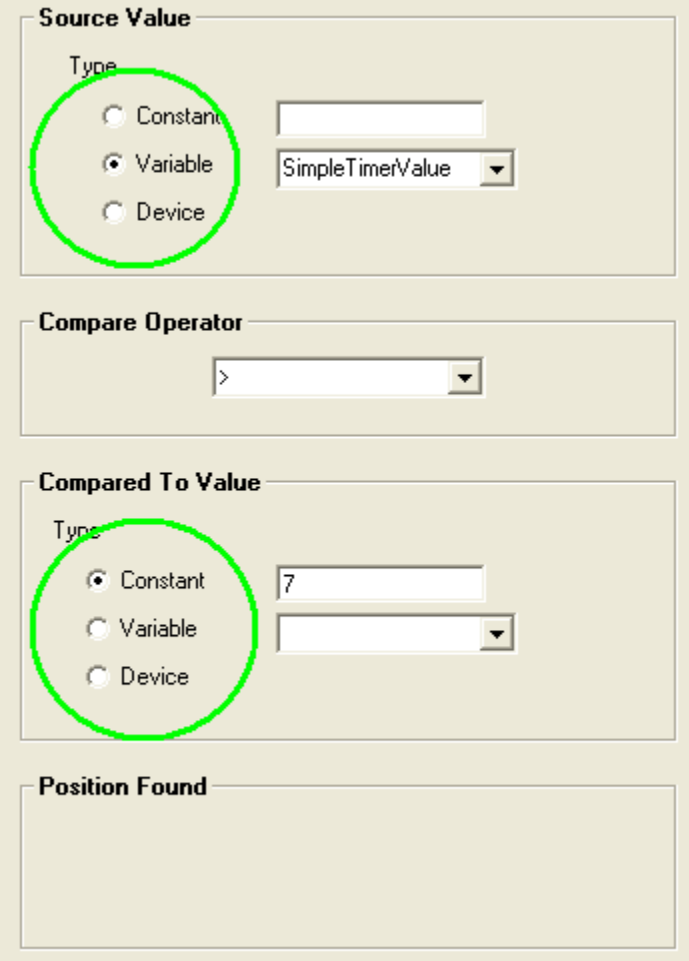
Each Logic Chart is a collection of configurable cells and is associated with a specific *Device*. Each time the associated *Device* is triggered all of the Logic Charts associated with the *Device* are executed.

Logic Chart Value Types

PW supports 3 different types of data within the Logic Charts

- Constant
- Variable
- Device

These three types of data allow you to create the logic that will drive the execution of actions within the Logic Charts. Looking at the configuration for an *If Cell* below we see that for both the Source Value and the Compare To Value there are selection buttons to choose one of the three data types.



This is common amongst most of the Logic Chart configuration panels. This allows you to choose between the three data types as you create your configuration within the Logic Charts.

Data Type Definition:

Constant – A value that never changes typed in by the person configuring the panel

Variable – The current value of the variable selected

Device – The value of the device that the chart is associated with.

Logic Chart Cell Types

Overview

- If – Determines if the logic cells that follow the *If* will be executed
- If Then – Determines if the logic defined within the *If Then* cell is to be executed
- Birth Tracks the creation of a new part within the production system
- ConsumeMaterial Tracks what materials have been consumed by parts in production
- UnconsumeMaterialAction Returns what materials have been removed from parts
- CollectDataPoint Associates a value to one of the parts within the production system
- WriteToDevice Writes a value to the datasouce used as a device
- WriteToIO Writes to a output point on the rack of IO
- WriuteToOPC Writes to a OPC data item
- WriteTOFile Writes a value to a text file
- ReadFromFile Reads a value from a text file
- TriggerExe Causes a executable file to be started
- Split – Creates a branch in the flow of logic by enabling the cells below the *Split*
- Math – Takes several values from constants, devices or variables, performs a mathematical operation on them and then places the results into a Variable that can be used elsewhere.
- File Manager – Allows interaction with the file system to copy, rename delete files.
- Database Browser – allows bi directional communication with a SQL database
- Substring – extracts a part of a string out of a *Device* or *Variable*
- UnUsed – If you want to delete a cell, you choose the UnUsed cell type.

If Cell

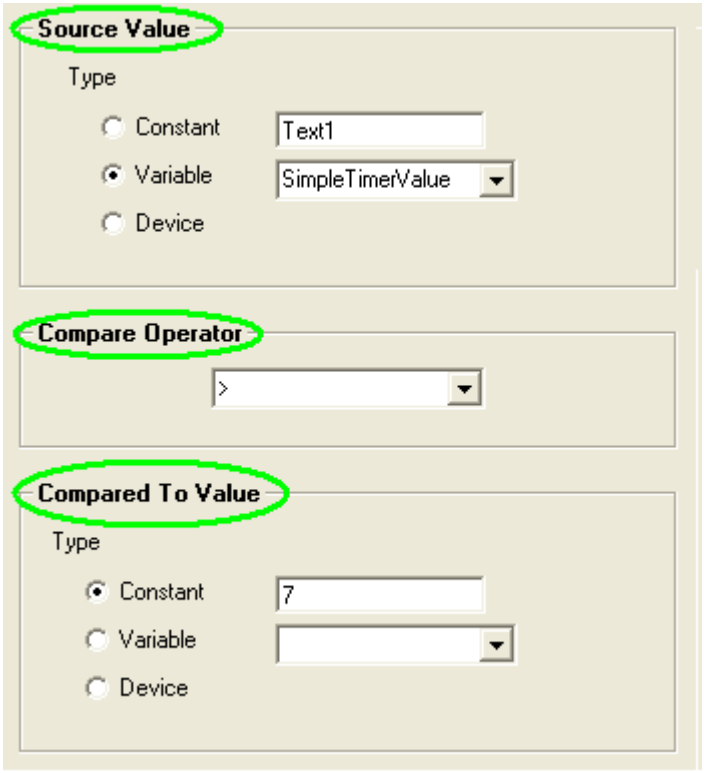
An *If Cell* is used to control logic flow within the logic chart. It will evaluate an expression and if the result is true will allow the cells following the *If* to execute. An example of the type of logic an *If Cell* evaluates is

Is variable SimpleTimerValue greater than 7?

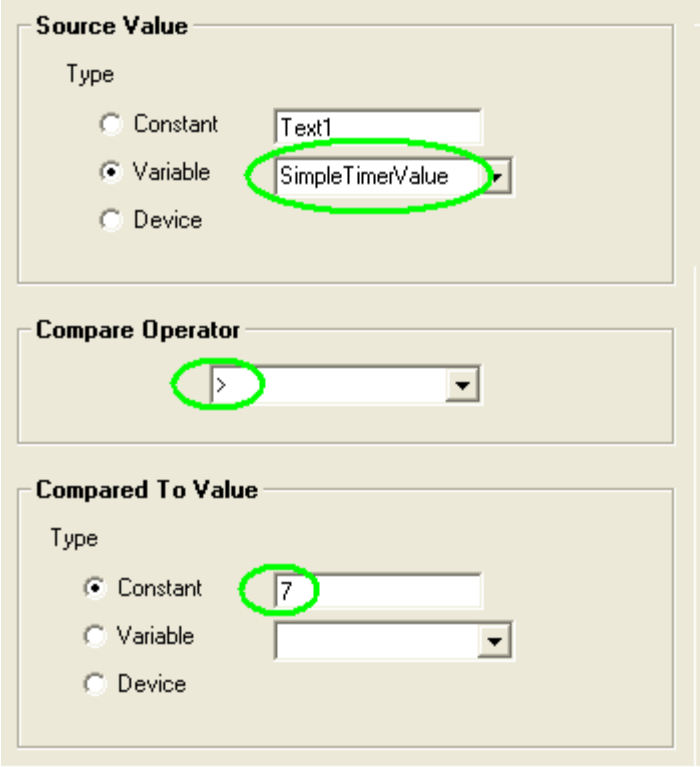
To create an expression you utilize

Source Value
Compare Operator
ComparedToValue

The *Source Value* will be compared to the *Compare To Value* with the operator selected in the *Compare Operator* field.



For our example SimpleTimerValue is the SourceValue, GreaterThan is the CompareOperator and the constant 7 is the CompareToValue.



Source Value

Source Value is the information that will be compared to the *Compare To Value*. *Source Value* has available to it all three of the common PW value types of Constant, Variable and Device.

Compare Operator

The Compare Operator determines what type of comparison is made between the Source and Compare To values. The list of operator includes:

>
<
<>
DoesNotContain
Contains
LogicalOr2Tags
LogicalAnd2Tags

If you select =, the expression will be true if the *Source Value* is equal to the *Compare To Value*

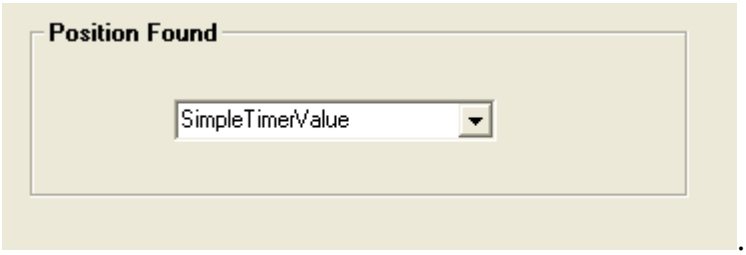
If you select >, the expression will be true if the *Source Value* is greater than the *Compare To Value*

If you select <, the expression will be true if the *Source Value* is less than the *Compare To Value*

If you select <>, the expression will be true if the *Source Value* is not equal to the *Compare To Value*

If you select *DoesNotContain,* the expression will be true if the *Source Value* does not contain the *Compare To Value,* with both values interpreted as a string.

If you select *Contains,* the expression will be true if the *Source Value* contains the *Compare To Value,* with both values interpreted as a string. This particular compare operator also allows you to capture the location in the source value where the string was found by using the Position Found. Select a variable for Position Found and that variable will be set to the location that the compare to value was found.



If you select LogicalOr2Tags or LogicalAnd2Tags, the panel will change and instead of Compare To Value, you will see Source Variable 2. Instead of Source Value, you will see Source Variable 1.

If you select *LogicalOr2Tags,* the expression will be true if the *Source Value 1* or *Source Value 2* is true, with true being a value of non zero.

If you select *LogicalAnd2Tags,* the expression will be true if the *Source Value 1* and *Source Value 2* is true, with true being a value of non zero.

If Then Cell

An *IfThen Cell* is used to set a value into a PW variable if the expression is true. It does NOT control logic flow within the logic chart. It will evaluate an expression and if the result is true will set a value into a PW variable.

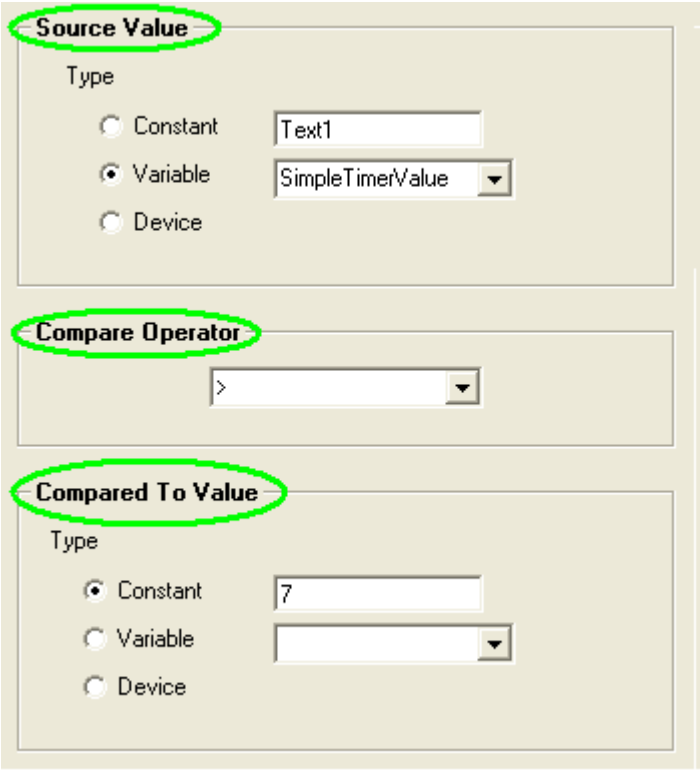
An example of the type of logic an *IfThen Cell* evaluates is

Is variable SimpleTimerValue greater than 7?

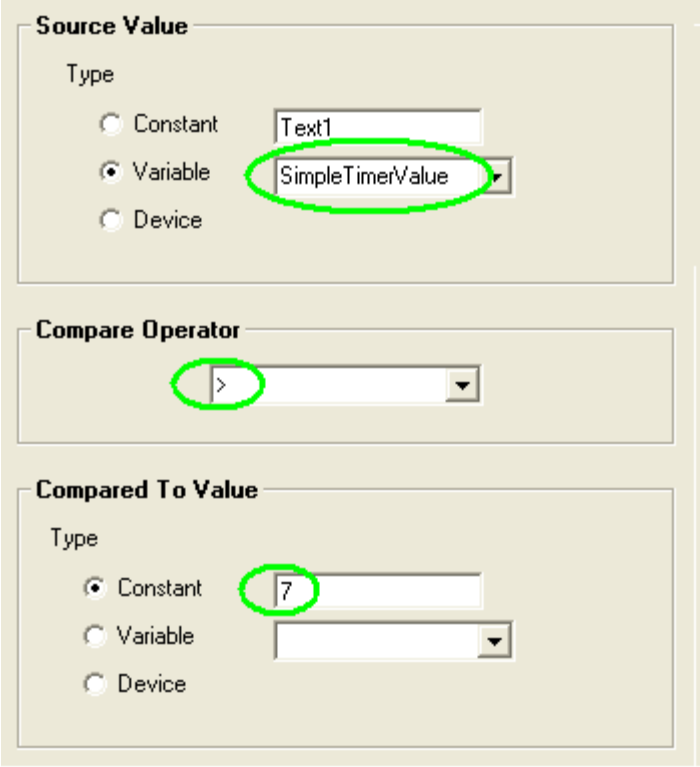
To create an expression you utilize

Source Value
Compare Operator
ComparedToValue

The *Source Value* will be compared to the *Compare To Value* with the operator selected in the *Compare Operator* field.



For our example SimpleTimerValue is the SourceValue, GreaterThan is the CompareOperator and the constant 7 is the CompareToValue.



Source Value

Source Value is the information that will be compared to the *Compare To Value*. *Source Value* has available to it all three of the common PW value types of Constant, Variable and Device.

Compare Operator

The Compare Operator determines what type of comparison is made between the Source and Compare To values. The list of operator includes:

>
<
<>
DoesNotContain
Contains
LogicalOr2Tags
LogicalAnd2Tags

If you select =, the expression will be true if the *Source Value* is equal to the *Compare To Value*

If you select >, the expression will be true if the *Source Value* is greater than the *Compare To Value*

If you select <, the expression will be true if the *Source Value* is less than the *Compare To Value*

If you select <>, the expression will be true if the *Source Value* is not equal to the *Compare To Value*

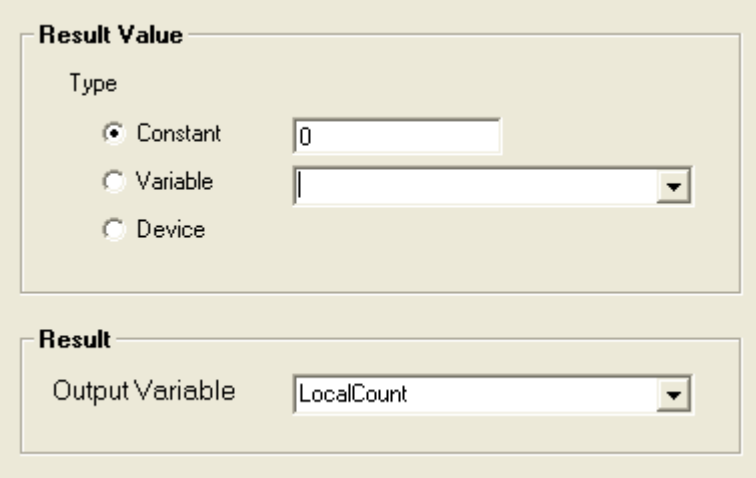
If you select *DoesNotContain,* the expression will be true if the *Source Value* does not contain the *Compare To Value,* with both values interpreted as a string.

If you select LogicalOr2Tags or LogicalAnd2Tags, the panel will change and instead of Compare To Value, you will see Source Variable 2. Instead of Source Value, you will see Source Variable 1.

If you select *LogicalOr2Tags,* the expression will be true if the *Source Value 1* or *Source Value 2* is true, with true being a value of non zero.

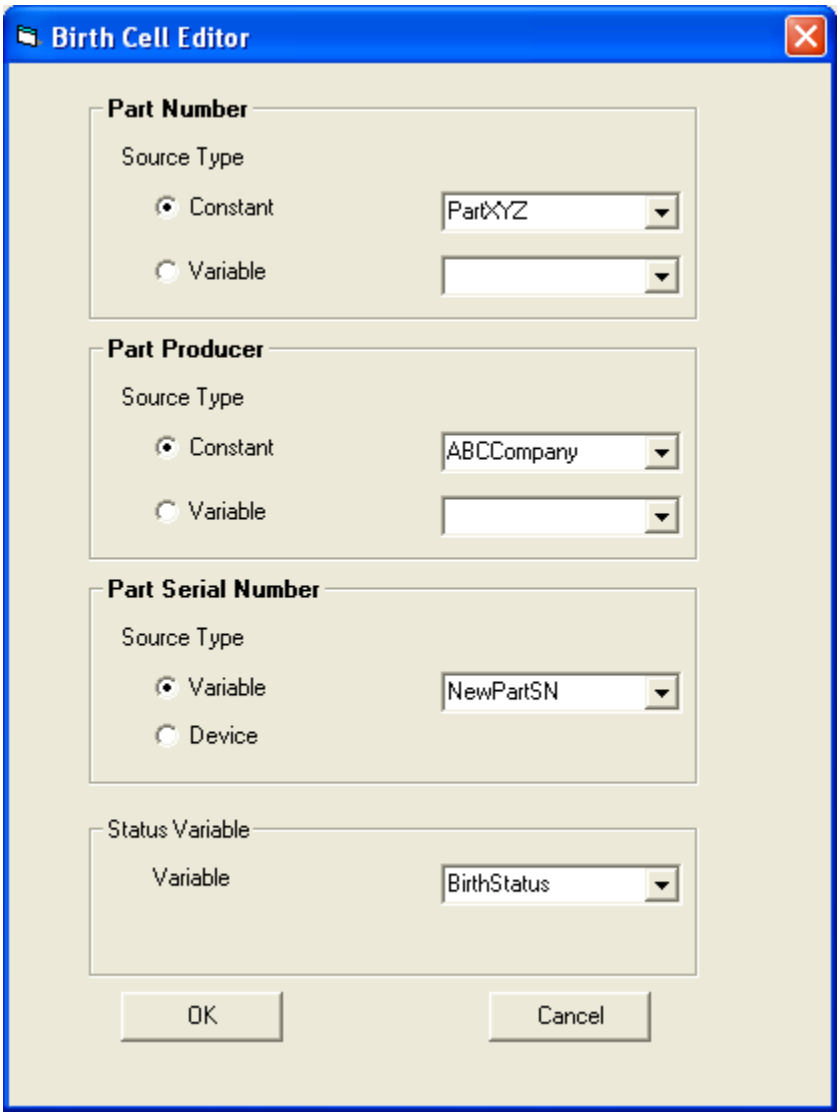
If you select *LogicalAnd2Tags,* the expression will be true if the *Source Value 1* and *Source Value 2* is true, with true being a value of non zero.

If the expression is determined to be True, the Result value will be placed into the Result Variable.



Birth Cell

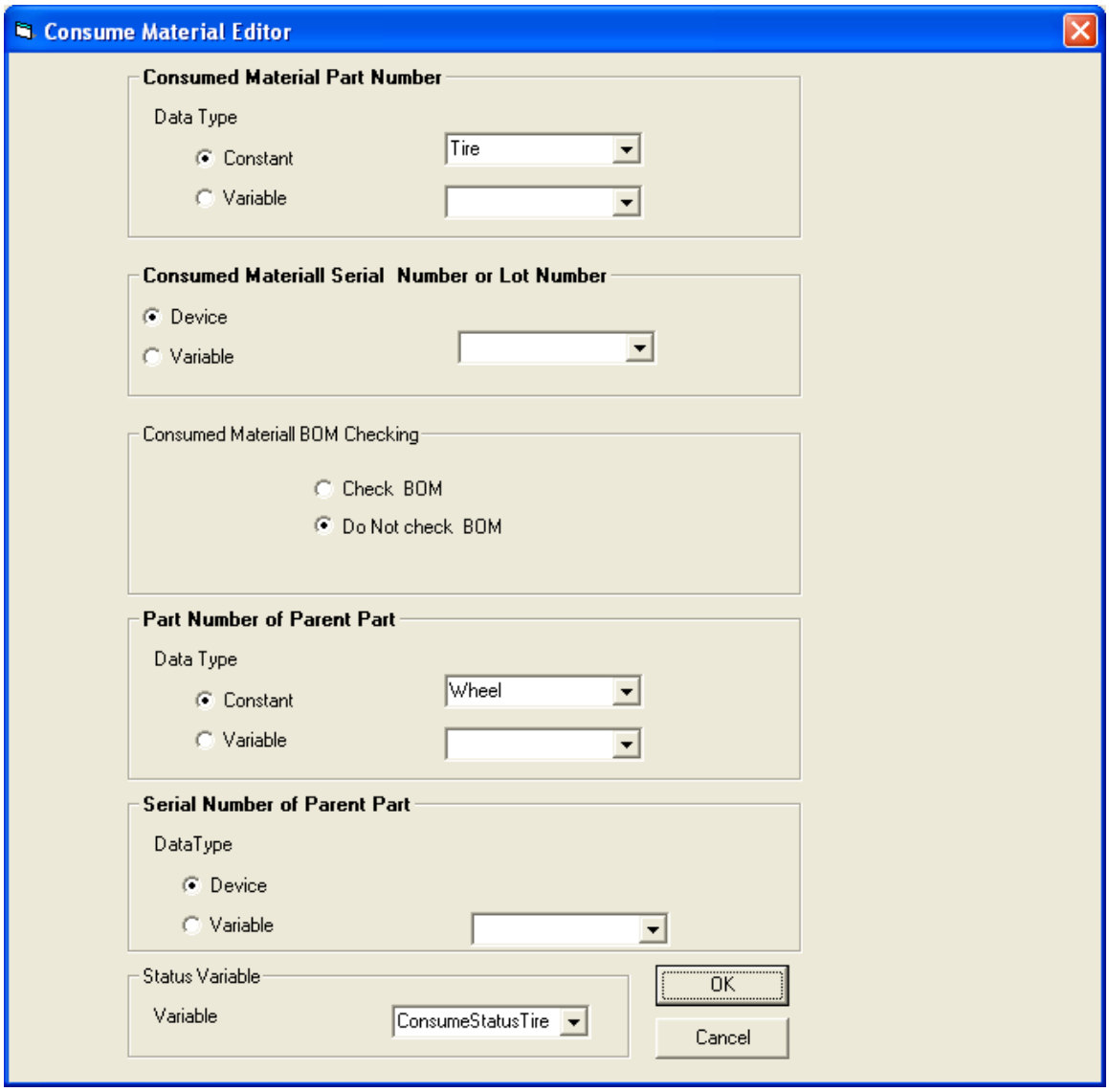
A Birth cell is used to track the creation of a unique part within the Plant. For a part to be Birthed, there must be an associated part number, part manufacturer and unique serial number. Both the part number and part manufacturer must be defined within the DB by using the *Part Number Setup* utility located in *Settings*.



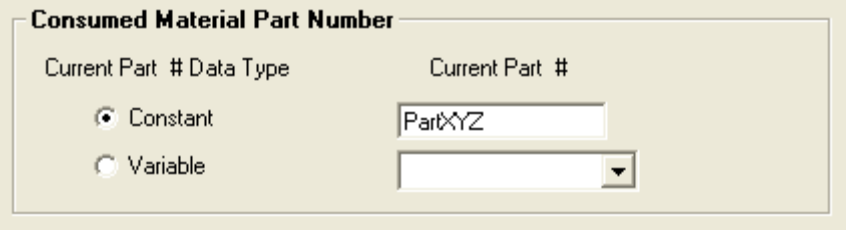
The Status Variable will provide feedback upon the activity requested. If the requested Birth occurred, the Status Variable will be set to “Pass”. If the Birth failed the Status Variable will be set to “Fail”

ConsumeMaterial Cell

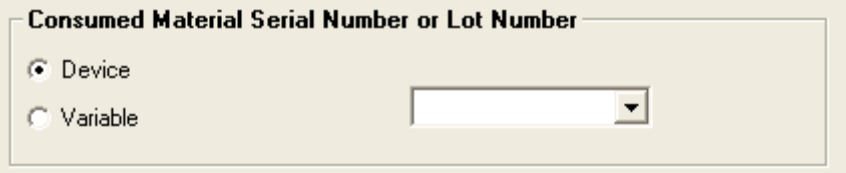
A *ConsumeMaterialCell* is used to track the raw materials and components that are used to create a finished good. All material consumed must be identified with a part number and their unique serial / lot number.



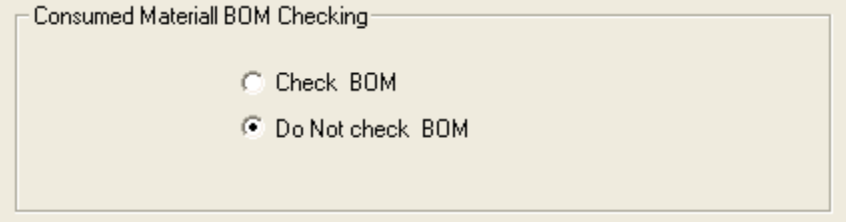
The *Consumed Material Part Number* panel must provide the part number of the material consumed. It can come in via a constant or a Variable. This must be a part number recognized within the DB.



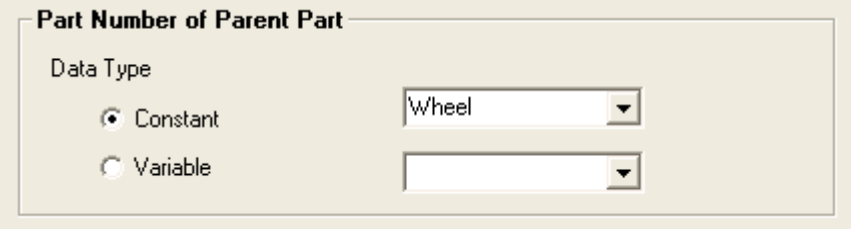
The *Consumed Material Serial Number or Lot Number* panel will supply the part number of the consumed material.



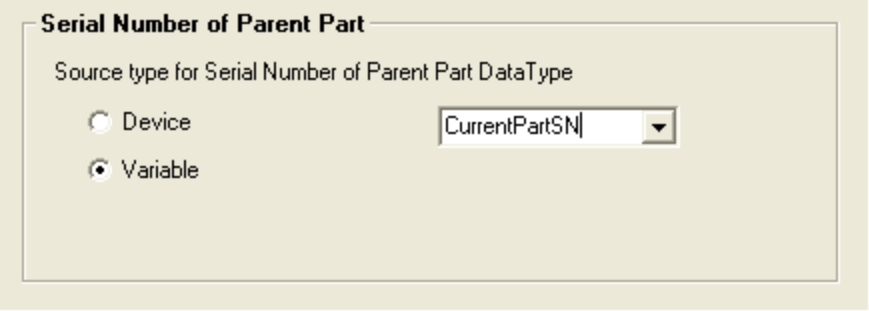
The ConsumedMaterial BOM Checking panel will determine if the BOM determines if this consume is allowed.



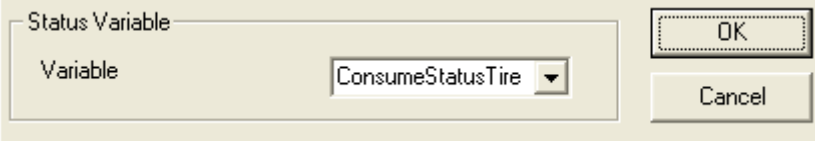
The *Partl Number of Parent Part* panel must provide the part number for the part that is consuming material.



The *Serial Number of Parent Part* panel must provide the unique serial number for the part that is consuming material.



The Status Variable will provide feedback upon the activity requested. If the requested Consume occurred, the Status Variable will be set to zero. If the Consume failed the Status Variable will be set to a non zero numeric value. The numeric value can be used to determine what failed by comparing it to the error codes listed below.




```
Error Code Definitions:
                  0 -- Success
                  -1-- Unknown Failure
                  1 -- Bad station
                  2 -- Bad Parent Part Number
                  3 -- Bad Parent Serial Number
                  4 -- Bad Consumed Material Part Number
                  5 -- Bad Consumed Material Serial Number
```

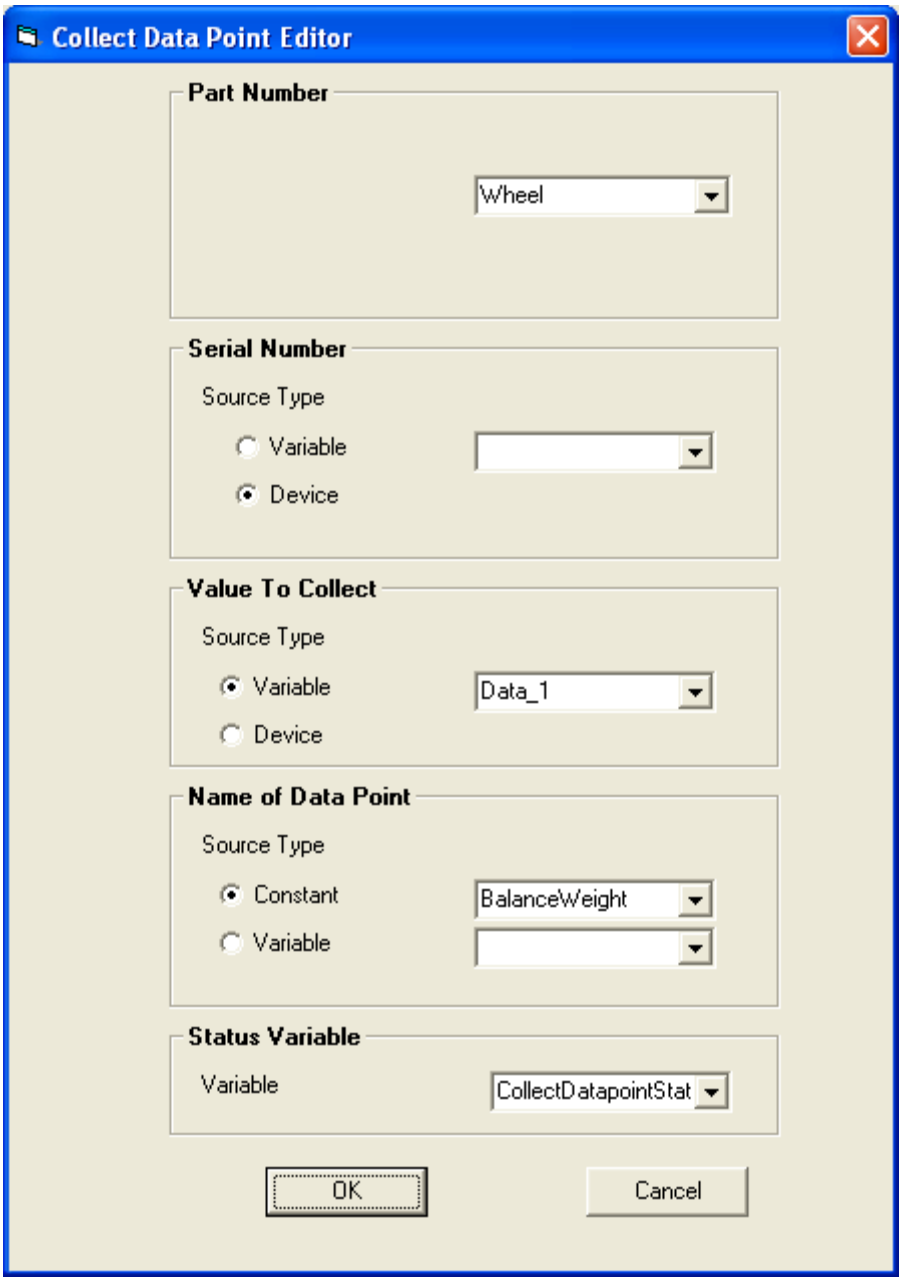
CollectDataPoint Cell

A *CollectDataPoint* cell is used to associate a piece of data to a part identified within the Database. The new piece of data will be associated to the new part identified by the name of a *Data Point* already configured within the DB. Data Points are configured with the Part Number Setup dialog.

To access the part number setup click on *Part Number Setup* in the *Setup* menu.

A *CollectDataPoint* cell has five fields required.

Part Number
Serial Number
Value to collect
Name of the Data Point
Name of the status variable



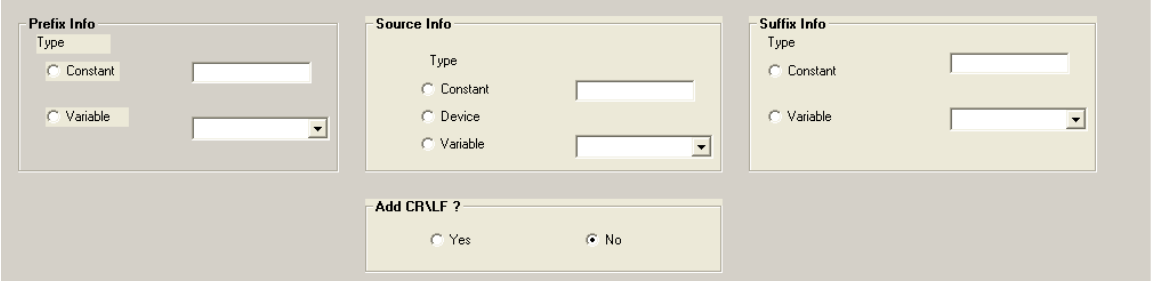
The Part Number is the part number of the part the data will be associated to
The Serial Number is the serial number of the part the data will be associated to
The Value to Collect is the value that will be saved within the database
The Name of Data Point is the ID of the data being saved
The Status variable is used to determine if the action succeeded.

WriteToDevice Cell

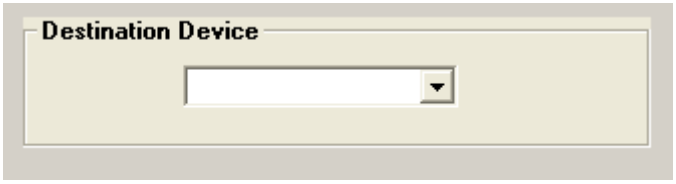
A *WriteToDevice* cell is used to send data out of PW. Devices are places where data can come from and be sent to. The Write To Device is used to send data to a PW Device. For example, if the Device chosen to write to is a ComPort type device, writing to the device will result in a data stream going out the com port.

A *WriteToDevice* cell has five fields, four dedicated to creating the data and one to choose the device.

For ease of creating complete data strings, there is a Prefix, Source and Suffix available to be used to create the desired string. The Prefix, Source and Suffix will be concatenated together to create what will be written to the device. An optional button allows you to add a carriage return and line feed to the data string.



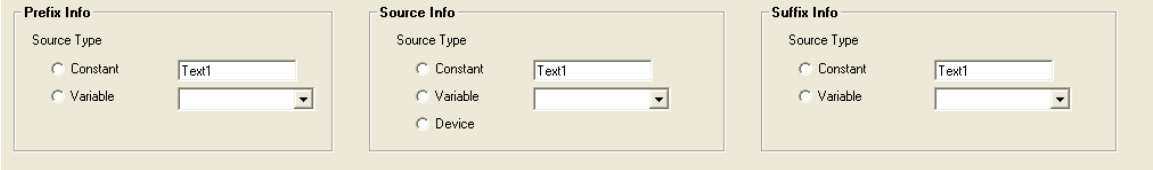
The Device that will be written to is selected via a drop down box.



WriteToOPC Cell

A *WriteToOPC* cell is used to send data out of PW to an OPC server.

For ease of creating complete data strings, there is a Prefix, Source and Suffix available to be used to create the desired string. The Prefix, Source and Suffix will be concatenated together to create what will be written to the device.

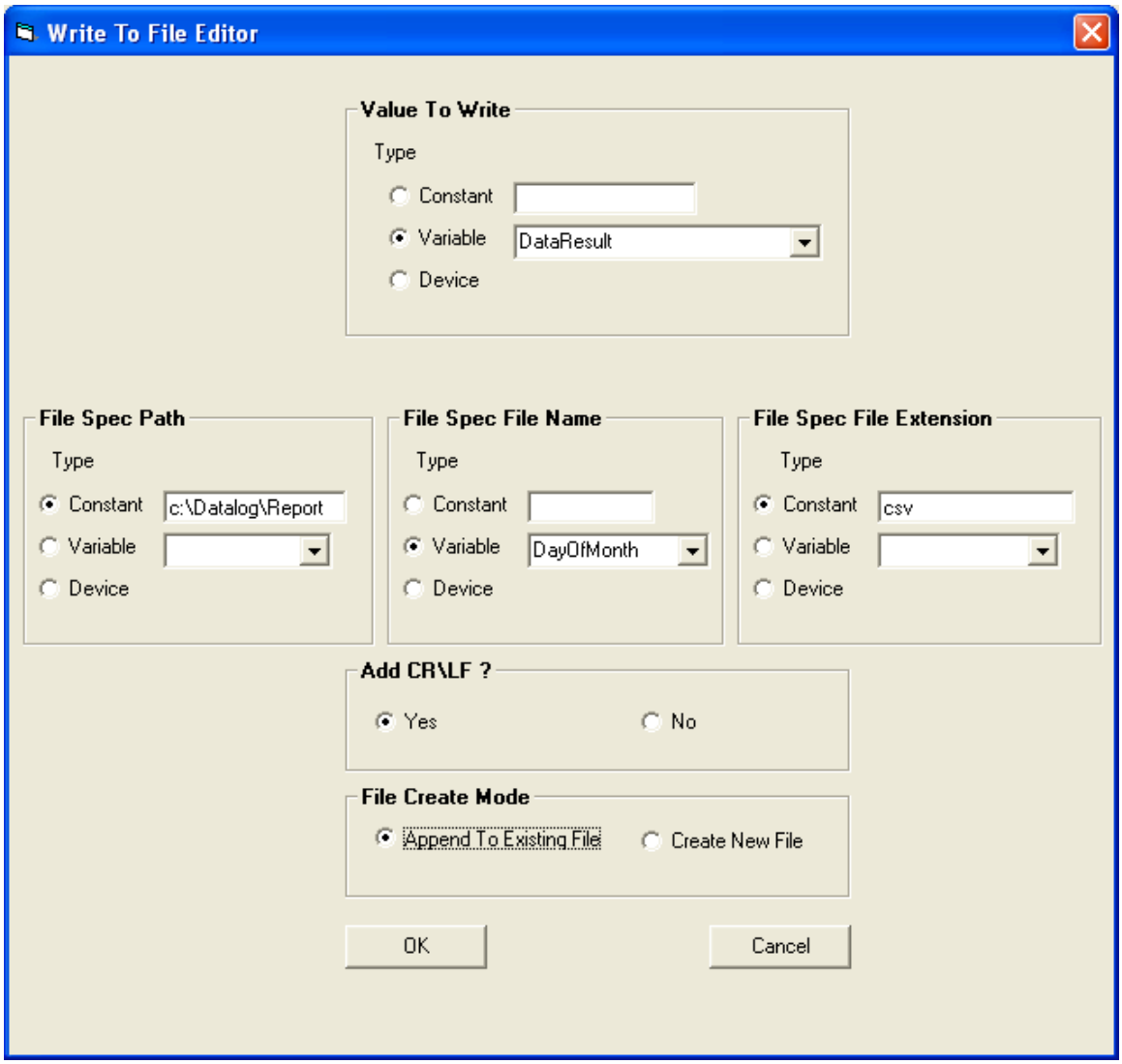


To specify precisely where to send the data you must enter the OPC server and the OPC Item address.

The OPC Server is selected from a list of previously configured servers. (See the section on OPC for Configuring OPC Servers) The OPC Item Address is the location within the OPC server to place the data.

WriteToFile Cell

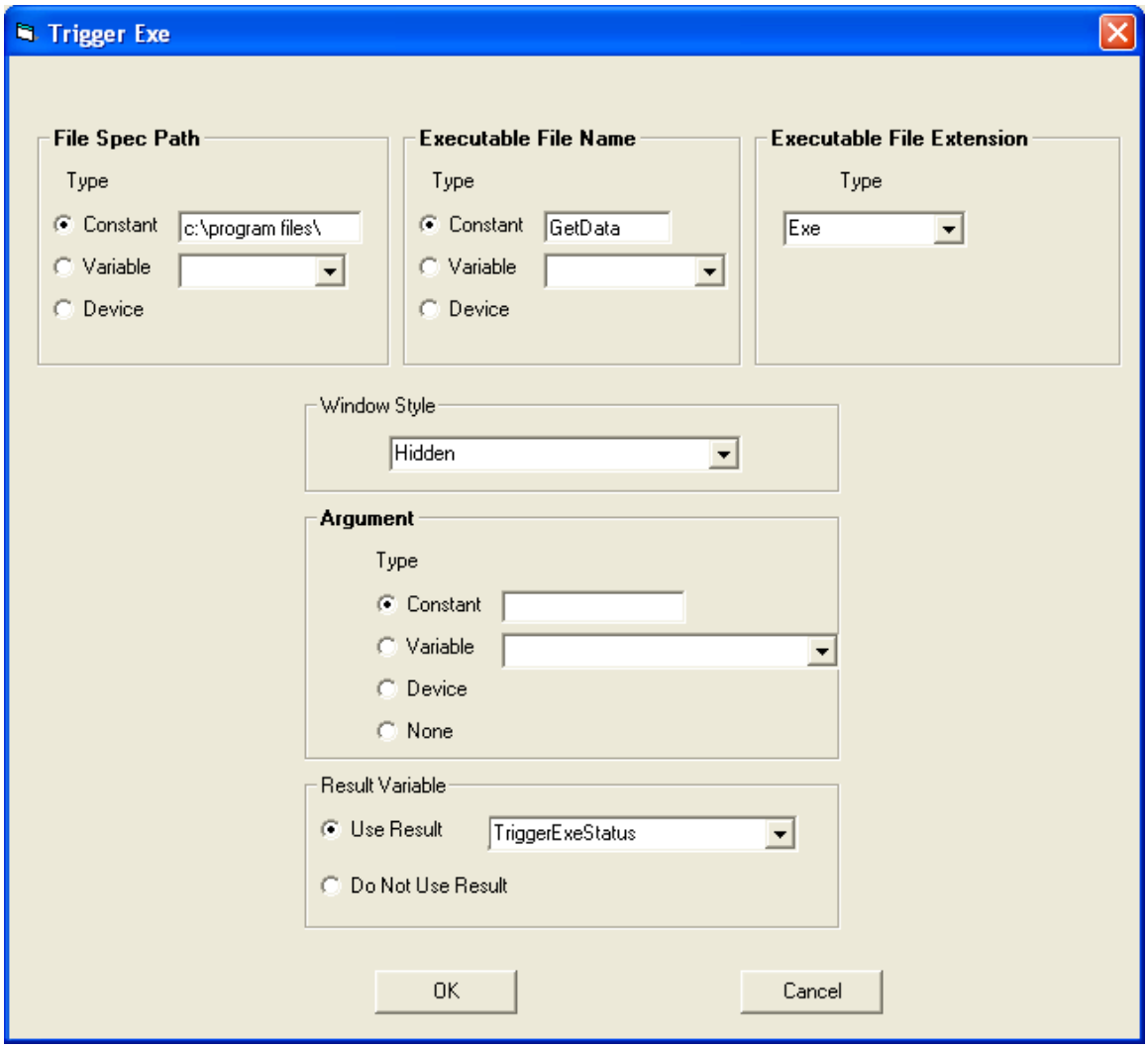
A *Write To File* cell is used to send data out of PW to a Text File. It can create a file or add to an existing file. There are 6 configurations, one for the value to be written, three to specify the path and file name, one for adding a carriage return and line feed to the data and one to determine if it will append to an existing file or create a new file each time.



Trigger Exe Cell

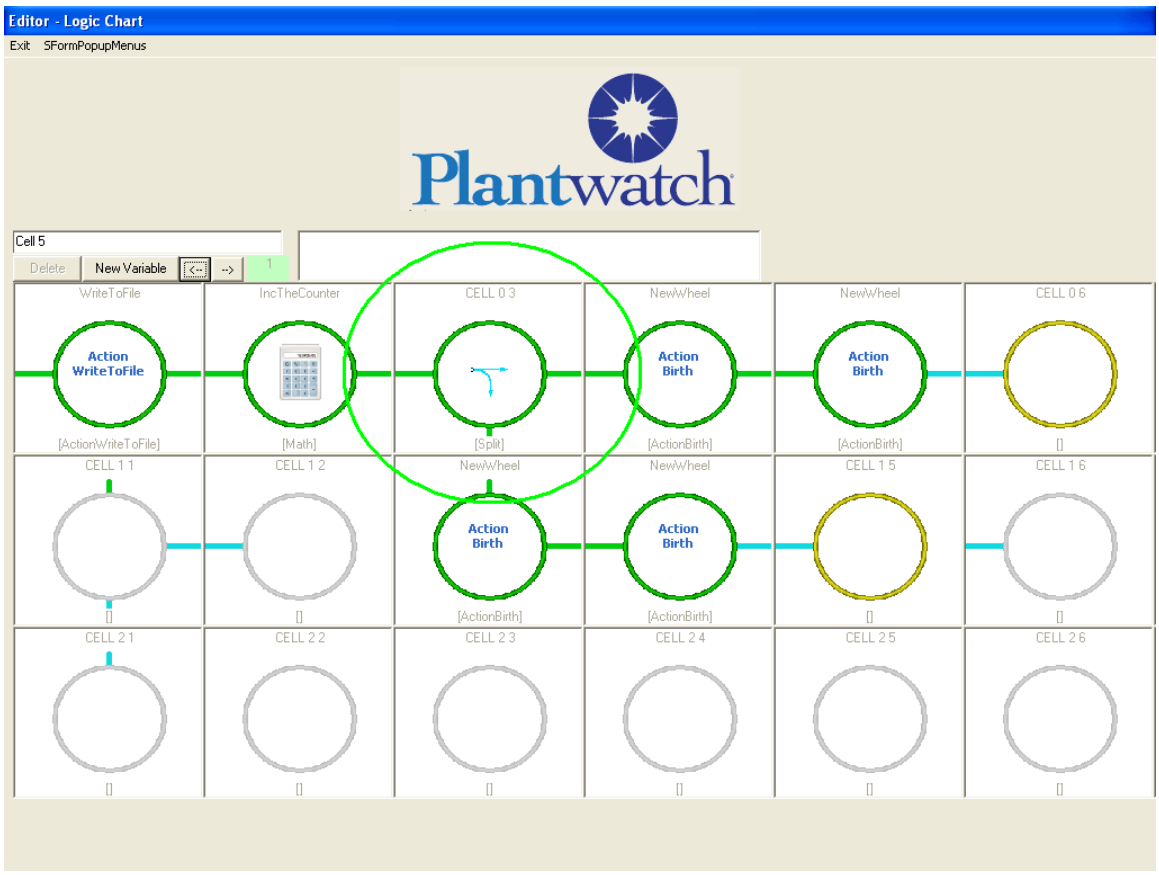
A *Trigger Exe* cell is used to cause a file to be executed. Any file of the supported file types on the available hard drives or network locations can be executed.

There are 6 Configurations, three to completely specify the file to execute, one to determine how the executed file will look, one to pass arguments to the executed file and one to get back status on the result of executing the file.



Split Cell

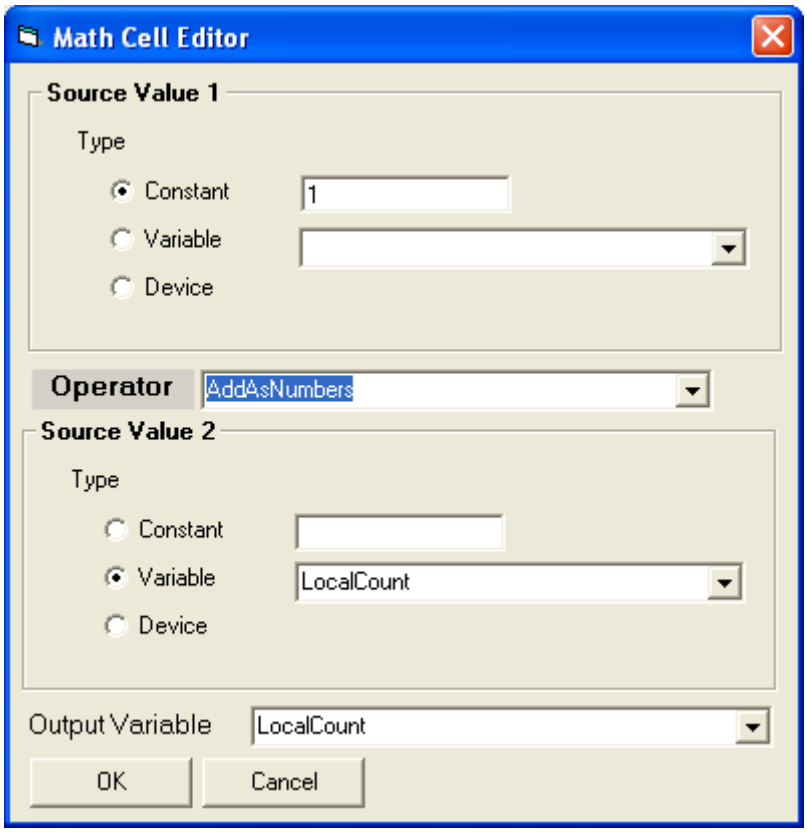
A *Split Cell* is used to control logic flow within the logic chart. It will create a new path for logic chart flow by enabling the cell directly below it's own location in the logic chart.



Math Cell

A *Math Cell* is used to do calculations. There are two source values and one result value. You can perform many different operations against the source values and get the results. Operators include addition, subtraction, multiplication, string manipulation and many more.

There are 4 configurations, two for the source values, one to choose the operator and one to choose the variable the result will be put into.



Supported operators:

SetToValue – Sets Output Variable equal to Source Value1
AddAsNumbers – Sets Output Variable equal to Source Value 1 plus Source Value 2
AddStringToEnd – Sets Output Variable equal to Output Variable and Source Value 1 as strings
Add2StringsToEnd – Sets Output Variable equal to Output Variable and Source Value 1 and Source Value 2 as strings
AddStringToBegining – Sets Output Variable equal to Source 1 and Output Variable
StringLength – Sets Output Variable equal to the number of characters in

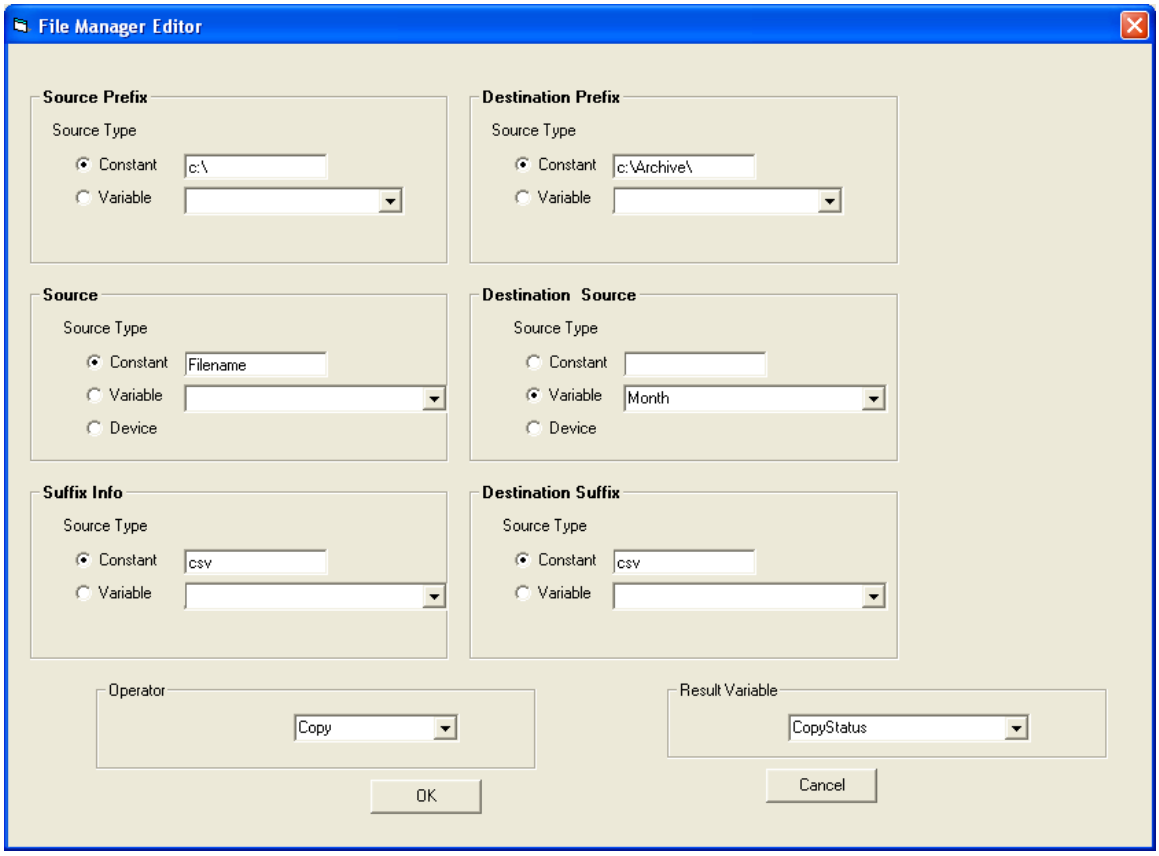
Source Value 1.
SubtractFromVariableAsNumber – Sets Output Variable equal to Source1 – Source 2
MutiplyAsNumber – Sets Output Variable equal to Source1 times Source 2
DevideAsNumber – Sets Output Variable equal to Source1 divided by Source 2
IncrementAsNumber – Sets Output Variable equal to Output Variable + 1
DecrementAsNumber – Sets Output Variable equal to Output Variable - 1

File Manager

A *File Manager cell* is used to do file operations on the hard drives and other storage media. These operations include copy, rename, delete, move, make a directory, check if a file exists and more.

File Manager works just like the windows command prompt. Depending on the command you may need arguments such as the source and destination paths needed for a copy command.

Command	Uses Source	Uses Destination
Copy	Yes	Yes
Rename	Yes	Yes
DeleteDir	Yes	No
Delete	Yes	No
Move	Yes	No
CheckForExistance	Yes	No
CreateDirectory	Yes	No



There are 8 fields, three for the source values, three for the destination values, one to choose the operator and one to choose the variable the status result will be put into.

If the command succeeds the *Result Variable* will be set to Pass.

If the command fails the *Result Variable* will be set to Fail.

DBBrowser Cell

Overview
The Database Browser cell allows the application to interact with a SQL database(s). New records can be put into the database(s), existing records can be retrieved from the database(s) or existing records can be changed within the database(s).

Before a Database Browser logic cell can be configured, you must configure the Databases branch of the tree. There you will enter information to allow Plantwatch to connect to the database(s). This will include the type of connection you choose to use as well as user names, passwords etc.

Database Browser has two layers of configuration.

On the first layer you setup the type of command you wish to use, such as *Select.* The name of the table you wish to interact with. If you want to interact with more that one record at a time by using array variables. And if you wish to associate a variable to receive the pass fail result of the attempted action.

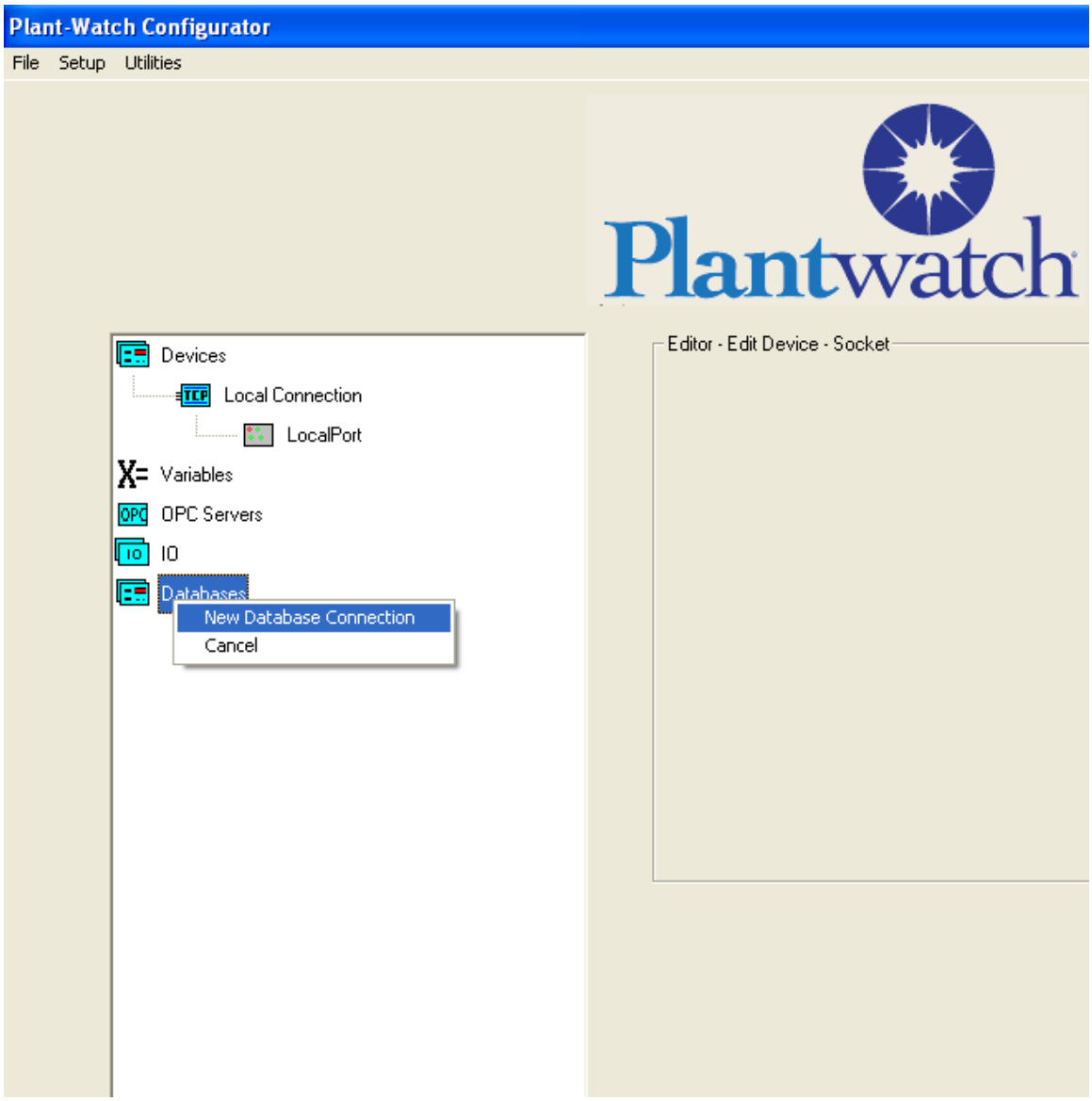
On the second layer you setup which columns are being used in the table and associate variables to each column. You also create any where-clause logic on the second layer.

Database Browser supports Array Type variables as well as non array variables. If Array variables are used, multiply records can be retrieved for each event, one for each element of the Array variable. You configure if you are using Arrays on the first layer of configuration.

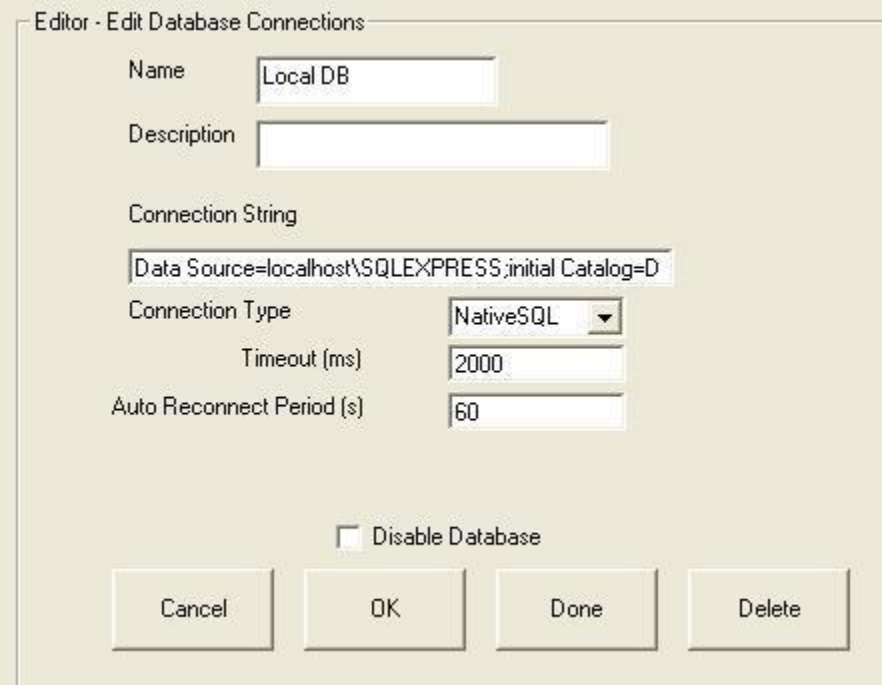
Configuration of Databases
To use the Database Browser Logic Chart cell type you must first configure the *Databases* branch of the configuration tree.

This will only be done one time for each SQL database you wish to engage.

To add a Database Connection, right click on *Databases* and select *New Database Connection*.



You will be presented with a dialog for creating the Database Connection.



Enter a name and description. Any name will work. This name will be used when configuring a Database Browser cell within a *Logic Chart.*

Enter a valid SQL connection string as needed for your database. The default connection string will point to a local SQL Express database with a database called PW in it.

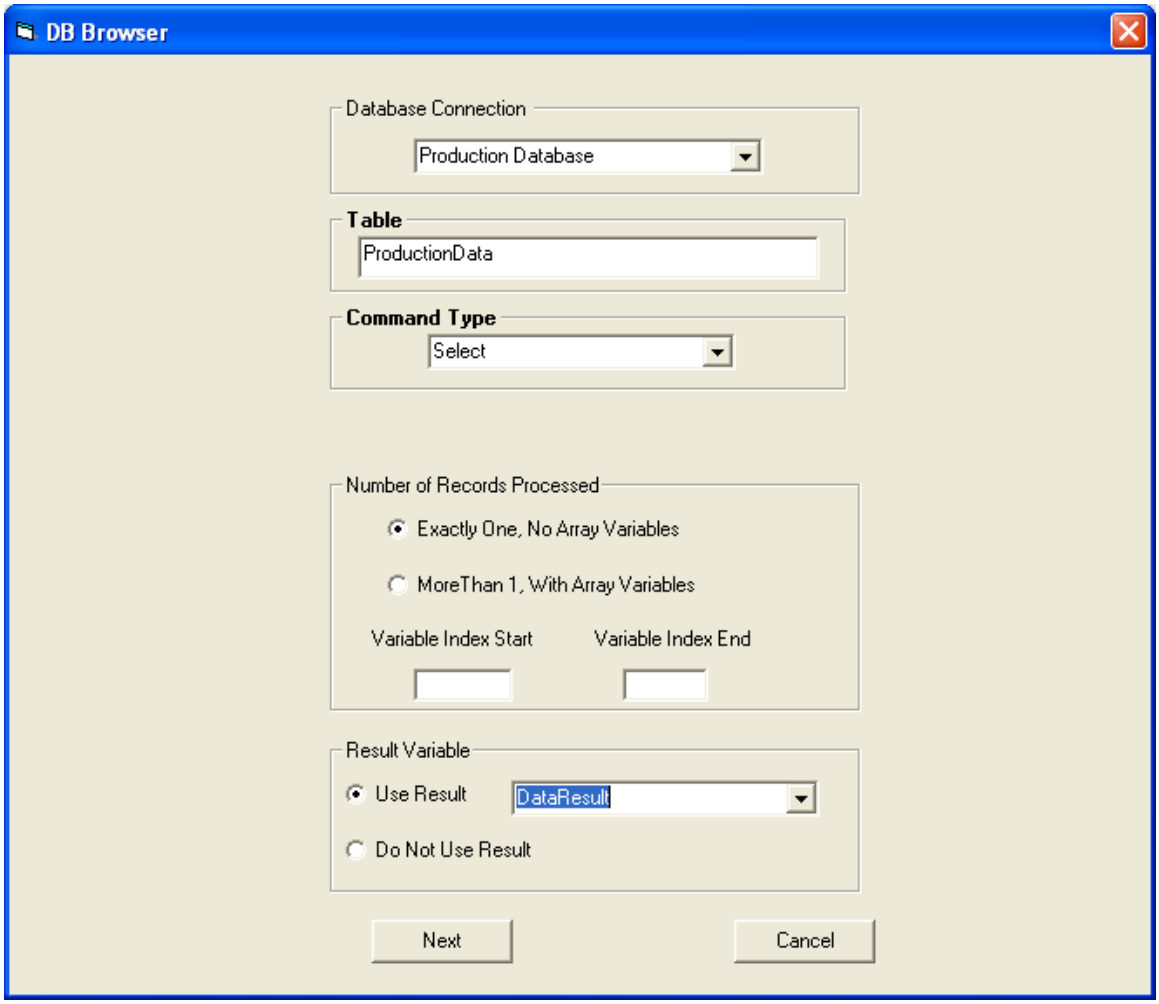
Choose between NativeSQL and ODBC. Native SQL is used when communicating to Microsoft SQL databases. ODBC is used to communicate to other databases such as Oracle, Sysbase or Ingress. (Note that ODBC can also be used to communicate to Microsoft SQL databases)

The default Timeout and reconnect period should not be changed

Save the work and exit.

Configuration of First Layer

The first layer of configuration is where you choose the Database Connection, choose the table name, the function to execute such as *Select*, if you are going to use Arrays and if you want to use the Result Variable.



Database Connection:
Select the database you wish to interact with the Database Connection dropdown. (There will be one entry for every database you added to the *Databases* branch of the configuration tree.)

Table:
Enter the name of the table you wish to interact with.

Command Type:
Select the command type, such as Select that you want from the list in the Comamnd Type dropdown. Commands include Insert, Select, Update and Delete.

Number Of Records Processed:
Set the Number of Records Processed configuration based on if you want to interact with more than one record. If you do want to interact with more than one record select *More Than 1, With Array Variables.*

If you do select *More Than 1, With Array Variables* the variables you use will have to be Array variables and you will need to enter the Variable Index Start and End fields.

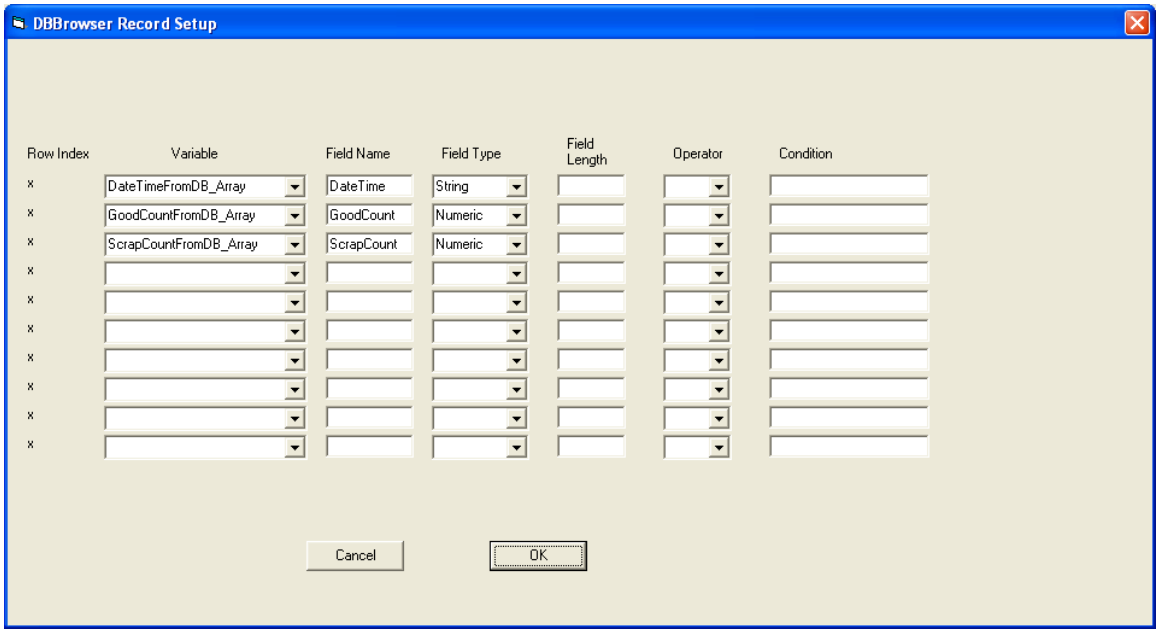
As an example, if the Select returns 5 records it will place the data from record 1 into the Array variable position of what is configured in the Variable Index Start. For each additional record it will fill the next position within the Array variables until it reaches the position of what is configured in the Variable Index End.

Result Variable:
If you wish to get feedback on the success or failure of the executed command, Set the *Use Result* button and select a variable from the dropdown. After the command is completed the pass fail result will be in the variable selected.

Configuration of Second Layer

The second layer of configuration allows you to

choose what columns of the table you wish to access
associate variables to the columns
Build up a where clause to limit the records accessed



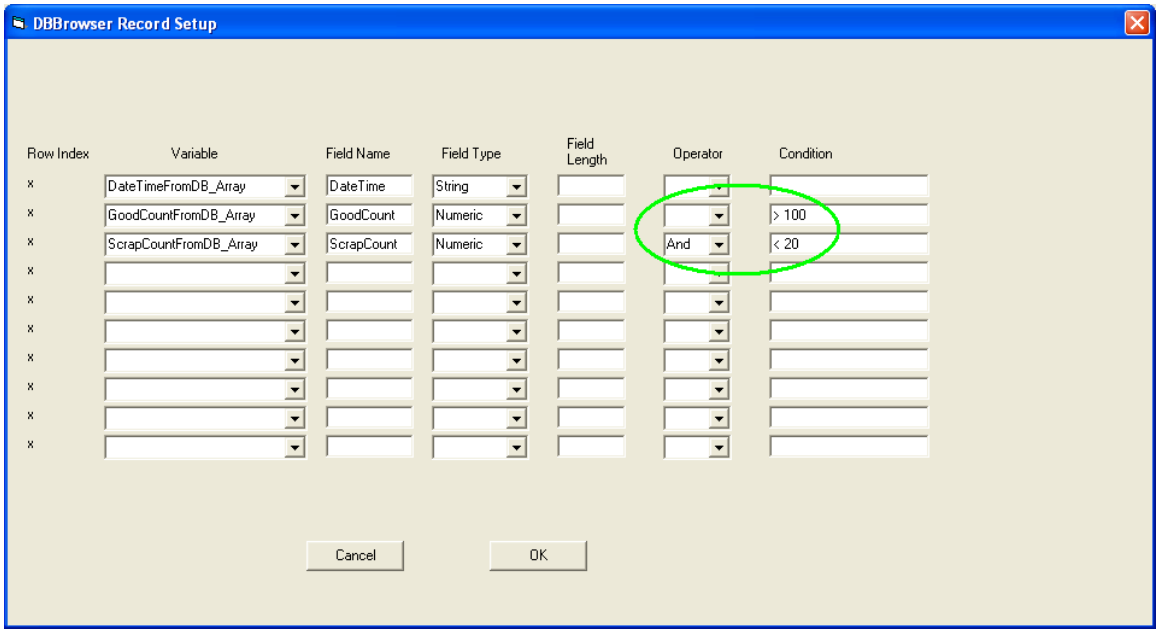
Variable:
Will be populated with data in a Select action.
Will be the source of information for an insert.

Field Name:
The name of the field in the table that will be populated with data in an Insert action.
The name of the field in the table that will supply the data in a Select action.
The name of the field in the table that will compared to if the Condition is used.

Field Type:
Specify how the data is to be interpreted. Either Numeric or String. If set to numeric all values are managed as numbers. If set to String the vales are managed as a string and therefore encapsulated with single quotes as appropriate.

Field Length:
Reserved for future functionality. Not currently used.

Operator:
Allows the construction of logic in the where clause. Entries include And, Not and Or. This allows you to construct queries like select from where GoodCount > 100 *And* ScrapCount < 20. This would be accomplished as shown below.



Condition: Allows you to choose what records within the database are affected. The supported operators are the same set of operators supported by the database. This would include operators such as greater than, less than equal to etc.

An example of a simple condition with a numeric field type would be:
= 23
The resultant SQL command would contain = 23 within it.

You can also use a *Variable* within the *Condition*. To use a *Variable* within the *Condition* place it in the condition bracketed by percent signs.

An example of using a *Variable* called SelectionCriteria with a numeric field type in a *Condition* would be.

= %SelectionCriteria%

If variable SelectionCriteria contained the value 92 the resultant SQL command would contain = 92 within it.

Example - Simple Insert

Lets say that we want to insert one record at a time into a table called ProductionData, which has columns of:

DateTime
GoodCount
ScapCount

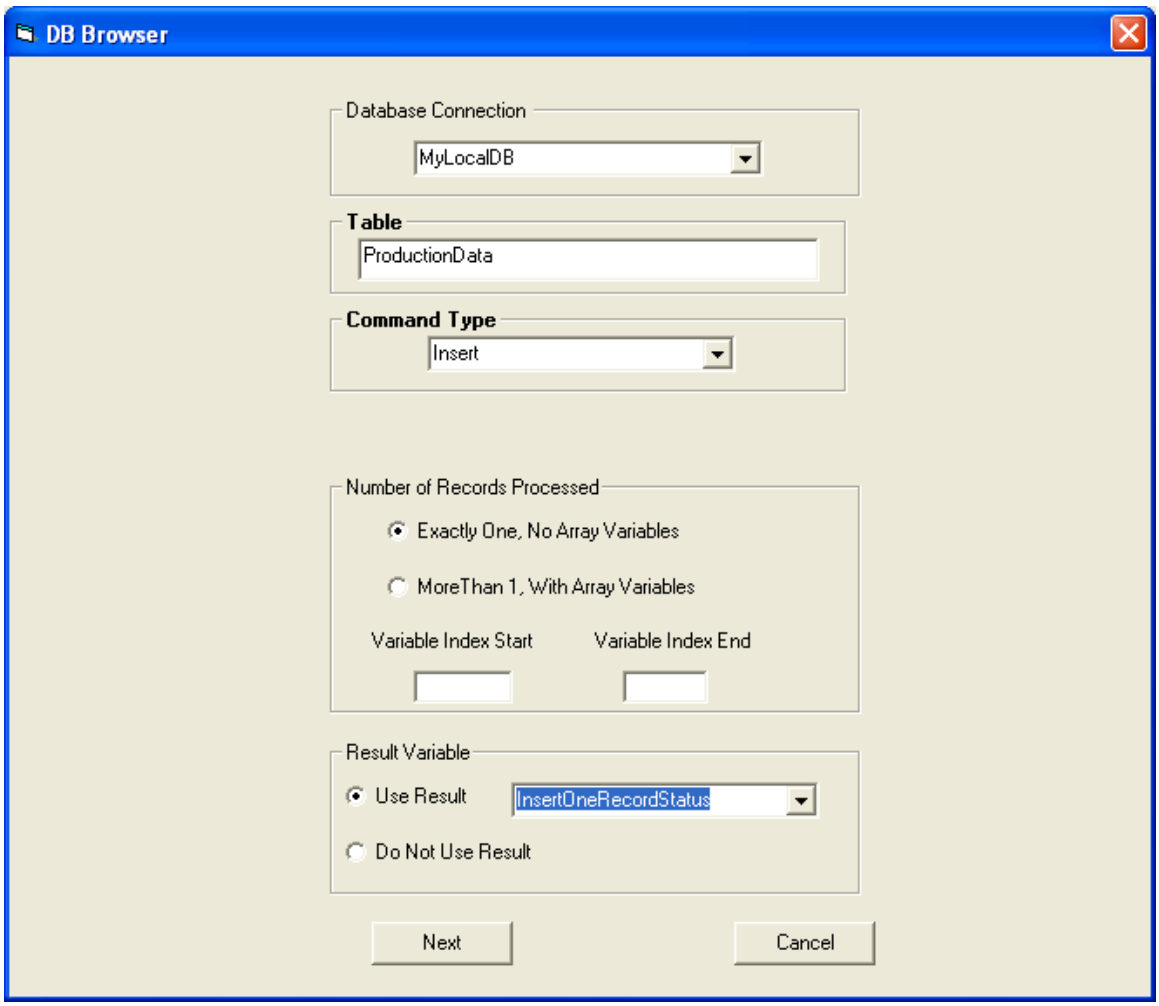
We will get the values from three variables to insert into tables columns:

DateTime
GoodCountToDB
ScrapCountToDB

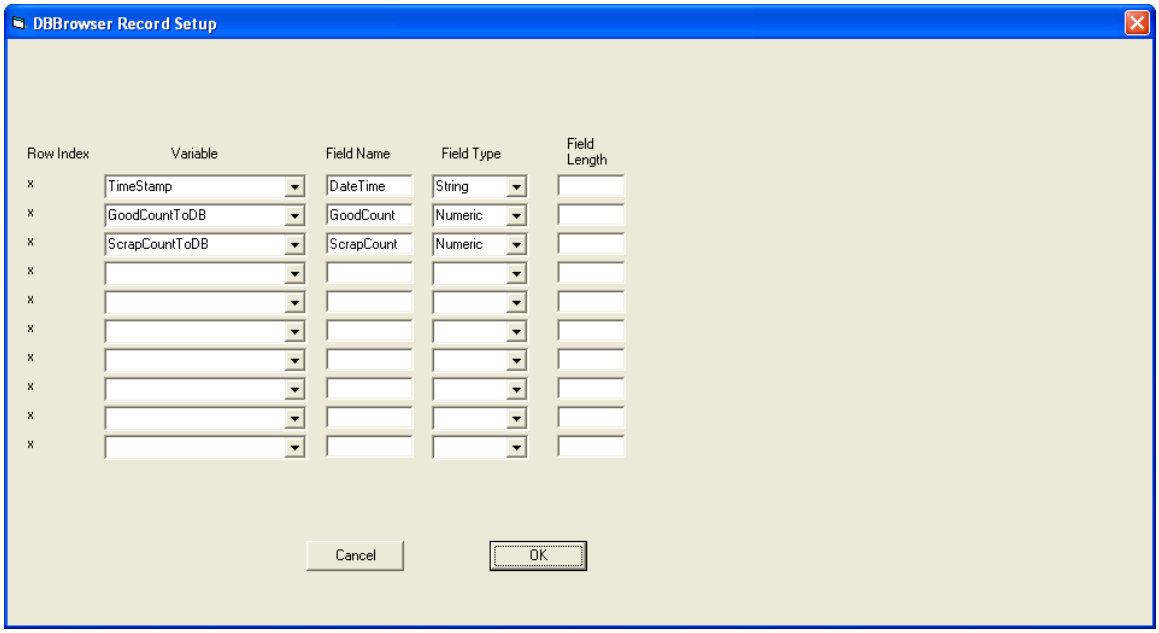
This converted into SQL would be something like:

'Insert into ProductionData (DateTime, GoodCount, ScrapCount) values (xxx,xxx,xxx)'
with the values xxx being the values from the three variables

First layer configuration would need to be set to the correct Database Connection, table, command set to Insert and Records to process set to 1.



Second Layer configuration would need to select the three columns in the table and associate one of the three variables to each column for it's data source. You also must specify if each column is numeric or string data.



Configuration for a simple insert

To set up for the simple insert you select the Variable names you wish to get data from and on the same line you enter a column in the table for the data to be placed into. As configured:

The data from variable TimeStamp will be placed into column DateTime.

The data from variable GoodCount will be placed into column GoodCount.

The data from variable ScrapCount will be placed into column ScrapCount.

Example - Simple select

Lets say that we want to get the first record from a table called ProductionData, which has columns of:

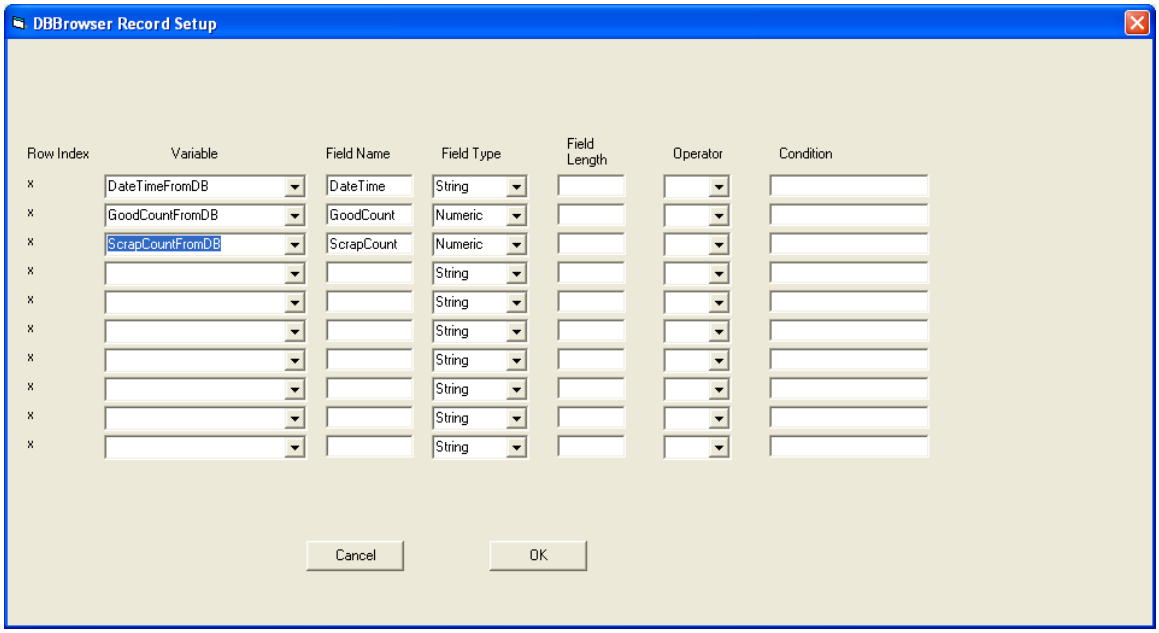
DateTime
GoodCount
ScapCount

This converted into SQL would be something like:

'Select DateTime, GoodCount, ScrapCount from ProductionData'

We will put the values from the three columns into variables we have on a screen:

DateTimeFromDB
GoodCountFromDB
ScrapCountFromDB



Configuration for a simple select

To set up for the simple select you enter the field names you wish to get data from and on the same line you select a variable for the data to be placed into. As configured:

The data from field DateTime will be placed into variable DateTimeFromDB.
The data from field GoodCount will be placed into variable GoodCountFromDB.
The data from field ScrapCount will be placed into variable ScrapCountFromDB.

Example – Select records where ScrapCount is greater than 30

Lets say that we want to get the records from a table called ProductionData that had a ScrapCount value of more than 30 . The table called ProductionData has columns of:

DateTime
GoodCount
ScapCount

This converted into SQL would be something like:

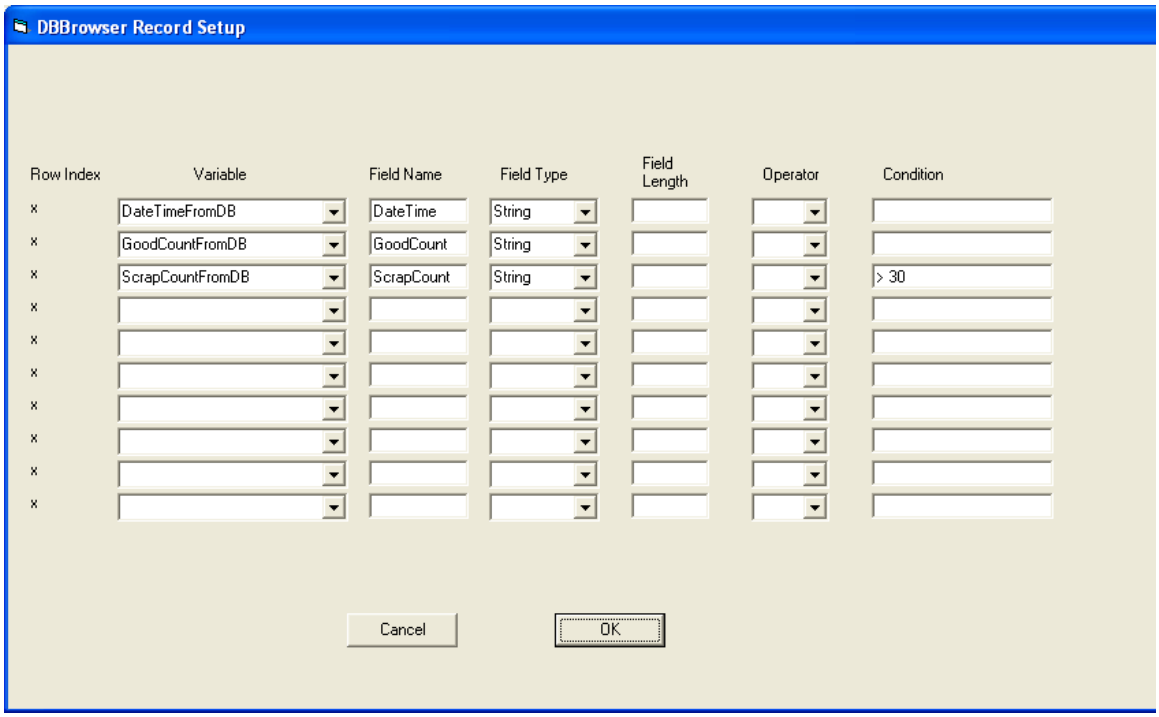
'Select DateTime, GoodCount, ScrapCount from ProductionData
Where ScrapCount > 30'

We will put the values from the three columns into variables we have on a screen:

DateTimeFromDB
GoodCountFromDB
ScrapCountFromDB

If there are any records in the table ProductionData that have a ScrapCount value greater than 30, the first of these records will be retrieved.

If there are no records in the table ProductionData that have a ScrapCount value greater than 30, no records will be retrieved.



Configuration for a select with a where clause

To set up for the simple select you enter the field names you wish to get data from and on the same line you select a variable for the data to be placed into. You then configure your where clause by entering "> 30" in the condition field. As configured:

The data from field DateTime will be placed into variable DateTimeFromDB.
The data from field GoodCount will be placed into variable GoodCountFromDB.
The data from field ScrapCount will be placed into variable ScrapCountFromDB.

<u>Example – Select records where ScrapCount is greater than variable DBBrowserSelect</u>
Lets say that we want to get the records from a table called ProductionData that had a ScrapCount value of more than variable DBBrowserSelect. The table called ProductionData has columns of:

DateTime
GoodCount
ScapCount

This converted into SQL would be something like:

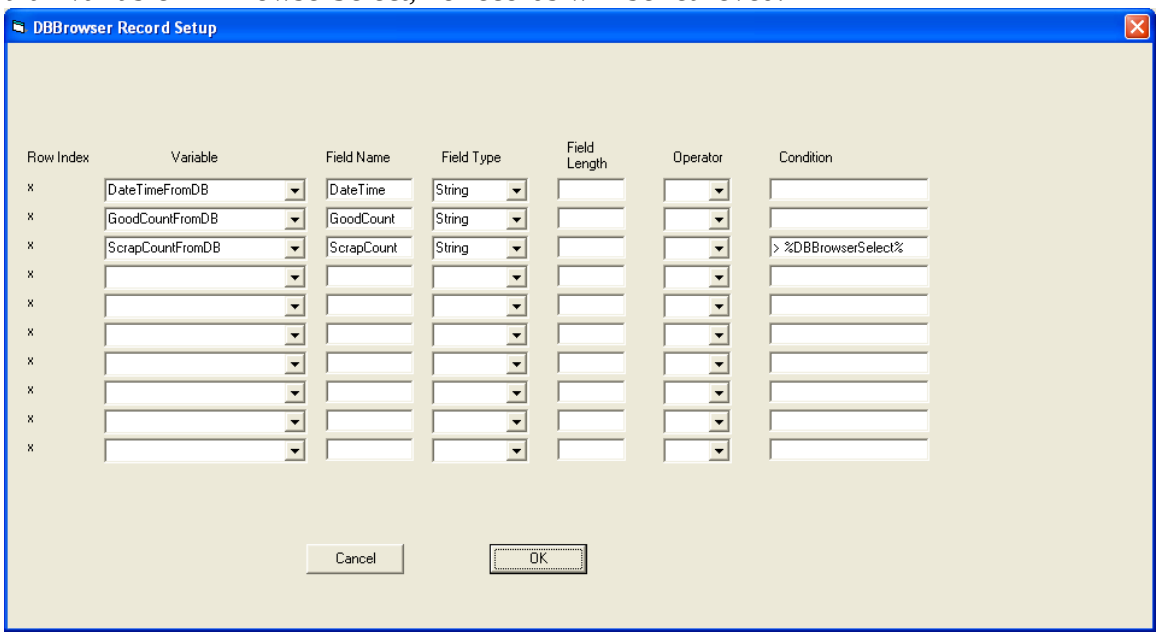
'Select DateTime, GoodCount, ScrapCount from ProductionData
Where ScrapCount > variable DBBrowserSelect'

We will put the values from the three columns into variables we have on a screen:

DateTimeFromDB
GoodCountFromDB
ScrapCountFromDB

If there are any records in the table ProductionData that have a ScrapCount value greater than variable DBBrowserSelect, the first of these records will be retrieved.

If there are no records in the table ProductionData that have a ScrapCount value greater than variable DBBrowserSelect, no records will be retrieved.



This configuration has a Condition field that has a variable referenced in it. When DBBrowser finds a variable encased in percent times it will replace it with the value of the variable.

<u>Example – Select the last record inserted</u>
Lets say that we want to get the last record from a table called ProductionData, which has columns of:

DateTime
GoodCount
ScapCount

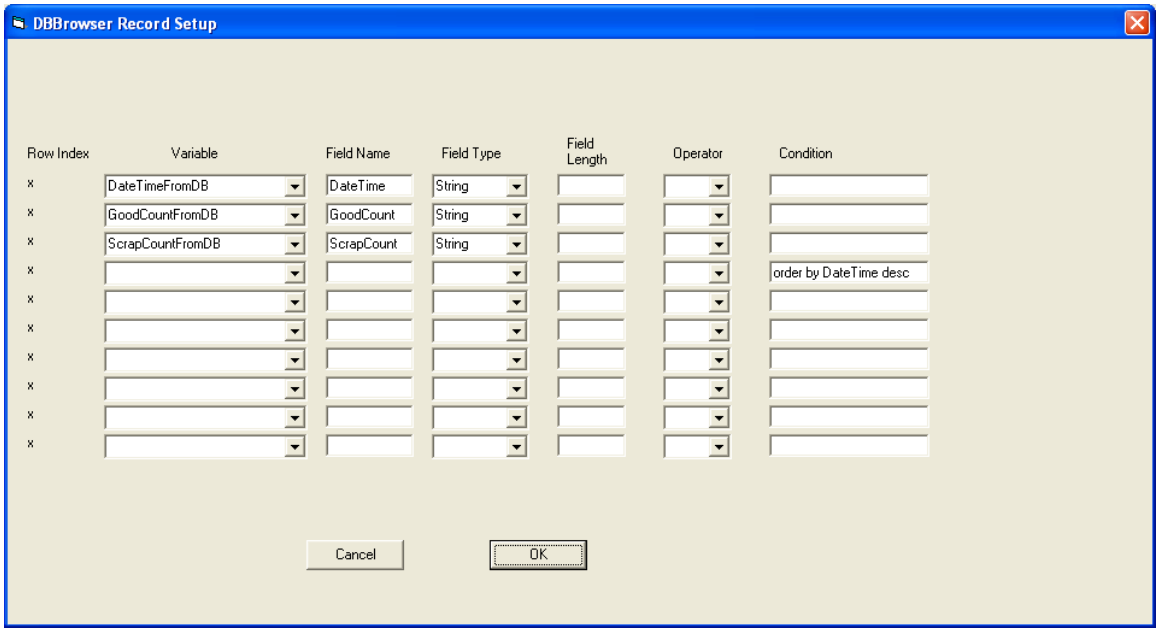
We know that a SQL database will, unless instructed otherwise, return the first record in the table. To change what record is returned in SQL we could tell it to re-order the records based on a specific column. This command is Order By and expects a column name to follow it. It can also be directed to order in ascending or descending order. Since we want the last record we will choose descending order so our command is ‘order by datetime desc.

This converted into SQL would be something like:

'Select DateTime, GoodCount, ScrapCount from ProductionData order by DateTime Desc'

We will put the values from the three columns into variables we have on a screen in the order of the DateTime field in a descending fashion:

DateTimeFromDB
GoodCountFromDB
ScrapCountFromDB



This configuration has a Condition field populated in a row that has no field or variable selected. In this case DBBrowser will simply attach the string onto the end of the Select. This allows significant flexibility in what DBBrowser can do

.

Example - Array Insert

Lets say that we want to insert 5 records at a time into a table called ProductionData, which has columns of:

DateTime
GoodCount
ScapCount

We will get the values from three Array variables to insert into tables columns:

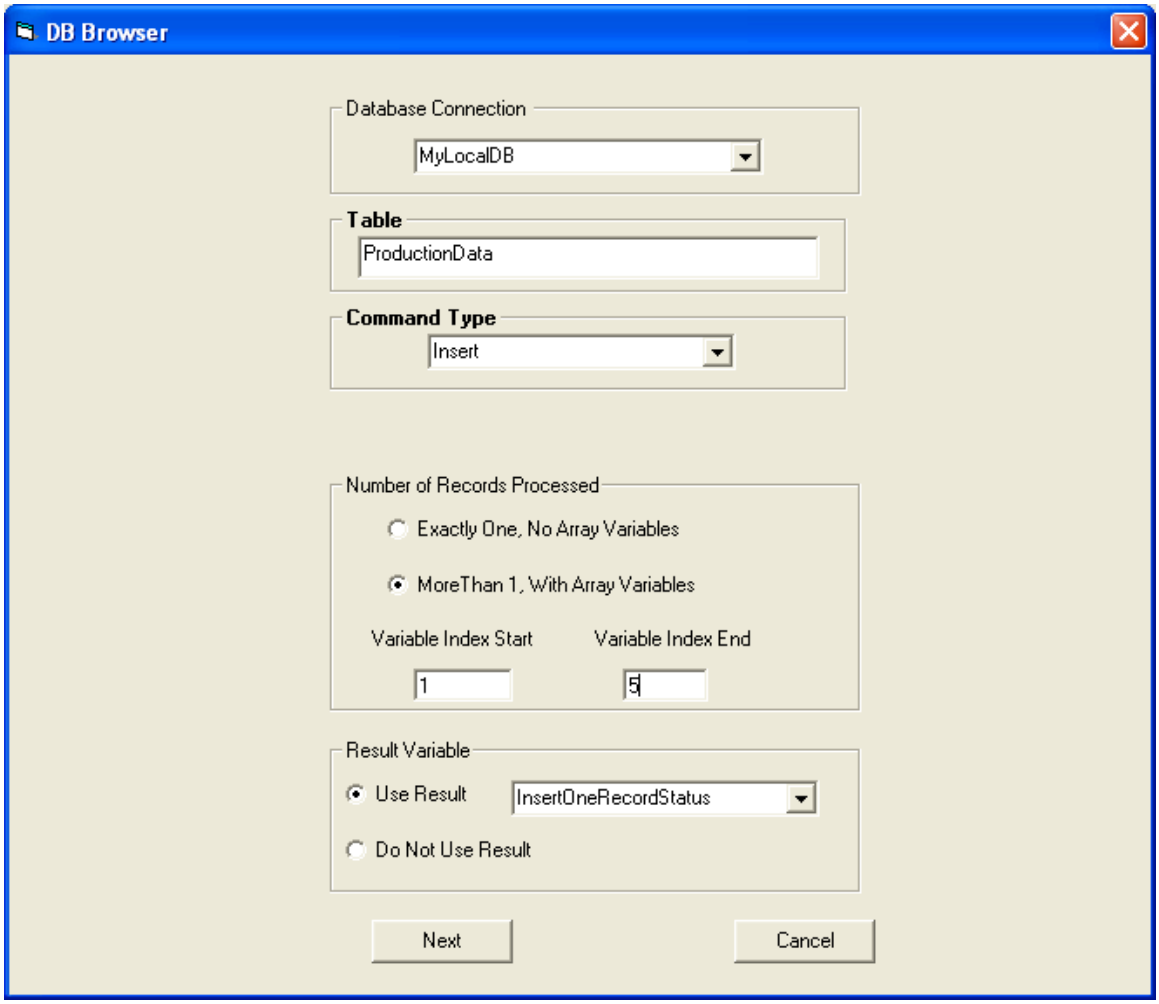
DateTime
GoodCountToDB
ScrapCountToDB

This converted into SQL would be something like the following executed five times:

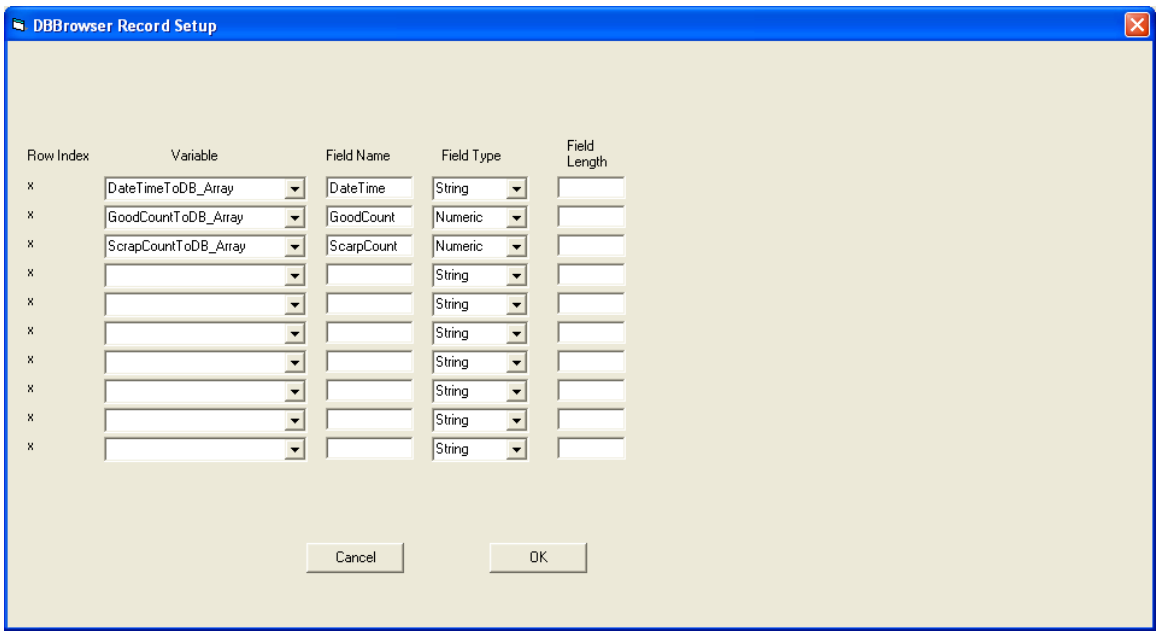
'Insert into ProductionData (DateTime, GoodCount, ScrapCount) values (xxx,xxx,xxx)'

with the values xxx being the values from the three variables

First layer configuration would need to be set to the correct Database Connection, table, command set to Insert and Number of Records to Process set to more than 1. Also set Variable Index Start to 1 and Variable Index End to 5.



This configuration is setup for Array Variables 5 long



The second level configuration with Arrays is the same as not using Arrays. The difference now is that 5 inserts will occur.

The first record will get data from variables generate by the existence of an Array:
DateTimeToDB_Array_1
GoodCountToDB_Array_1
ScrapCountToDB_Array_1

The second record will get data from variables generate by the existence of an Array:
DateTimeToDB_Array_2
GoodCountToDB_Array_2
ScrapCountToDB_Array_2

The third record will get data from variables generate by the existence of an Array:
DateTimeToDB_Array_3
GoodCountToDB_Array_3
ScrapCountToDB_Array_3

And similar for inserts four and five.

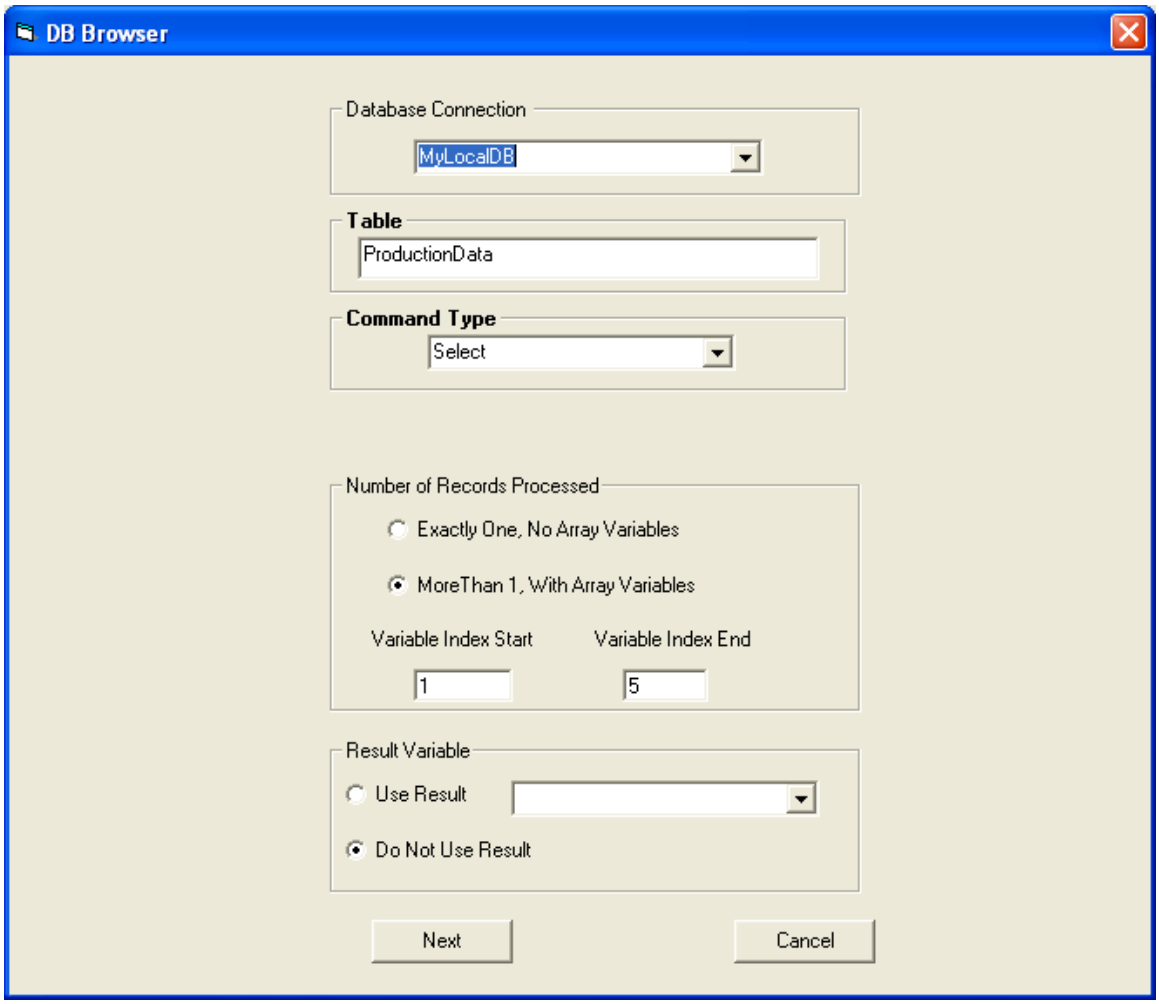
Example - Array Select

Lets say that we want to select up to five records at a time from a table called ProductionData, which has columns of:

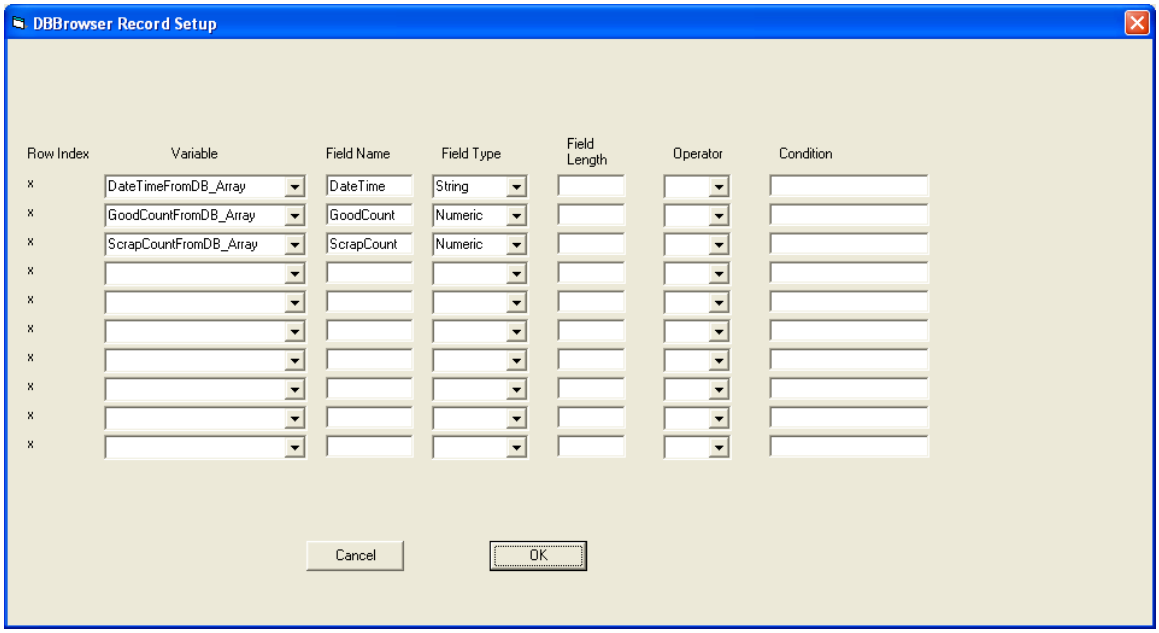
DateTime
GoodCount
ScapCount

We will get the values from three columns and place it into variables for up to five records.

First layer configuration would need to be set to the correct Database Connection, table, *Command* set to *Select* and *Number of Records to Process* set to more than 1. Also set *Variable Index Start* to *1* and *Variable Index End* to *5*.



Second layer configuration would associate Array type variables to the columns



When this executes it would return up to 5 rows of data.

The first row of data would go into variables:

DateTimeFromDB_Array_1
GoodCountFromDB_Array_1
ScrapCountFromDB_Array_1

The second row of data would go into variables:

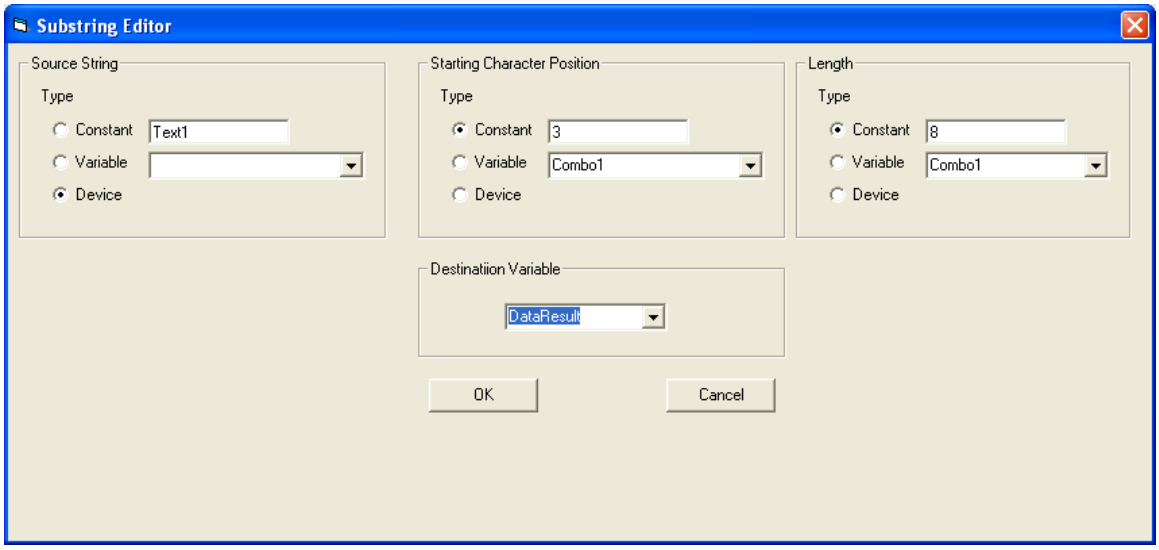
DateTimeFromDB_Array_2
GoodCountFromDB_Array_2
ScrapCountFromDB_Array_2

The third row of data would go into variables:

DateTimeFromDB_Array_3
GoodCountFromDB_Array_3
ScrapCountFromDB_Array_3

Substring Cell

A *Substring Cell* will extract a portion of a string based on a starting point and length. It places the extracted data as a string into the Destination Variable.



There are four fields. One for the Source data, one for the starting position, one for the length and one for the status variable.

When executed the Substring cell will get a part of the *Source String* starting at the *Starting Character Position* and ending based on the *Length* and put that part into the *Destination Variable.* .

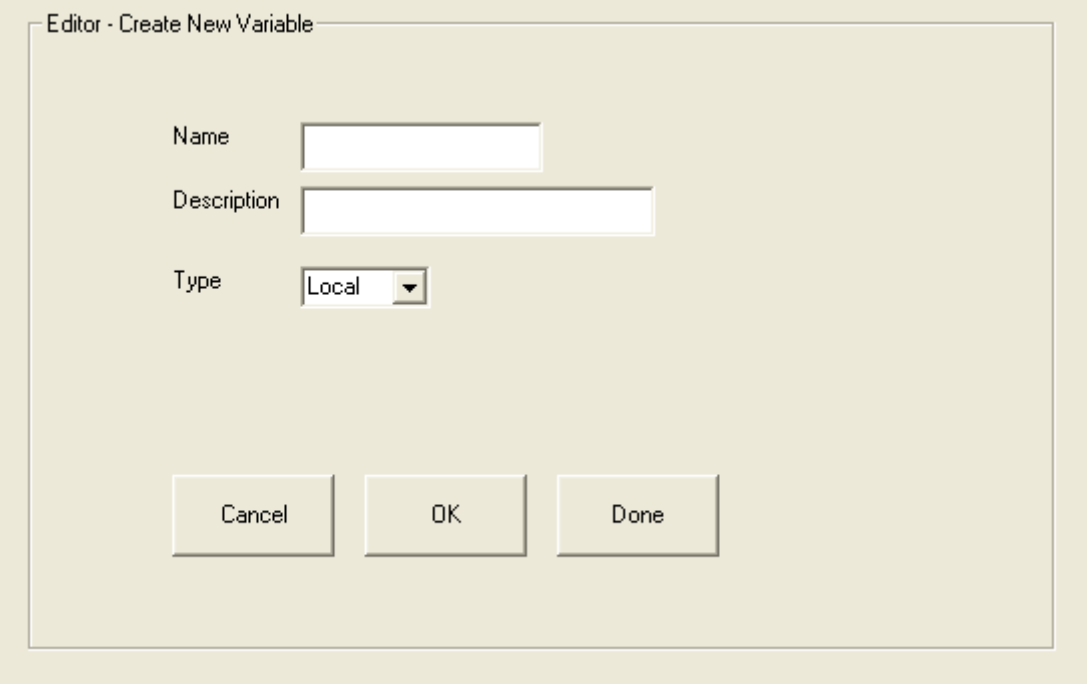
Variables
Within PlantWatch there can be a lot of data. To manage this data there are *Variables*. They can be used to store data, bring data in from OPC and to bring data in from an IO card. Within logic charts *Variables* are used extensively to manage how the cells act. For example the *WriteToFile* can be configured to write the value of a *Variable* to the specified file.

There are three types of *Variables, Local, OPC and Array*.

Local Type Variables

Local variables are used for data storage as needed by the application. They are created by right clicking within the *development Tree* on the *Variable* section, and then selecting New Variable. The default variable *Type* is *Local* so there is no need to edit the variable type drop down.

After selecting *New Variable* you will be presented the dialog to create a new *Variable*.



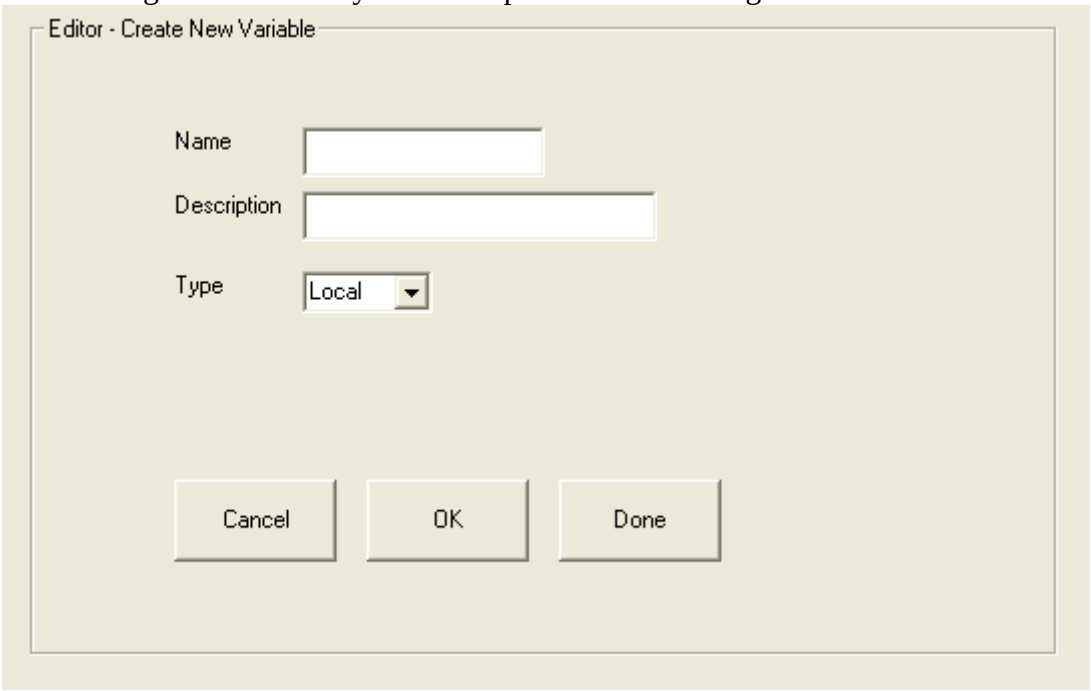
Enter a name and description, and then select the *OK* button. The system will generate the desired *Variable* and then erase the entered name and description. You can create another *Variable* or select the *Done* button to exit. All *Variables* must have unique names.

OPC Type Variables

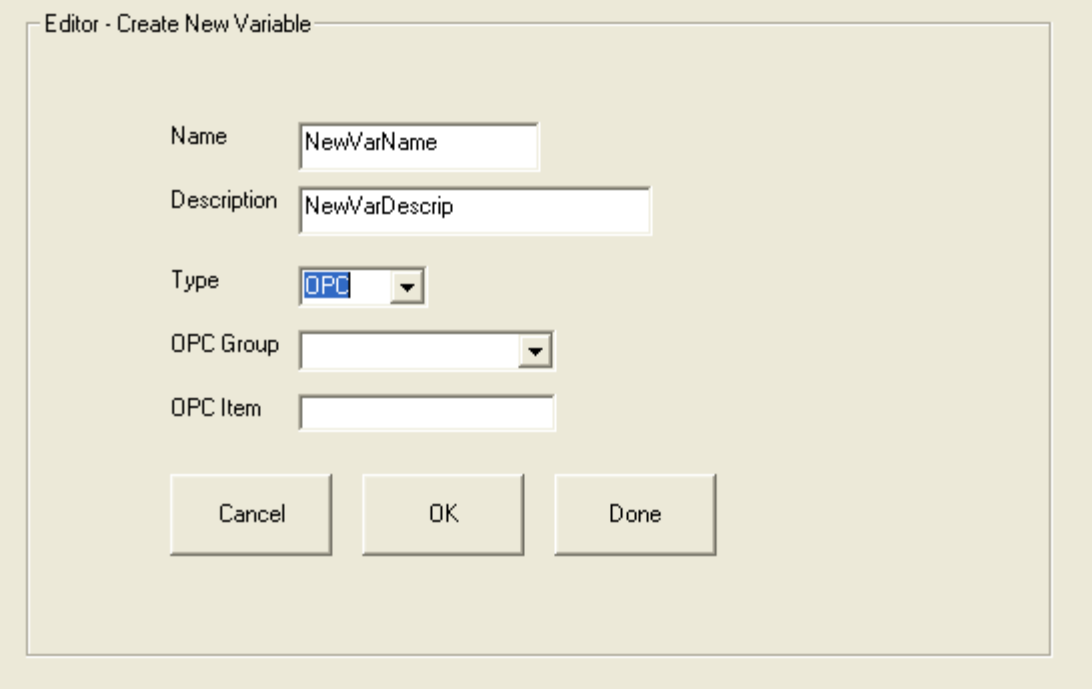
OPC type variables are used to bring in data thru OPC. An OPC type *Variable* is configured to be associated to a previously configured OPC server and Group. Additionally the *Variable* is configured to be associated with a specific item within the data available from the OPC server.

OPC Variables are created by right clicking on the *Development Tree*, on the *Variable* section, and selecting New Variable.

After selecting *New Variable* you will be presented the dialog to create a new *Variable*.

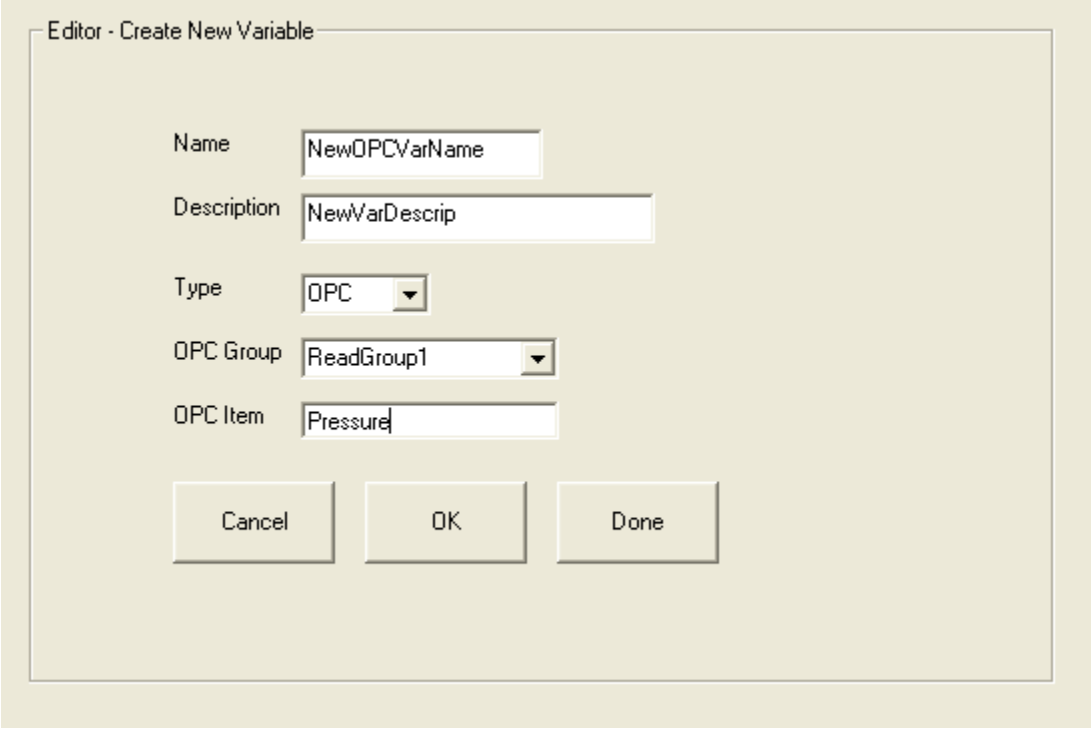


Enter a name and description, and then change the *Type* from *Local* to *OPC*. You will see two new fields appear for you to configure. OPC Group and OPC Item.



Select a previously configured OPC Group from the drop down list. (If there is not an OPCGroup to choose from the drop down list, you will have to create the OPC configuration as documented under OPC.)

Set the *OPC Item* to the address as suggested by the OPC server you are using.



When you are done entering data, select the *OK* button. The system will generate the desired *Variable* and then erase the entered name, description and OPC Item. You can create another *Variable* or select the *Done* button to exit. All *Variables* must have unique names.

(See the section of this document titled *Getting OPC Data into PW* for more information.)

Array Type Variables

Array type variables are variables that have room inside them to store many different values. When the Array type variable is created, it's maximum number of stored values is set. The maximum number of stored values is called the Array Length.

Each stored value within the Array type variable is addressable as if it was a local variable. For example with an Array type variable named *Data*, with an *Array Length* of three, you could use *Data_1* or *Data_2* or *Data_3* any place a PlantWatch variable is called for.

Note that the base name, for example *Data*, cannot be used in places that are not requesting an Array Type Variable. The base name has no relevance unless somehow coupled to a specific location in the Array Type Variable. In these places you will not see the base name in the list of available Variables. What you will see is all of the members of the Array Type Variable listed, *Data_1*, *Data_2* and *Data_3*.

The Array Type Variable Data with it's Array Length set to three will be able to store three unique values.

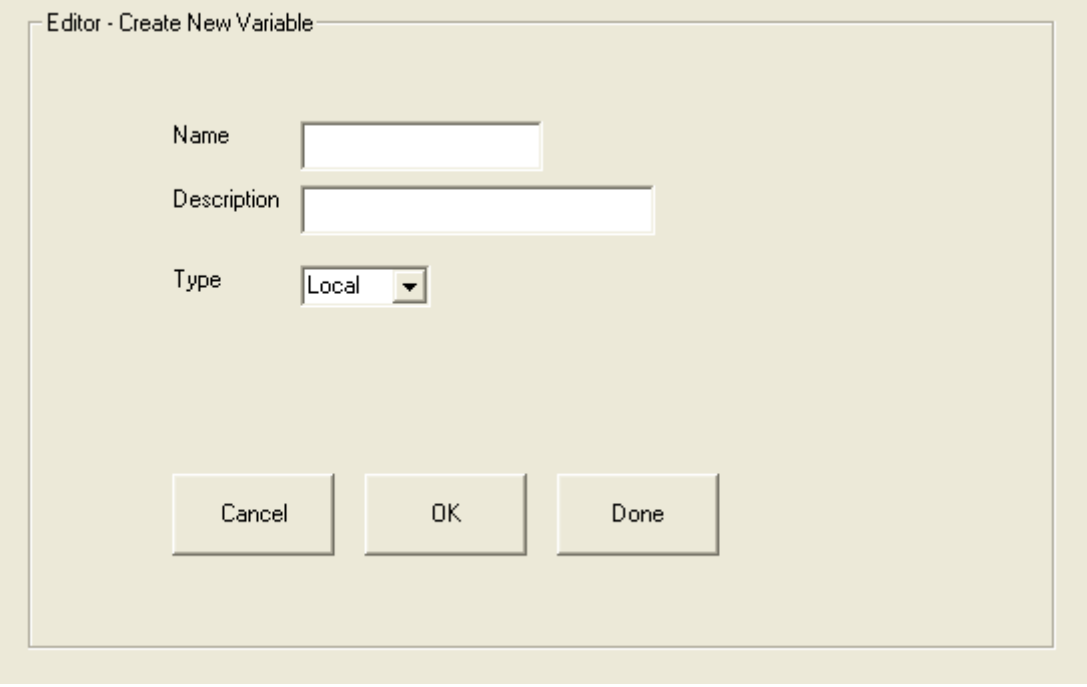
Note that there are specific places where an Array Type Variable can be used to manage data.

1 Database Browser:
In the Database Browser Logic Chart action you are able to retrieve data from a SQL database. This often results in more than one record returned. By Using Array Type variables you can accept many records from the Database Browser Logic Chart action with only referring to one variable. You can use the Database Browser Logic Chart action without using Array Type Variables but you would only be able to retrieve one record.

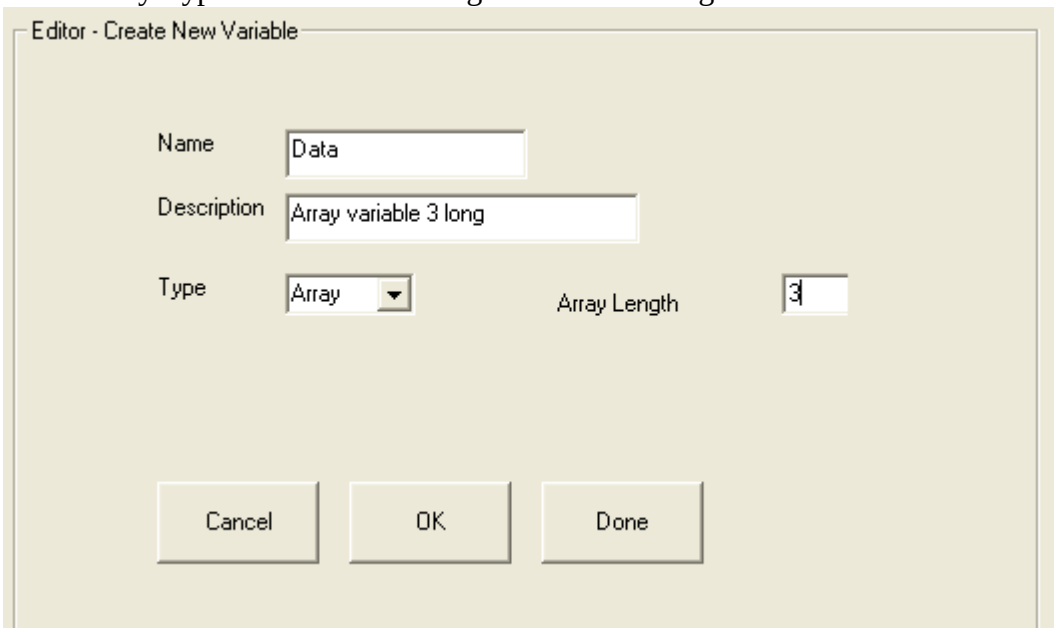
2 Graphics Editor:
When using the Variable List animation you must refer to an Array Type variable. This animation type used in conjunction with Array Type Variables allows the user to select from an unknown number of options. This could be used to select one of the records retrieved from a SQL Database.

Array Type Variables are created by right clicking within the *development Tree* on the *Variable* section, and selecting *New Variable*.

After selecting *New Variable* you will be presented the dialog to create a new *Variable*.

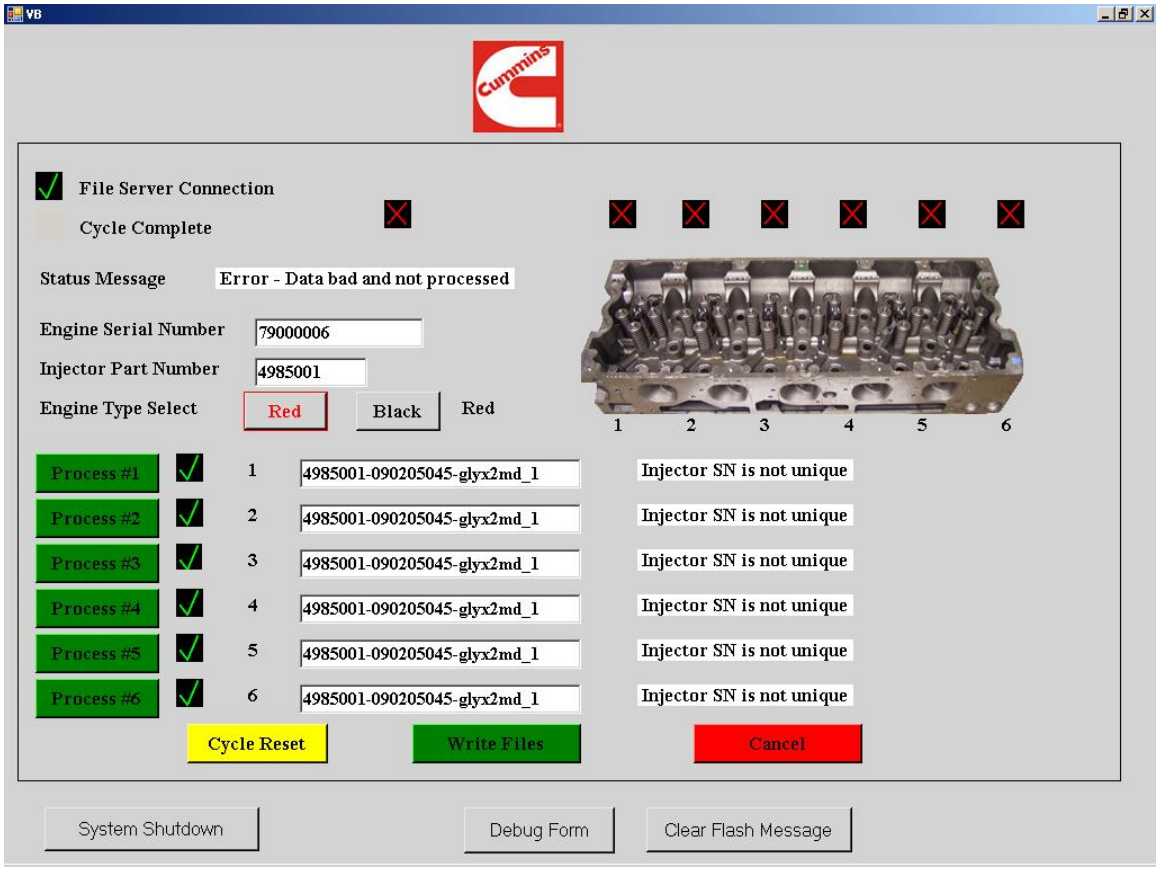


Enter a name and description, and then change the *Type* from *Local* to *Array*. You will see a new field appear for you to configure called *Array Length*. The example below shows an Array Type Variable three long called Data being created.



When you are done entering data, select the *OK* button. The system will generate the desired Array *Variable*, including it's members, and then erase the entered name, description and OPC Item. You can create another *Variable* or select the *Done* button to exit. All *Variables* must have unique names.

Graphical User Interface



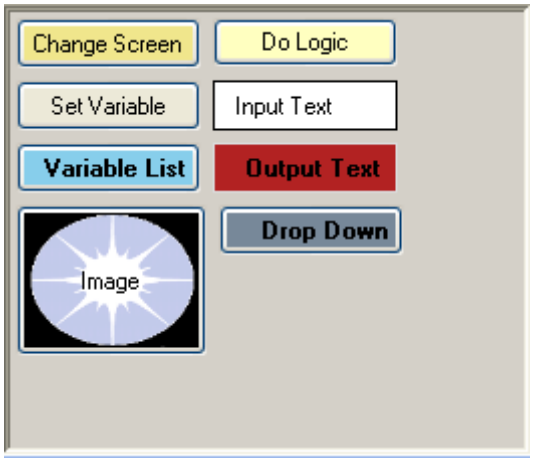
Example screen developed within PlantWatch

Overview

PlantWatch's Screen Editor allows you to build screens that have animation on them as well as the capacity to change the current display to a different *Screen*.

There are eight types of animation in the *Animation Toolbox*

- Change Screen
- Output Text
- Input Text
- Set Variable
- Image
- Do Logic
- Variable List
- Drop Down



Animation Toolbox

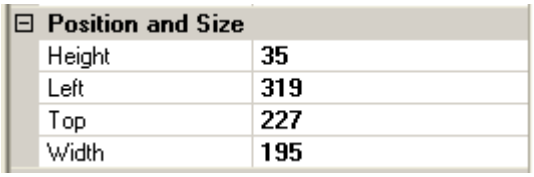
You can have multiple *Screens.*

The first *Screen* you make will be the *Screen* that is displayed when PlantWatch starts

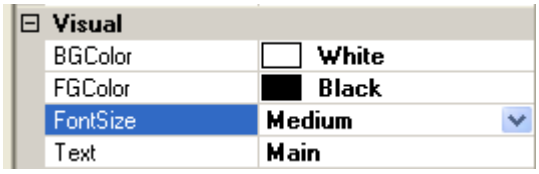
When PlantWatch is running you can switch to a different *Screen* with a *Change Screen* button animation.

Each animation type has its unique parameters to control its behavior.

Most animation types have setting for Position and Size. Animations can be positioned and sized by either using a mouse to stretch and drag or by entering numeric data into the dialog.

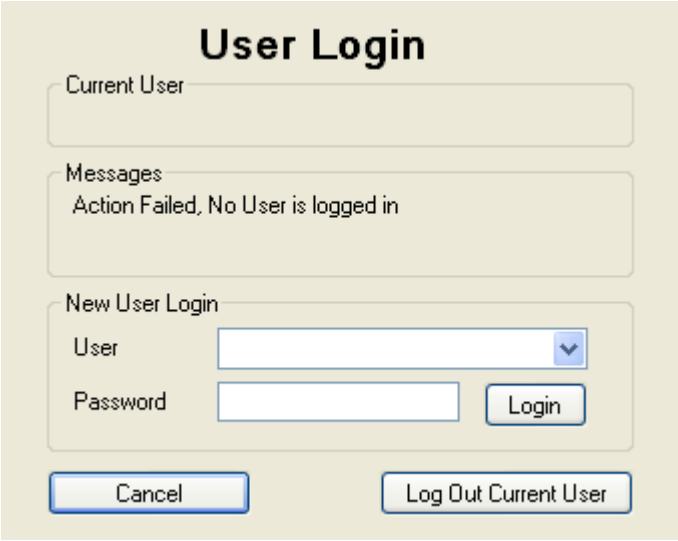


Most animation types have setting for Visual attributes such as foreground color, background color, text and font sizes of Small, Medium and Large. Animations can be positioned and sized by either using a mouse to stretch and drag or by entering numeric data into the dialog.



Some animations have security built into them with dialog that allows you to select any of the security levels built into the system. If a Security level is selected, the animation will not work unless a valid user is logged into Plantwatch.

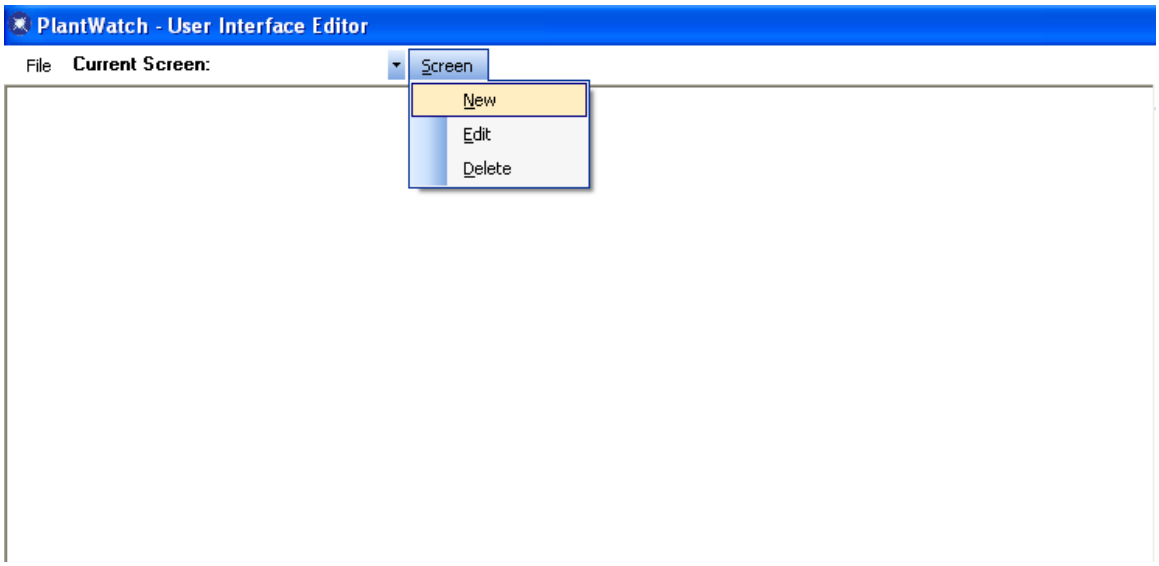
If an animation is invoked without a valid user logged into Plantwatch, the login dialog will appear.



This will allow the user to log in as a valid user. Then the user will be able to use the secured animation.

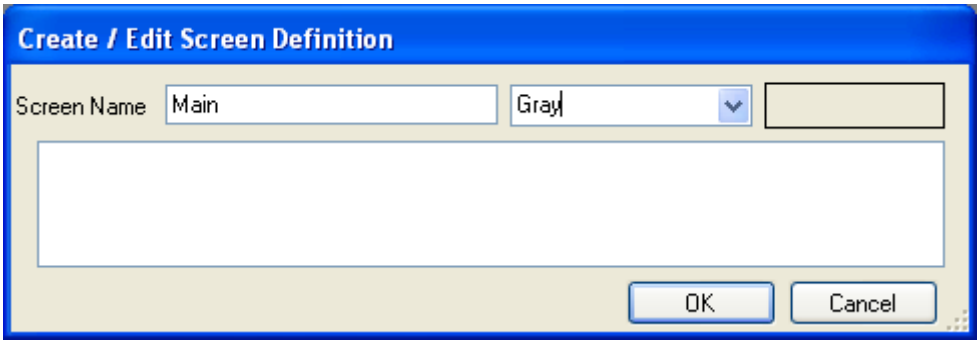
Adding a Screen to the application

When an application is initialized, there are no screens. Therefore if you desire to use the Graphics features you must first create at least one *Screen*.



To create a new screen, Click on *Screen* at the top of the Editor window, and then click on *New* as shown above.

You will be presented with a dialog that allows you to name your new screen. Enter your desired screen name into the *Screen Name* field, enter your screen background color and then select OK. The image below shows creating a new screen called *Main* with a *Gray* background.

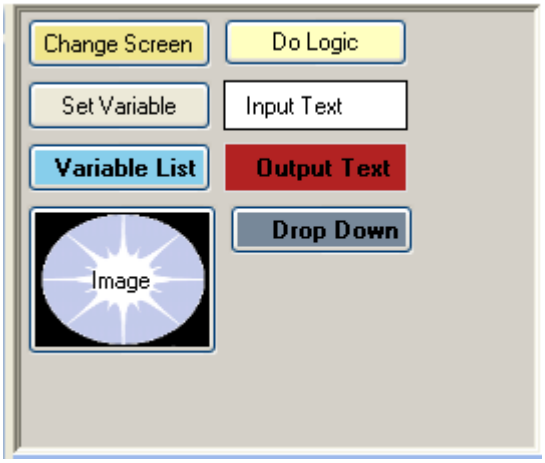


Note that the first Screen you create in an application will be the first screen that is displayed when PlantWatch starts.

Adding a Animation to the current Screen

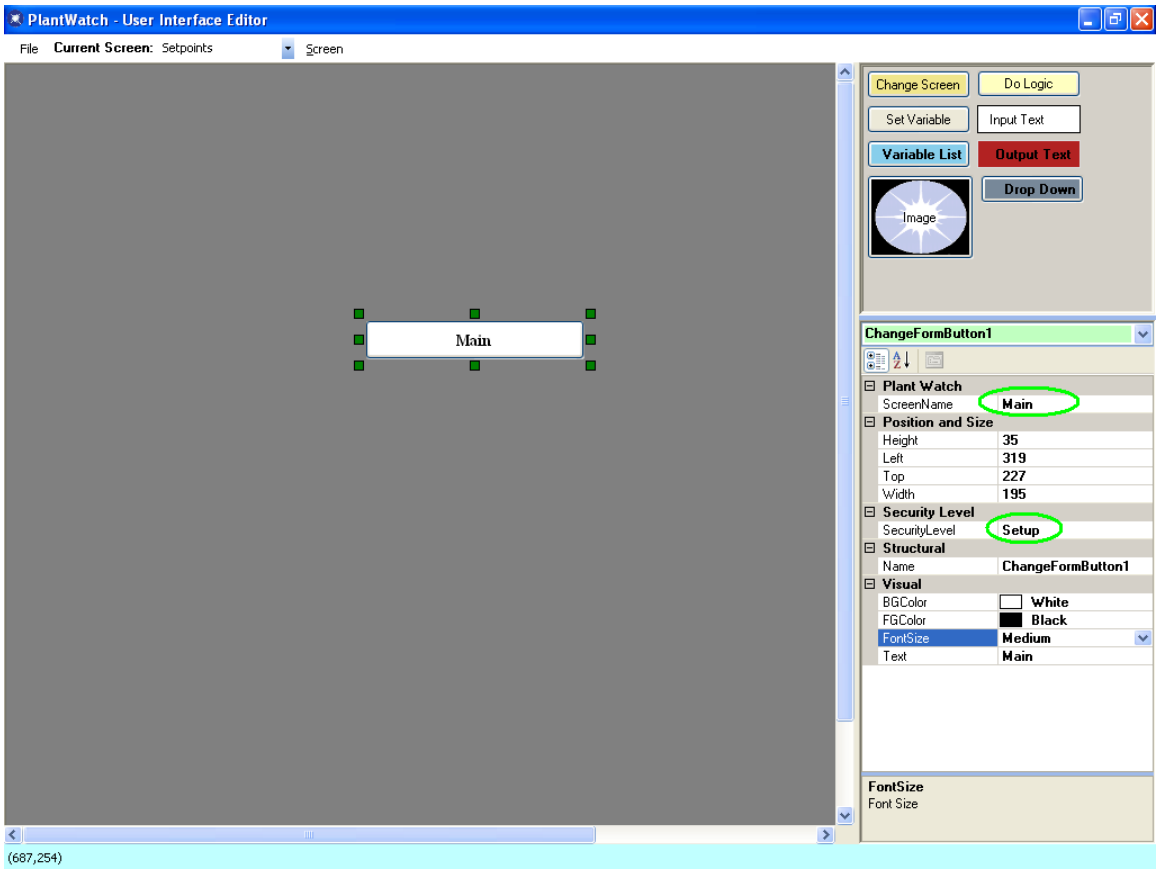
Clear any current selections by clicking on any empty area of the screen.

Click on the desired animation type in the Animation Toolbox and drag it onto the screen to the location you want it to be. Then release the mouse.

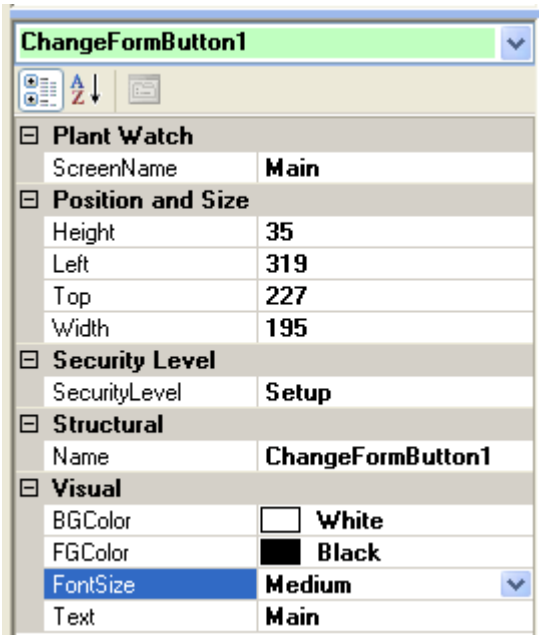


Animation Toolbox

After releasing the mouse, the animation will appear on the screen with 8 green boxes defining the animations outline.



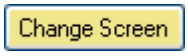
Below the toolbox you will be able to see and edit the animations settings. Note that each animation has different settings.



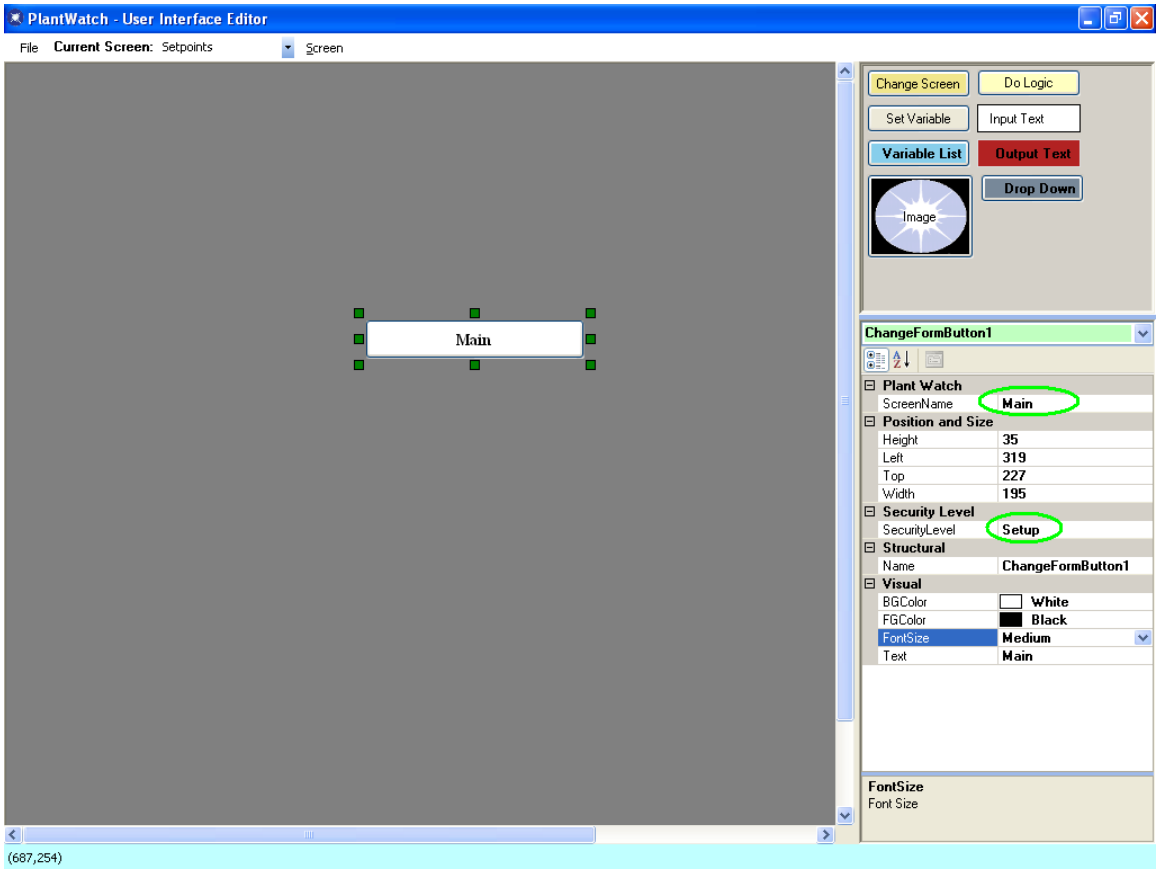
Screen Change animation settings

Whenever this animation is selected you will be able to edit it's associated fields.

Change Screen



A *Change Screen* animation is a button.
When a *Change Screen* animation is clicked on, the currently displayed *Screen* will be replaced with the one configured in the *Change Screen* animation.



Settings for *Change Screen* include the *Screen Name* you want to go to and the *SecurityLevel* required.

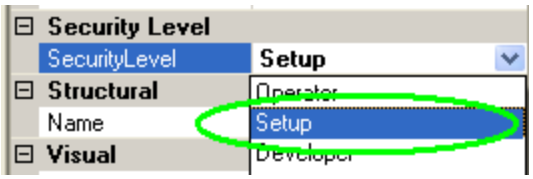
The size of the button can be adjusted by the eight green handles located on the animation within the Position and Size panel.

As with most of the animation types you can also adjust the Visual attributes of foreground color, background color and fontsize.

Unlike most animation types, the Text field is filled in by the system with the name of the screen you are going to.

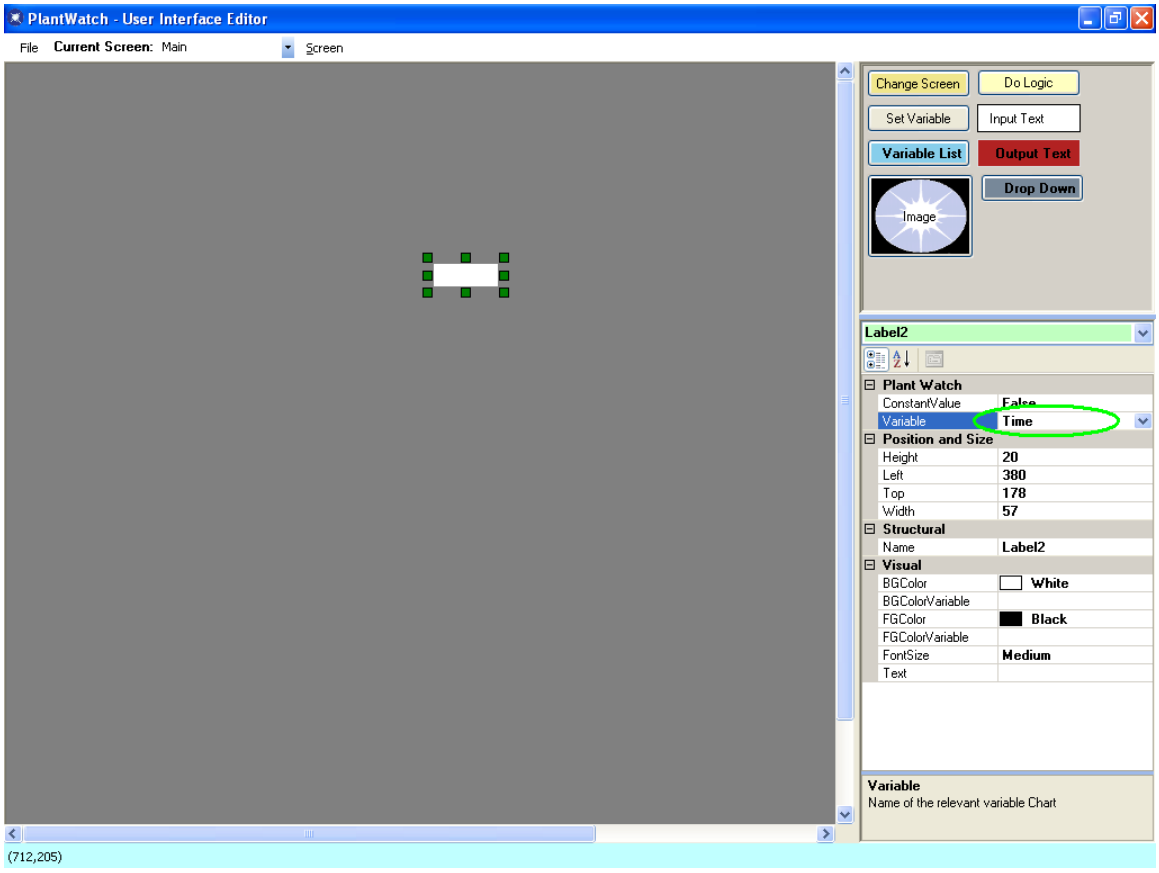
This animation type has a setting for the security level needed to invoke the action. If left blank then no security is applied.

To enable security, simply select one of the previously configured security levels available from the selection offered. The default levels include *Operator*, *Setup* and *Developer*.

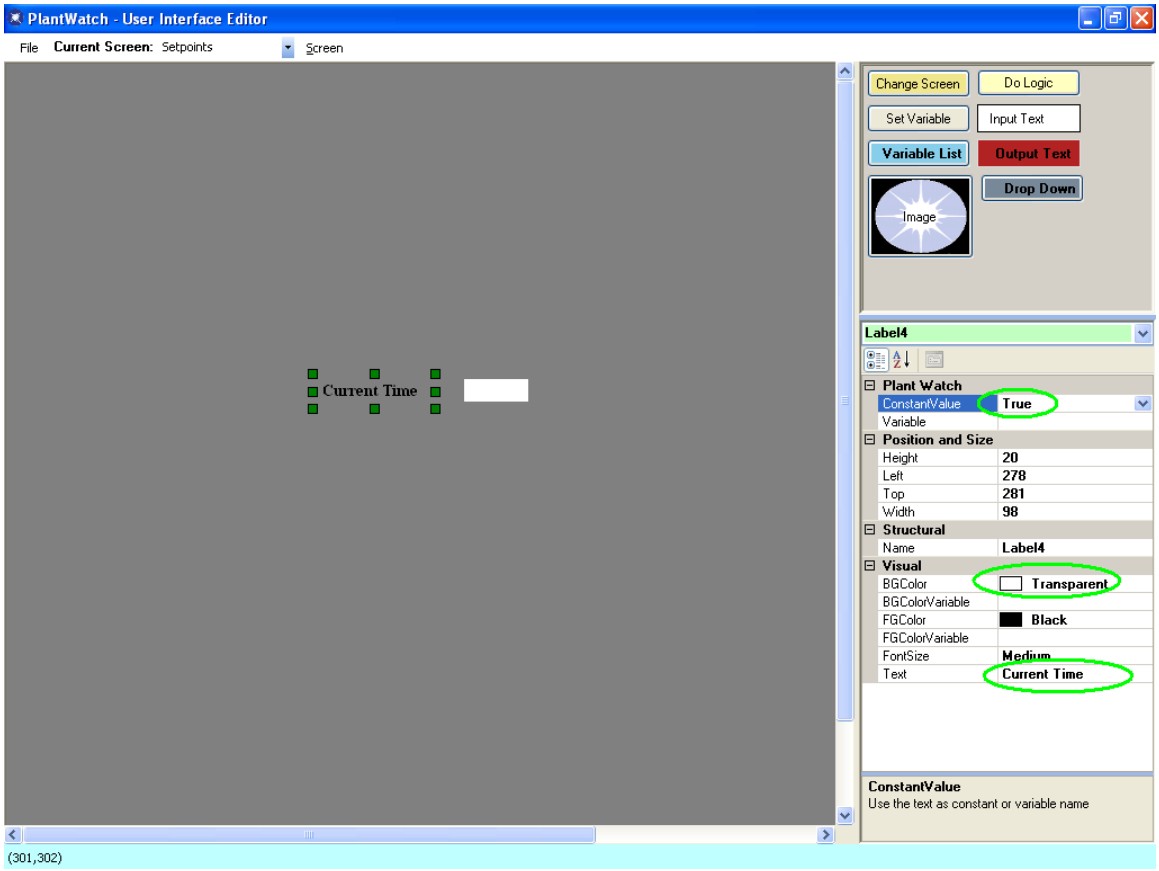


Output Text

An *Output Text* animation will display either a constant value or the value of a variable. If the field *Constant Value* is set to *True*, the *Output Text* will display the never changing value that you type into the field *Text*. If the field *Constant Value* is set to *False*, the Output Text will display the current value of the variable you must enter into the field *Variable*.



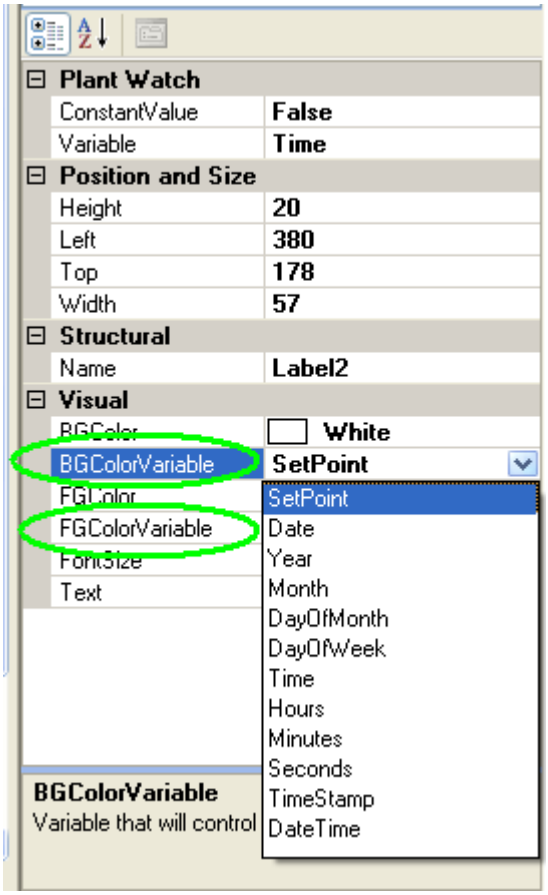
Settings to display the variable Time



Settings to display the constant text *Current Time*

As with most of the animation types you can also adjust the attributes of Position, Size and Visual.

It also has an additional two settings to allow you to control the foreground and background colors at runtime.



Runtime Color Control Settings

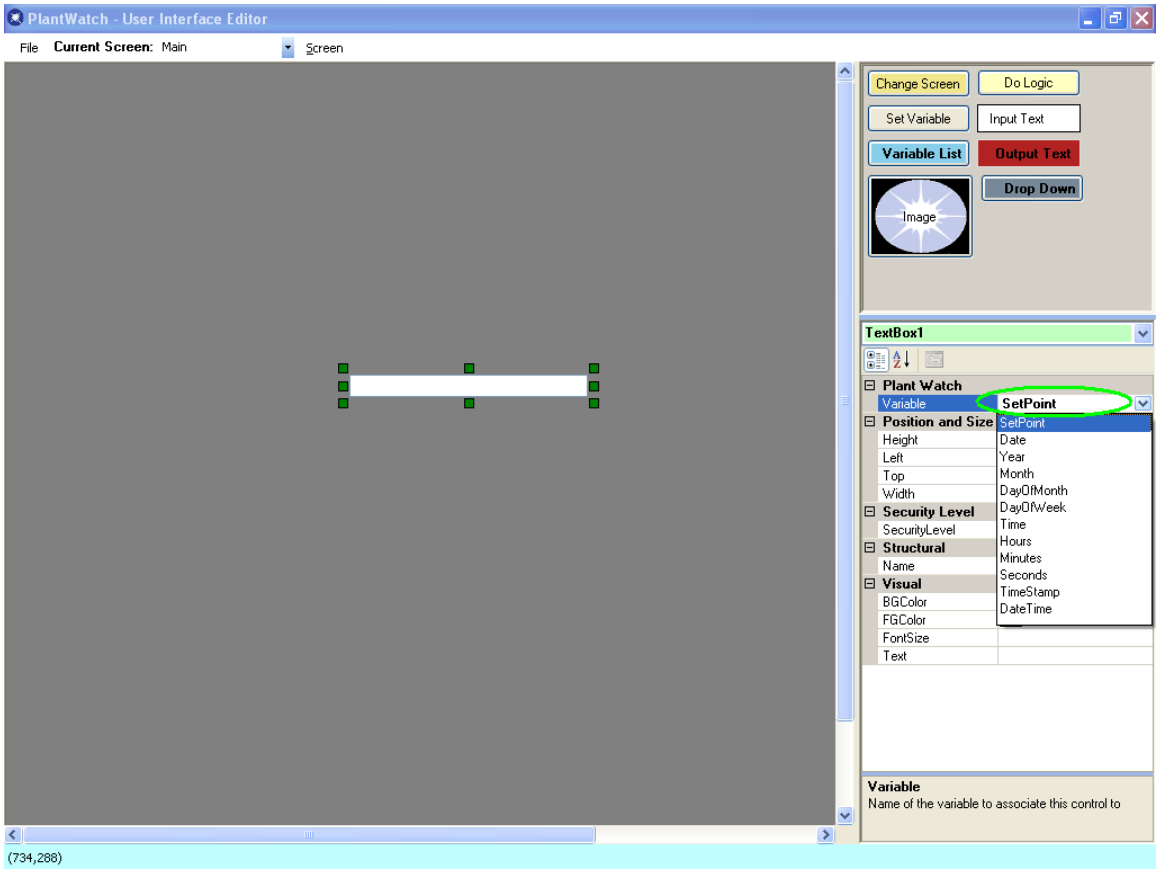
If you populate the *BGColorVariable* field with a variable, at runtime the value of the variable will be used to control the animations back ground color value.

The foreground color is controlled in the same fashion.

The value of the variable can be any valid color name understood by the operating system. Valid color names can be found by looking at the back ground color setting, where you will find names like *Firebrick*, a dark red.

Input Text

Input Text animation take input from the keyboard and puts it into a *Variable* and also displays the current value of the variable. The result is that a user will always be presented the value in the variable unless he is actively changing it. The field *Variable* allows you to select the variable that interacts with the Input Text animation.

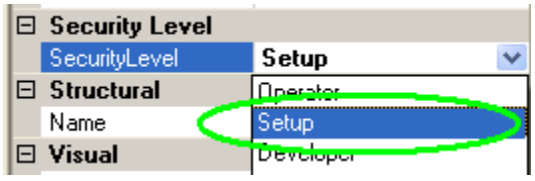


Settings to enter data into variable *SetPoint*

As with most of the animation types you can also adjust the common attributes of location, size and colors.

This animation type has a setting for the security level needed to invoke the action. If left blank then no security is applied.

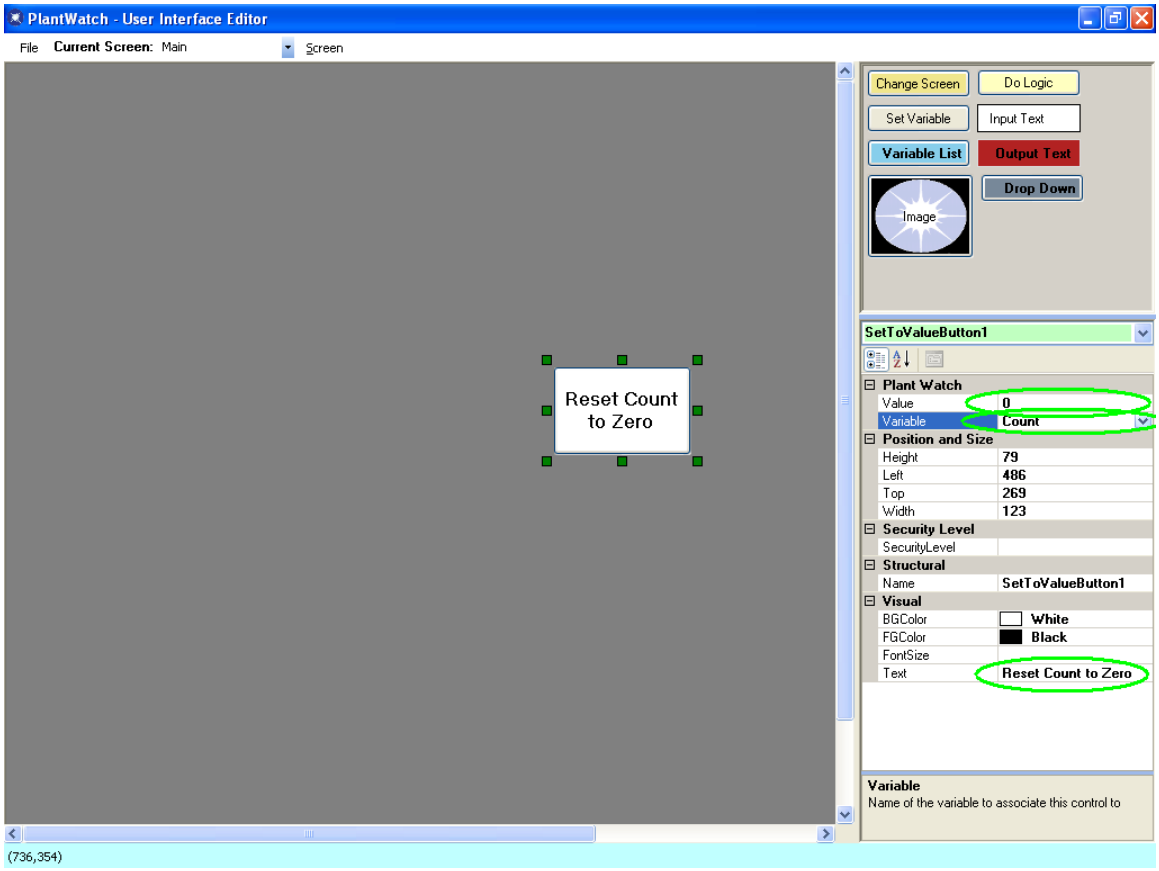
To enable security, simply select one of the previously configured security levels available from the selection offered. The default levels include *Operator*, *Setup* and *Developer*.



Set Variable

A *Set Variable* animation is a button.
Set Variable animations change the value of a variable to a pre-determined value. An example is resetting a count back to zero. There is a drop down box where you select the

variable you wish to change and a place to enter the value you want the variable to be set to. You also control the text in the button



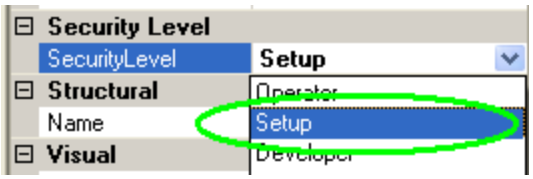
Settings to change the value of variable *Count* to zero

The size of the button can be adjusted by the eight green handles located on the animation.

As with most of the animation types you can also adjust the common attributes of location, size and colors.

This animation type has a setting for the security level needed to invoke the action. If left blank then no security is applied.

To enable security, simply select one of the previously configured security levels available from the selection offered. The default levels include *Operator*, *Setup* and *Developer*.

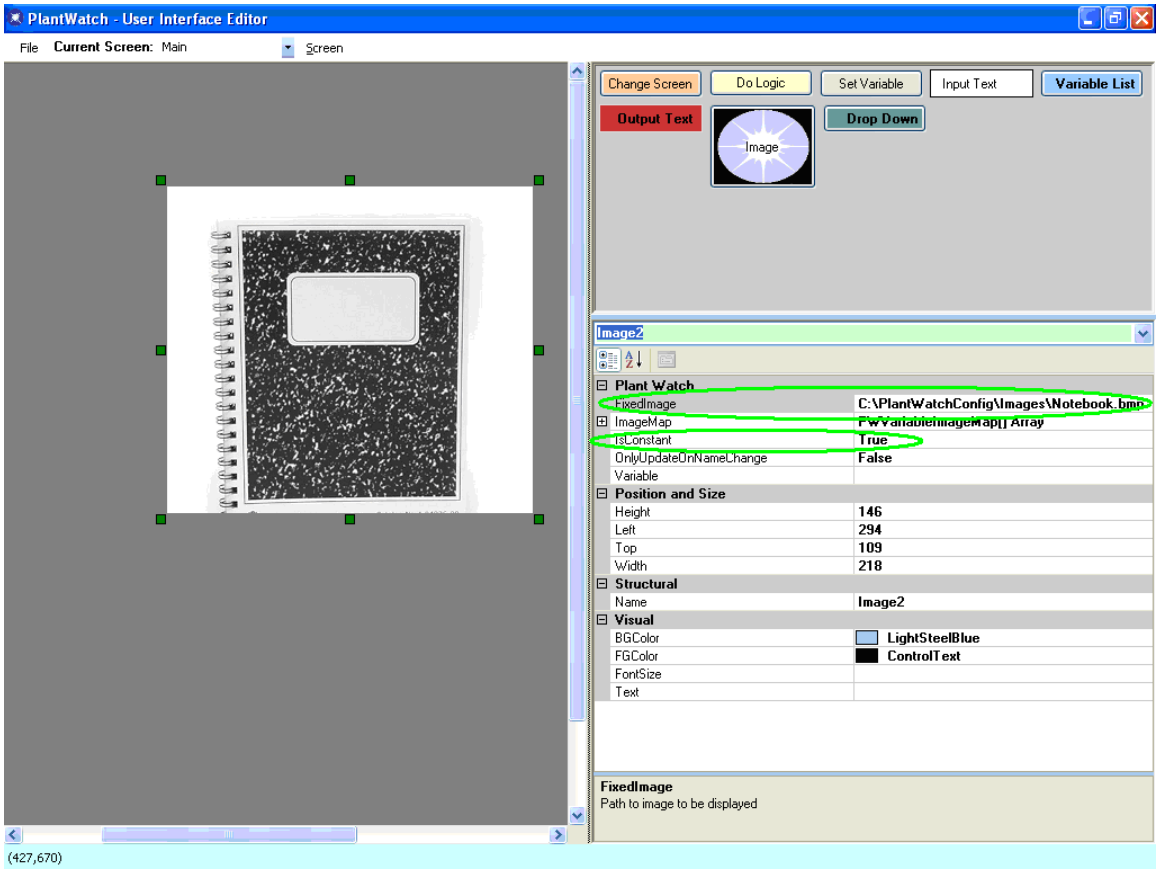


Images

An Image animation can display a constant image, or display an image referred to by a *Variable*.

The field *IsConstant* determines if the Image animation always displays the same image or if the Image animation displays an image based on the content of a variable.

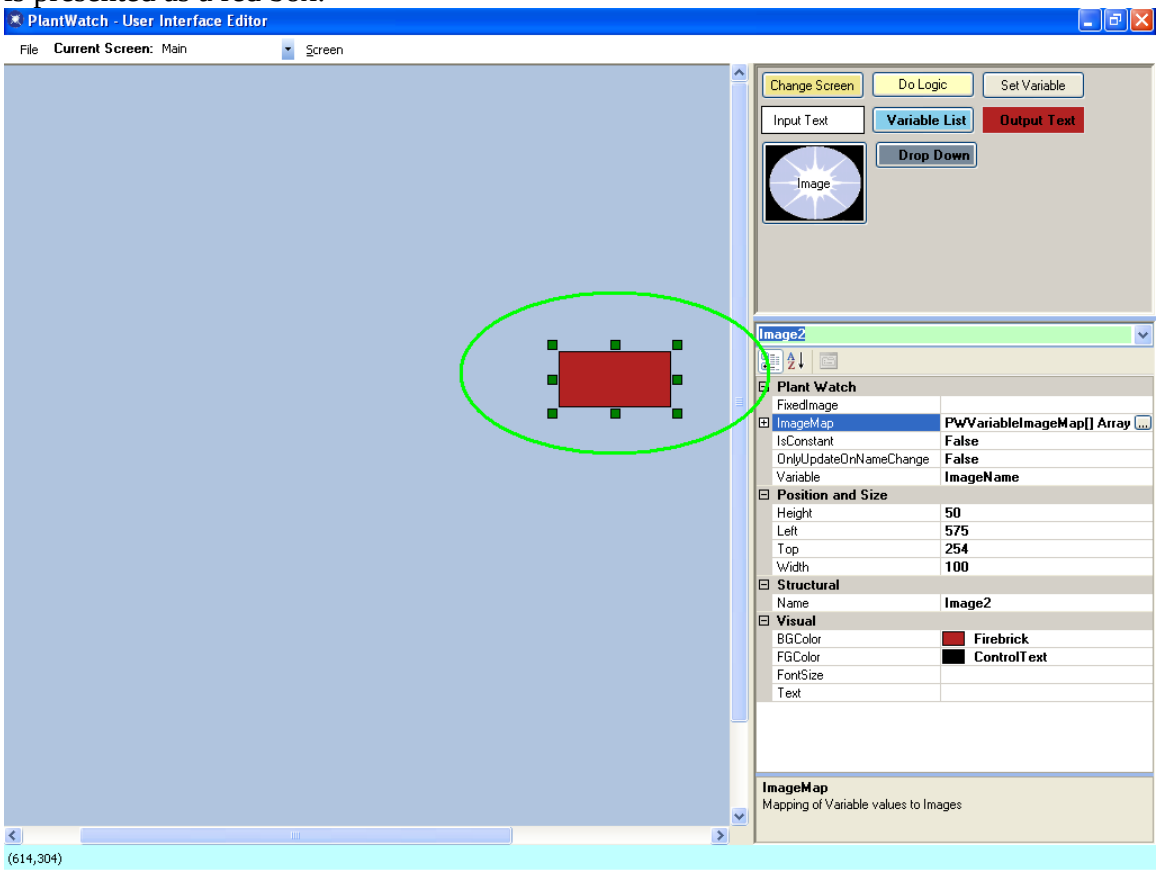
If the field *IsConstant* is set to *True* the Image animation will display the image specified in the *FixedImage* field. As soon as the configuration is entered the image will appear within the screen.



Settings to always display image notebook.bmp

You can also simply drag an image onto the screen. It will automatically setup a constant image configuration.

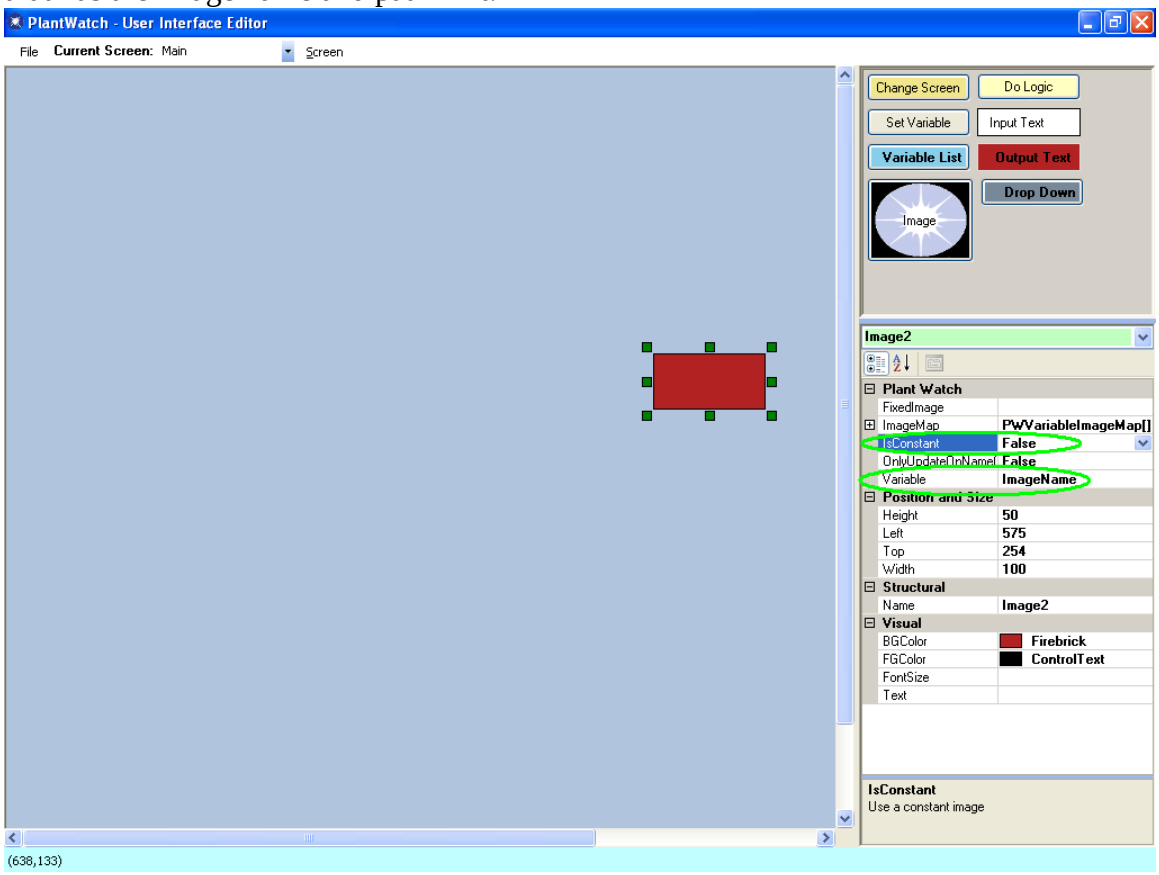
If the field *IsConstant* is set to *False* the Image animation will display an image based on the content of the variable specified in the *Variable* field. Note that this means during development the image to be displayed is not determined, therefore the Image animation is presented as a red box.



This can work one of two ways.

> If no entries are made to the *ImageMap* field, the image presented will always be exactly the file specified by the Variable content. This means that variable must contain a valid path and file name. This allows you to present any image during runtime without knowing the image name in advance.
>
> If entries are made to the *ImageMap* field, the image presented will always be determined by matching the content of the variable to *Key* in Image Set Selection dialog. This works well when you have a pre-determined set of images that you want to choose one from to display.

If you want to display the what ever image whose path and filename are contained within a variable, drag a new image onto the screen set the field to true and select the variable that has the image name and path in it.

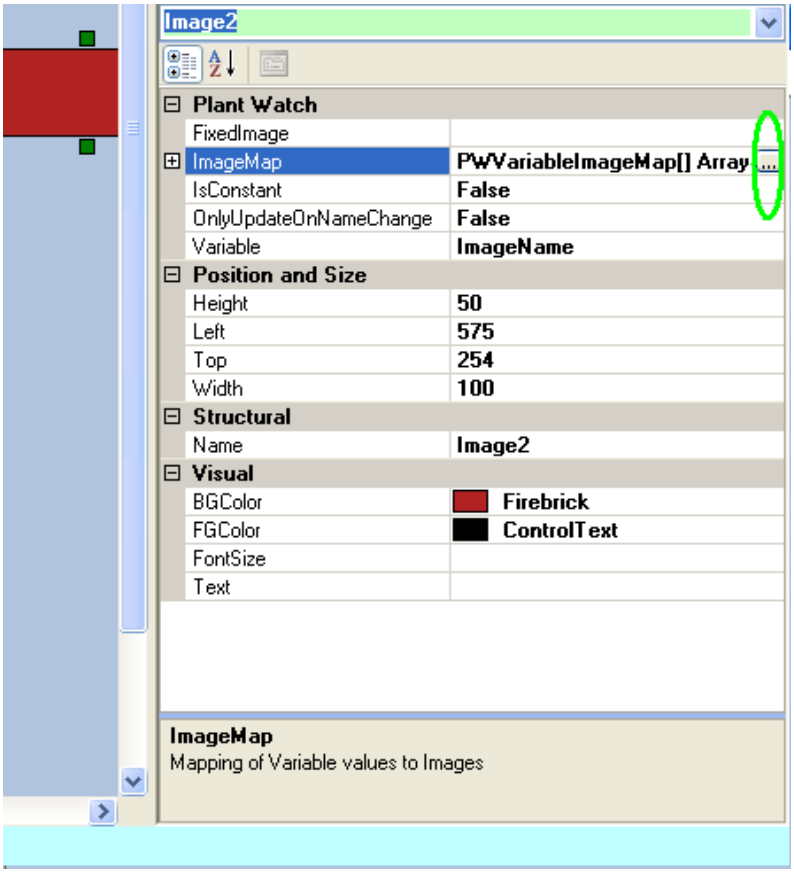


Settings to display the image whose path and name are in the variable ImageName

If you have a pre-determined set of images that you want to choose one from to display, you would follow the above instructions and then configure the *Image Map*.

The *Image Map* allows you to choose up to 8 specific images and then assign to each image a key value. If the chosen variable has one of the key values entered, the associated image will be presented.

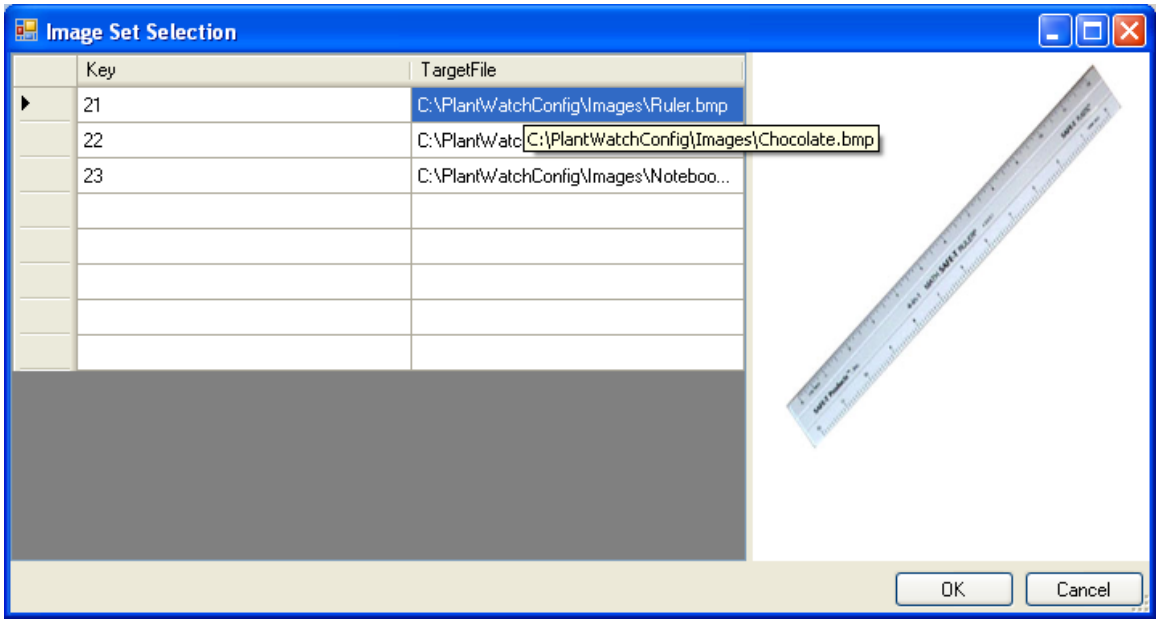
To configure the *Image Map* click on the text *PWVaribaleImageMap[]*, and you will be presented with three dots to the right of the text. Click on the three dots to open the *Image Map* editor.



Click on these three dots to open the Image Map editor

Associate a key and the path and filename for each Image you wish to display. The example below has three possible images it can display. For it to display anything the ImageName variable must equal 21, 22 or 23.

If the value of ImageName is 21, the ruler image will be displayed.
If the value of ImageName is 22, the chocolate image will be displayed.
If the value of ImageName is 21, the notebook image will be displayed.



Settings to display one of three images

As you select the images to display you will see them within the *Image Map* dialog.

If the image file does not change, by setting *OnlyUpdateOnNameChange* to *True,* you can stop the unnecessary action of continuously reading the file in from the hard drive.

Do Logic

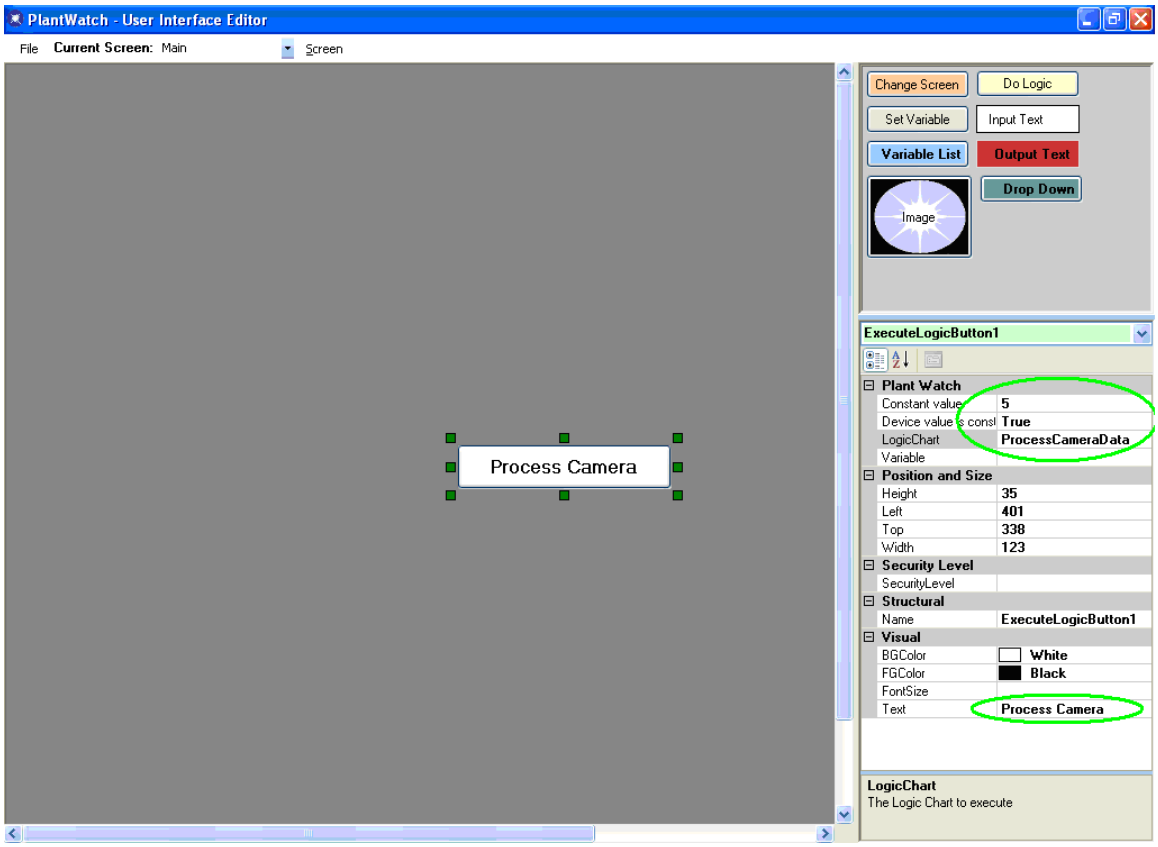
A Do Logic animation is a button that can cause a logic chart to execute.

Configuration includes selecting the Logic Chart to execute and specifying the value to use as the device value. The value to be used as the device value is used in the logic chart anytime a *logic cell* refers to the *Device* as the data source.

The value to use as the device value can either be a constant or refer to the value of a variable.

To create a button that will execute logic chart *ProcessCameraData* and always send a 5 as the device value you would:

Set the Constant value field to 5
Set the Device value is const field to true
Set the Logic Chart field to ProcessCameraData

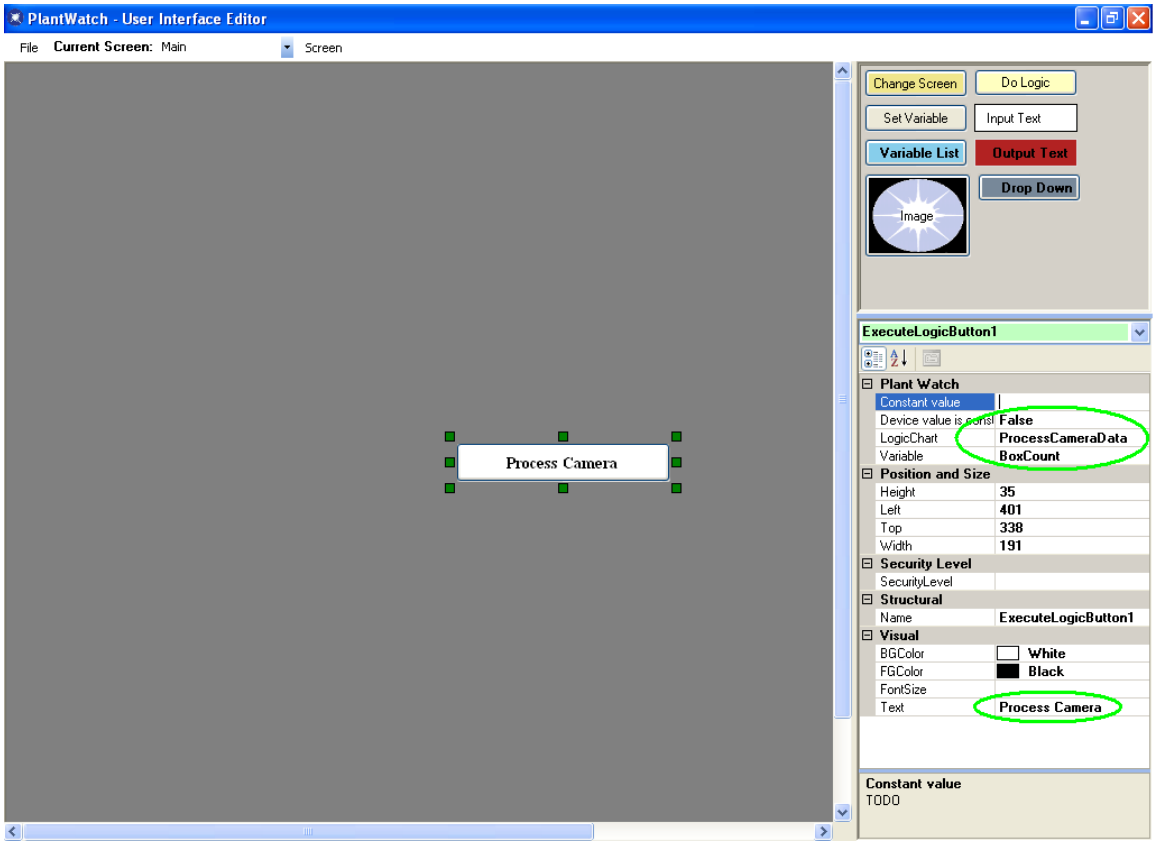


Settings to execute Logic Chart *ProcessCameraData* with a constant device value of 5

Note that the above image has also set the text of the button to *Process Camera.*

To create a button that will execute logic chart *ProcessCameraData* and send the contents of variable BoxCount as the device value you would:

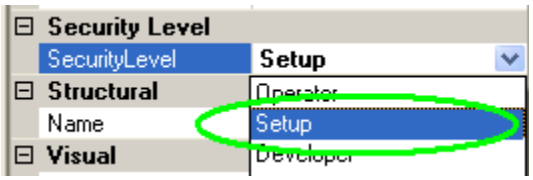
Ignor the Constant value field
Set the Device value is const field to False
Set the Logic Chart field to ProcessCameraData
Set the Variable field to BoxCount



Settings to execute Logic Chart *ProcessCameraData* with a variable device value from BoxCount

This animation type has a setting for the security level needed to invoke the action. If left blank then no security is applied.

To enable security, simply select one of the previously configured security levels available from the selection offered. The default levels include *Operator*, *Setup* and *Developer*.



Variable List

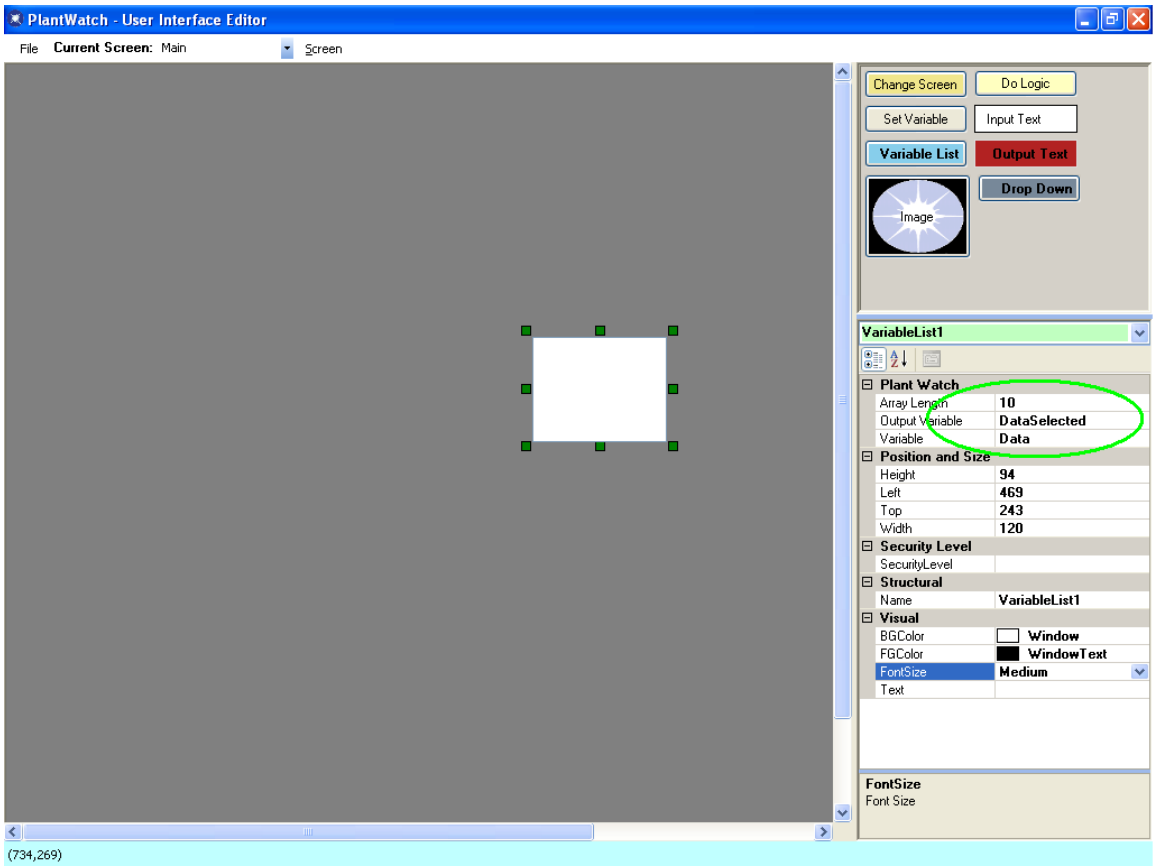
A Variable List animation displays the content of an Array Type Variable and allows a user to select any one of its many entries with the mouse. The value selected by the user will be placed into the Output Variable.

This allows a user to select one value from a list.

The height of the Variable List is set to display 6 values. If more than 6 values are displayed then scroll bars are invoked.

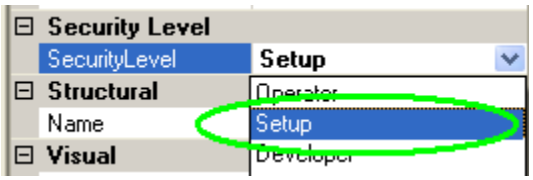
To configure a Variable List to display the first 10 values of the contents of the Array Type Variable *Data*..
And when clicked on to place the user selected value into variable *DataSelected*…
You would:

Set field *Variable* to *Data*. Note that only Array Type Variables will be displayed
Set field *Output Variable* to *DataSelected*
Set Field *Array Length* to array variable length of *10*



This animation type has a setting for the security level needed to invoke the action. If left blank then no security is applied.

To enable security, simply select one of the previously configured security levels available from the selection offered. The default levels include *Operator*, *Setup* and *Developer*.



Drop Down List

A DropDown animation is very similar to the Variable List. The significant difference is that it is always just one line high unlike a Variable List which is about 6 lines high.

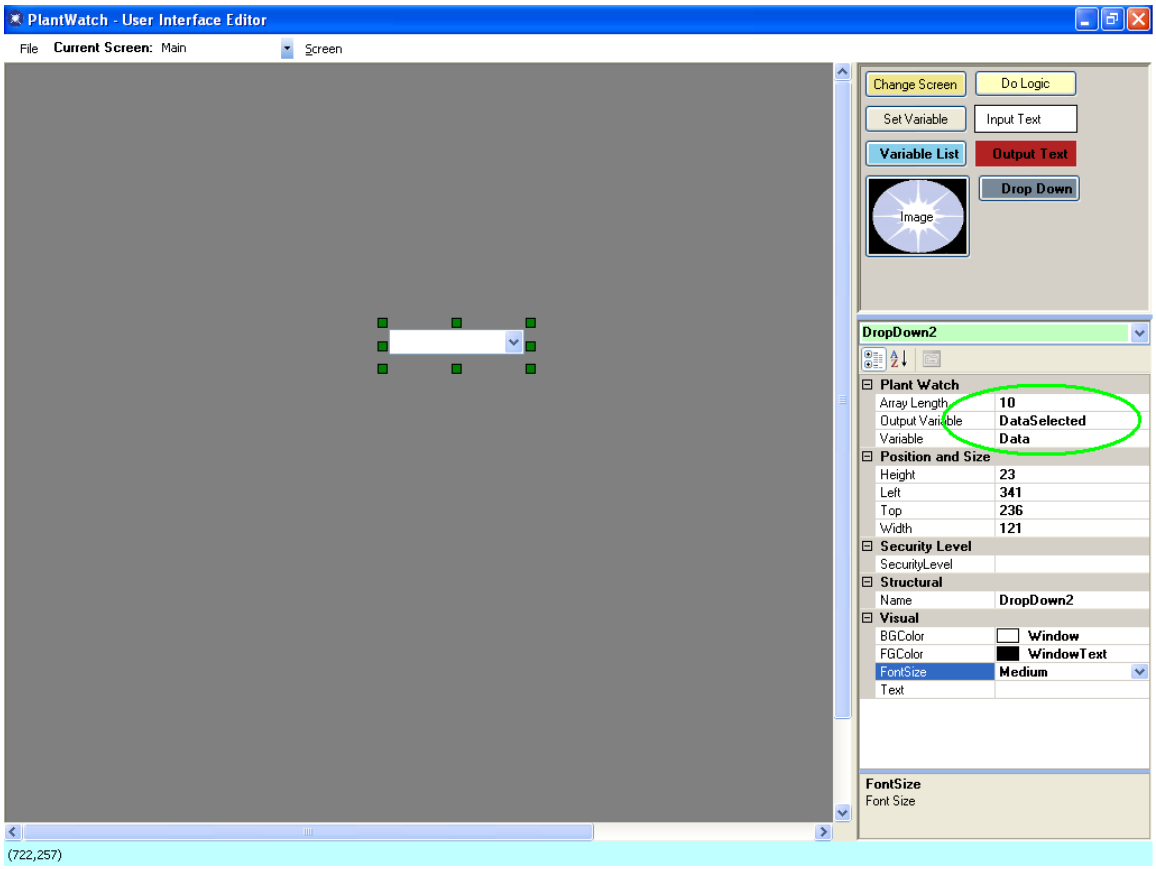
A Drop Down animation displays the content of an Array Type Variable and allows a user to select any one of its many entries with the mouse. The value selected by the user will be placed into the Output Variable.

This allows a user to select one value from a list.

The height of the Drop Down is always one except when being acted upon by a user.

To configure a Drop Down to display the contents of the Array Type Variable *Data* and place the user selected value into variable DataSelected:

Set field *Variable* to *Data*
Set field *Output Variable* to *DataSelected*
Set Field *Array Length* to array variable length of *10*

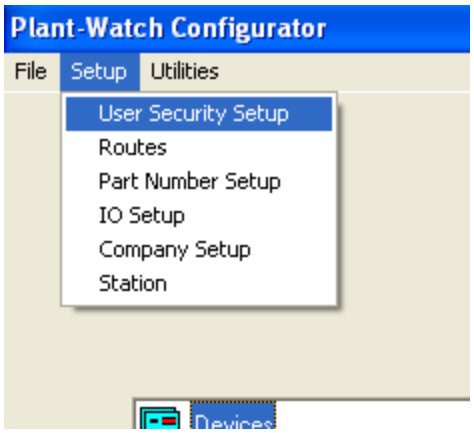


This animation type has a setting for the security level needed to invoke the action. If left blank then no security is applied.

To enable security, simply select one of the previously configured security levels available from the selection offered. The default levels include *Operator*, *Setup* and *Developer*.

Security Configuration

The creation of the *Security Levels* referenced by the Graphics animations are created with the Plantwatch Configurator. Select *Setup*, then *User Security Setup* to open the


Plant-Watch Configurator
File
Setup
Utilities
User Security Setup
Routes
Part Number Setup
IO Setup
Company Setup
Station
Devices

OPC

What is OPC?

OLE for Process Control (OPC) which stands for Object Linking and Embedding (OLE) for Process Control, is the original name for a standards specification developed in 1996 by an industrial automation industry task force. The standard specifies the communication of real-time plant data between control devices from different manufacturers.

After the initial release, the OPC Foundation was created to maintain the standard. Since then, standards have been added and names have been changed. As of June, 2006, "OPC is a series of standards specifications". (Seven current standards and two emerging standards.) "The first standard (originally called simply the OPC Specification"), is "now called the Data Access Specification", or (later on the same page) "OPC Data Access", or OPC Data Access Specification. [1]

While OPC originally stood for "OLE for Process Control", the official stance of the OPC Foundation is that OPC is no longer an acronym and the technology is simply known as "OPC". One of the reasons behind this is while OPC is heavily used within the process industries, it can be, and is, widely used in discrete manufacturing as well. Hence, OPC is known for more than just its applications within process control.

OPC within PlantWatch Overview

To gather information from an OPC device, you create Variables and select them to be OPC type Variables. This causes the Variable to be continuously updated with the value from the remote OPC device.

To send information to an OPC device you configure a Logic Chart cell as a WriteToOPC type. This allows you to send data from constants, variables and devices to the remote OPC device.

To configure gathering or sending information to OPC devices, we will create two organizational structures:

- OPC Servers
- OPC Groups

At a high level, OPC is comprised of several objects: the server, the group, and the item. The OPC server object maintains information about the server and serves as a container for OPC group objects. The OPC group object maintains information about itself and provides the mechanism for containing and logically organizing OPC items.

The OPC Groups provide a way for clients to organize data. For example, the group might represent items in a particular station. The rate that an OPC server provides the data changes to Plantwatch is configurable as the group refresh period.

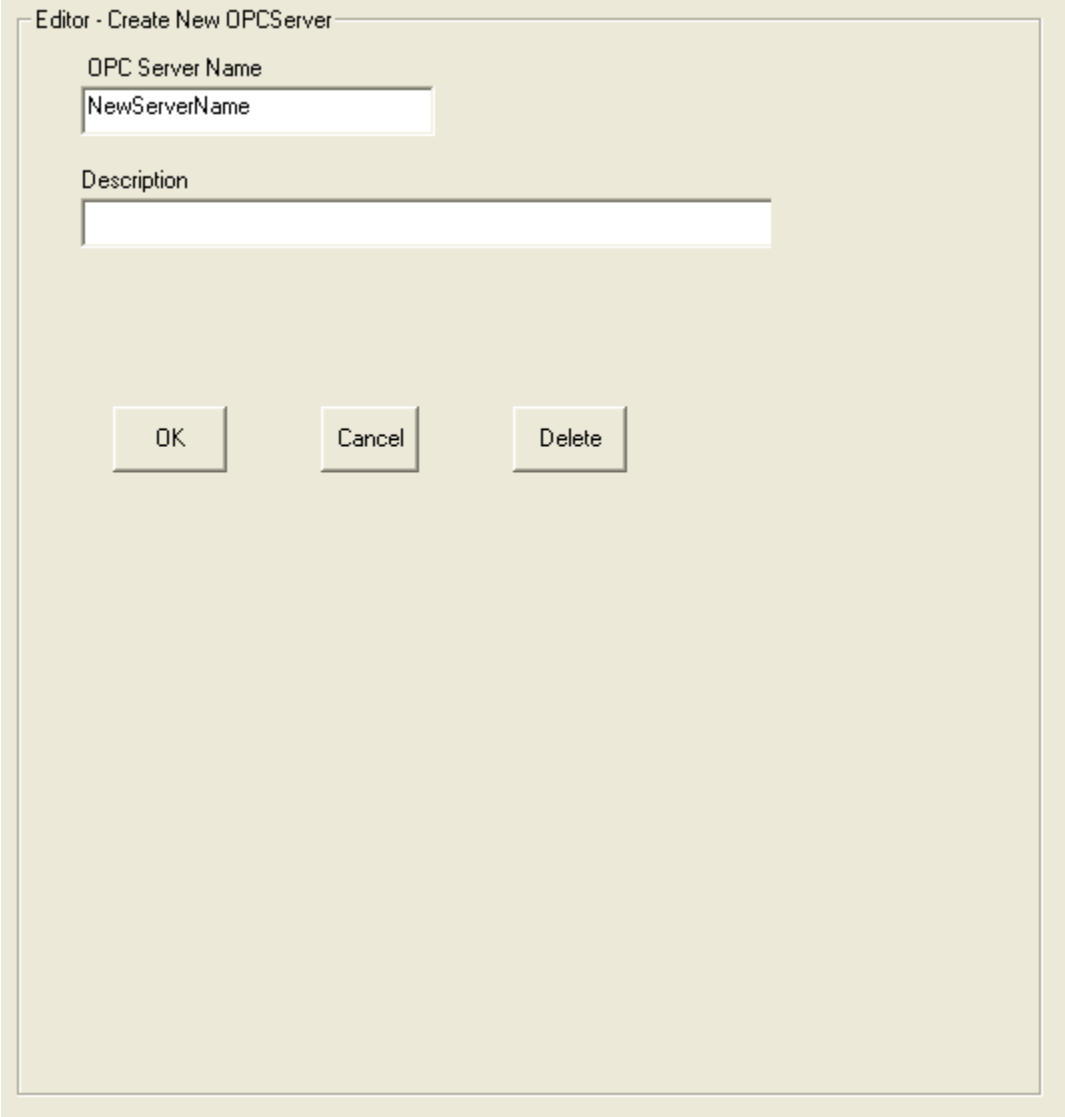
Each time that PW is configured to send data to an OPC device, a server and group is selected from an existing list. Then the OPC item value is entered.

Servers and Groups

Configuring OPC Servers

When configuring PW for communication to OPC, the first thing to do is to create a connection to the OPC server that you want to communicate to. This is done with the OPC Servers configuration.

Right click on OPC servers in the Application Tree View, and select *New OPC Server.* Then enter the formal name of the OPC server, as specified by the manufacturer.



Note that you must use this new OPC Server in a group or it will not be created.

Create a Group for the OPC server

Now that you have a reference to the OPC server, you will create groups that have an associated refresh rate. To create a new Group right click on the OPC server you want to create a group for, and select *New OPC Group* button.

The configuration needed is a name, description and refresh rate.

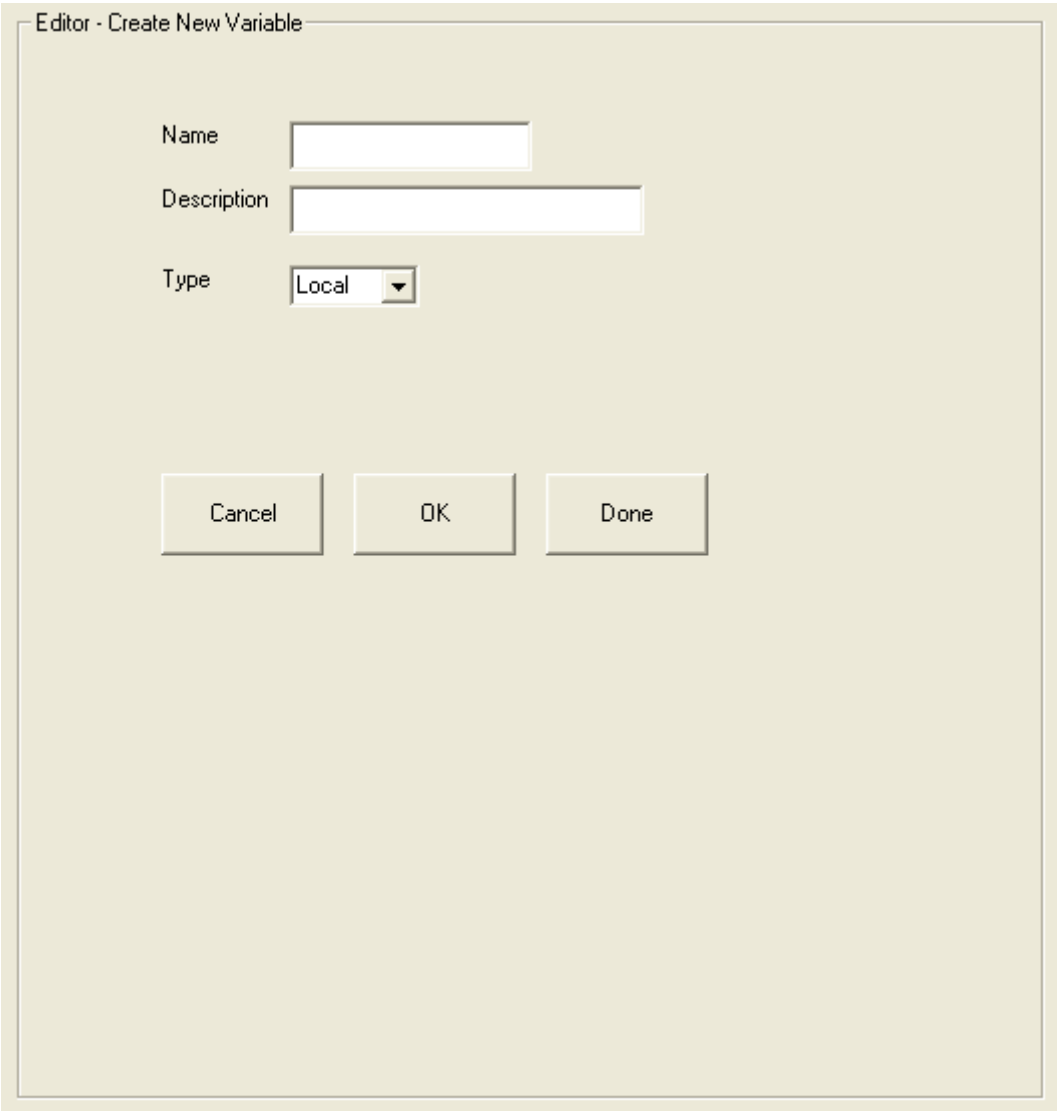
Editor - Create New OPC Group
Name
MotorStatusGroup
Description
Refresh Rate in .001 Seconds
1
OK
Cancel
Delete

Getting OPC Data into PW.

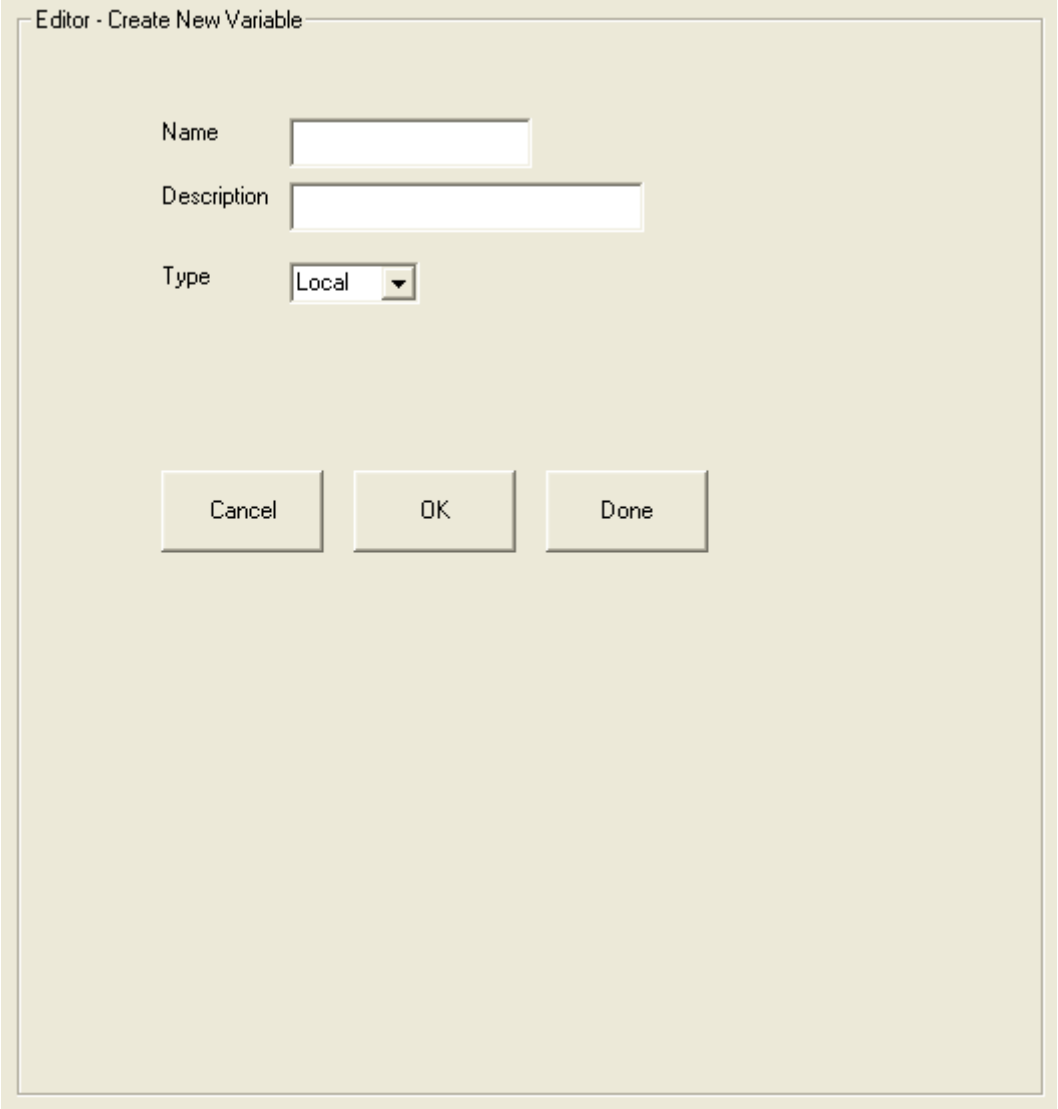
To get information from an OPC device into PW, you create a Variable of OPC type.

Navigate to the Variables configuration within the application tree.

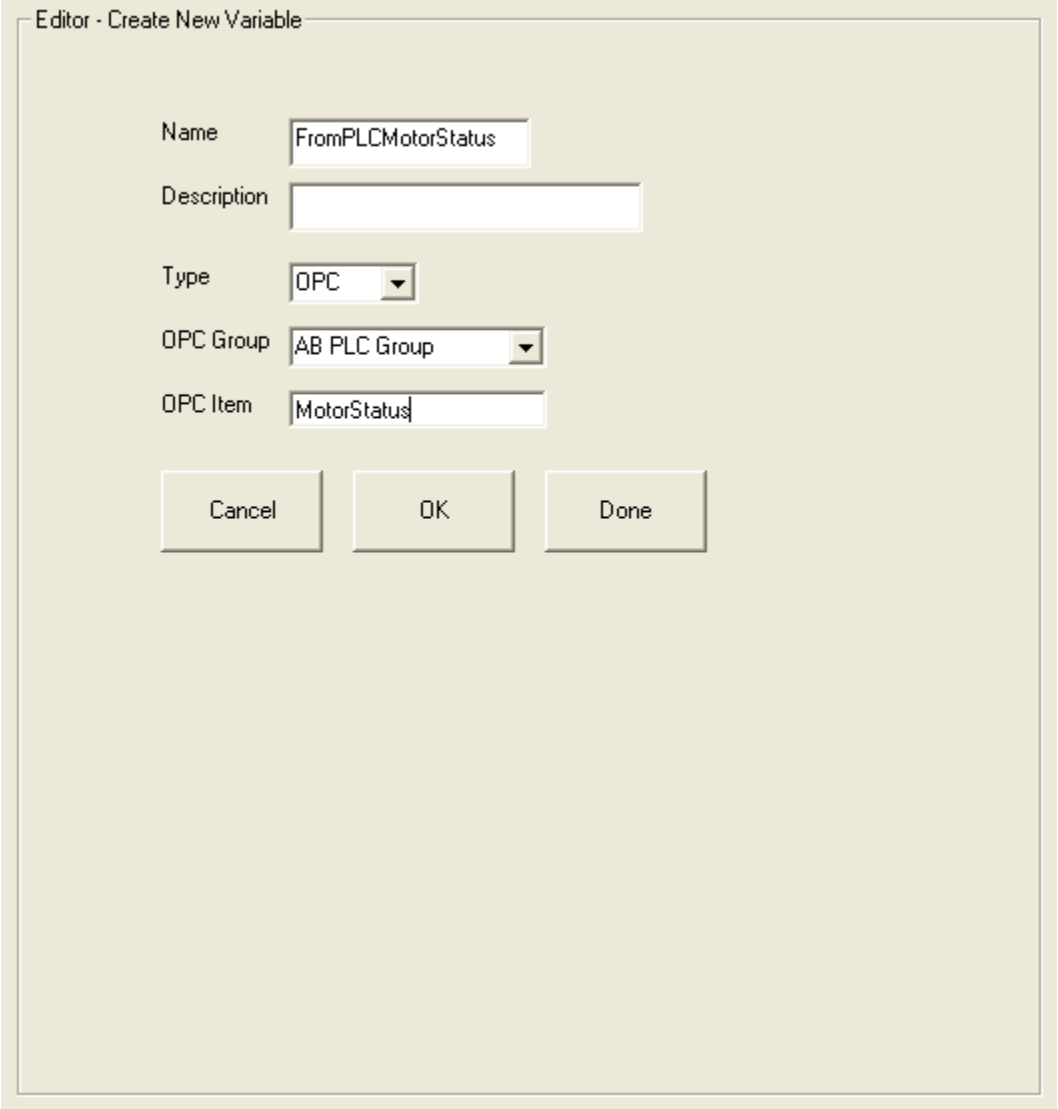
You will see a listing of all of the variables. To create additional variable, right click on Variables, and select *New Variable.*



Select the *Create New Variable* button and additional dialog will be exposed allowing you to enter in the name, Type and description of a new variable.



By default the *Type* will be set to *Local.* To use the variable to bring in OPC information, change the type setting to OPC. This will expose configuration allowing you to select the OPC group to put this variable into. It will also allow you to enter in the actual OPC items name. The example below will bring the data from OPC item MotorStatus into PlantWatch variable FromPLCMotorStatus



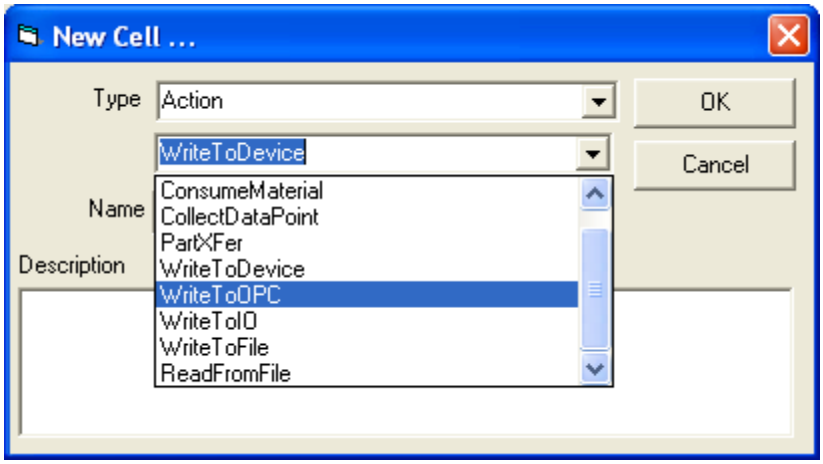
This new variable will now be connected to the groups associated server and will be populated with the value for the item at the rate selected within the group.

Getting OPC Data out of PW.

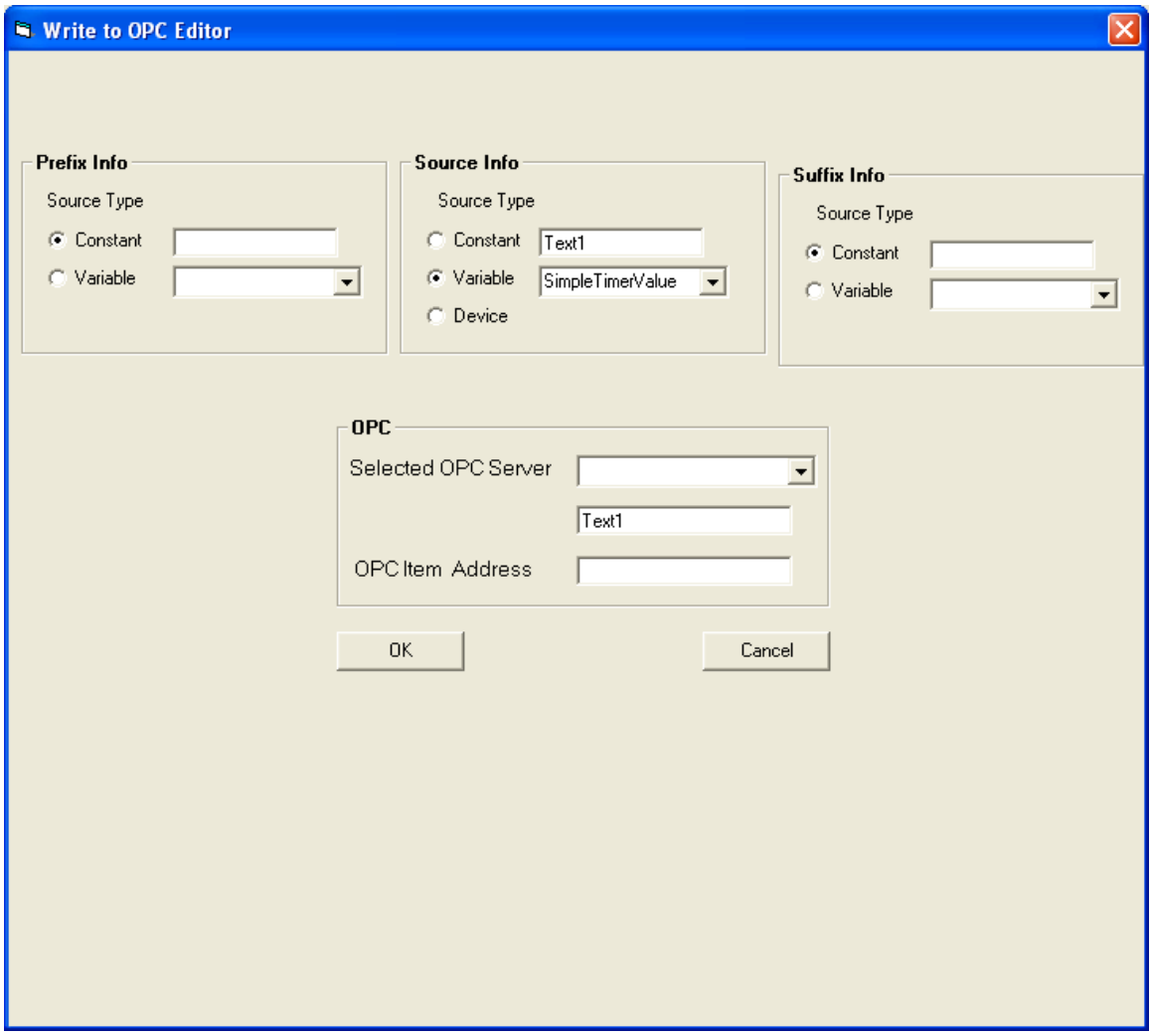
To get information from PW into an OPC device, you use the Logic Charts, selecting a Action cell type with a sub action of WriteToOPC.

(To learn how to use the Logic Charts see the tutorial earlier in the document.)

Start off by activating a cell.



After accepting the base configuration of the cell, open the cell and you will be able to select the OPC server and enter the address the information is written to, as well as what is to be written.



Configuration

Prefix, Source and Suffix frame – All three of these fields will be appended together to create a single string. That string will be sent to the OPC items address. For each of the Prefix, Source and Suffix there are several ways information can be generated. This applies to many logic chart cell types.

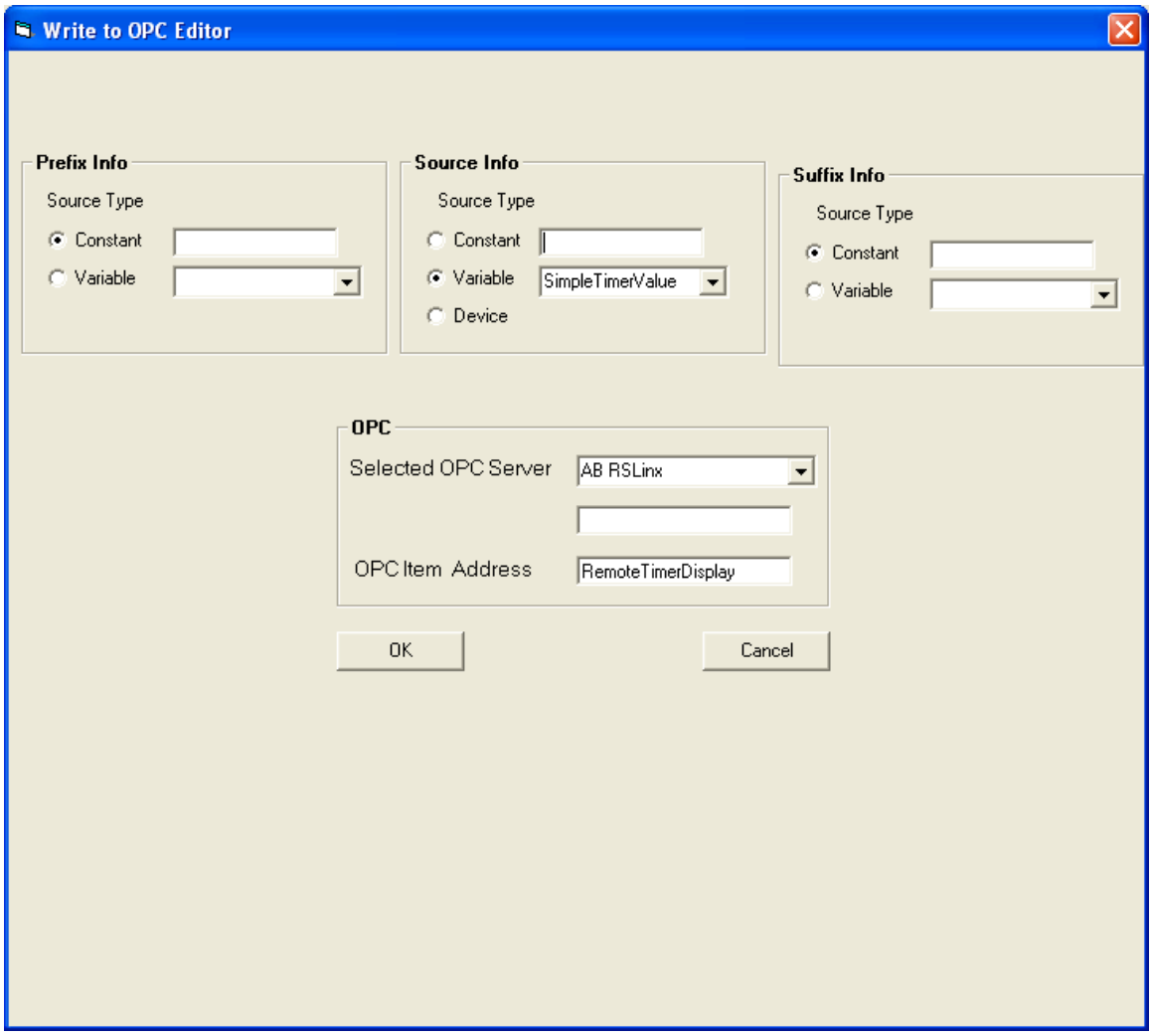
Constant – Configuration person enters in exactly the information desires
Variable – Data is taken from the referenced variable at run time
Device – Data is taken from the device value that triggered the logic chart.

OPC frame – Here you will select an existing OPC server and then enter in the item address on the server that you want the data written to.

For our example, I will write the value of or *SimpleTimerValue* variable to an item address of *RemtoteTimerDisplay* within OPC server AB RSLinx

To do this we will

Select a constant of nothing for both suffix and prefix
Select the variable SimpleTimerValue
Select existing OPC server AB RSLinx
Enter in the OPC Item Address of RemoteTimerDisplay



Select *Accept Edits* to save the configuration.

IO

Hardware

PlantWatch's input output system uses the Turck IO system components.

The Turck system consists of Racks which have *Slots* that are populated with a variety of different *Cards,* digital or RF Reader.

There are single slot racks, multi slot racks and IP 67 rated racks.
Currently digital IO and RF ID reading are supported thru the IO system.

FEN20-16DXP

BL67 Modular I/O System

Each rack is set to an address and then filled with IO cards as needed.

Each rack must have 24 VDC and an Ethernet connection to the PC.

Incoming Data

Data comes into PlantWatch from an IO point thru a Variable. Each input point is mapped to a *Variable*. After being mapped to an input point, a *Variable* will always reflect the value of the input.

If it is a digital type input the value will be either one or zero.

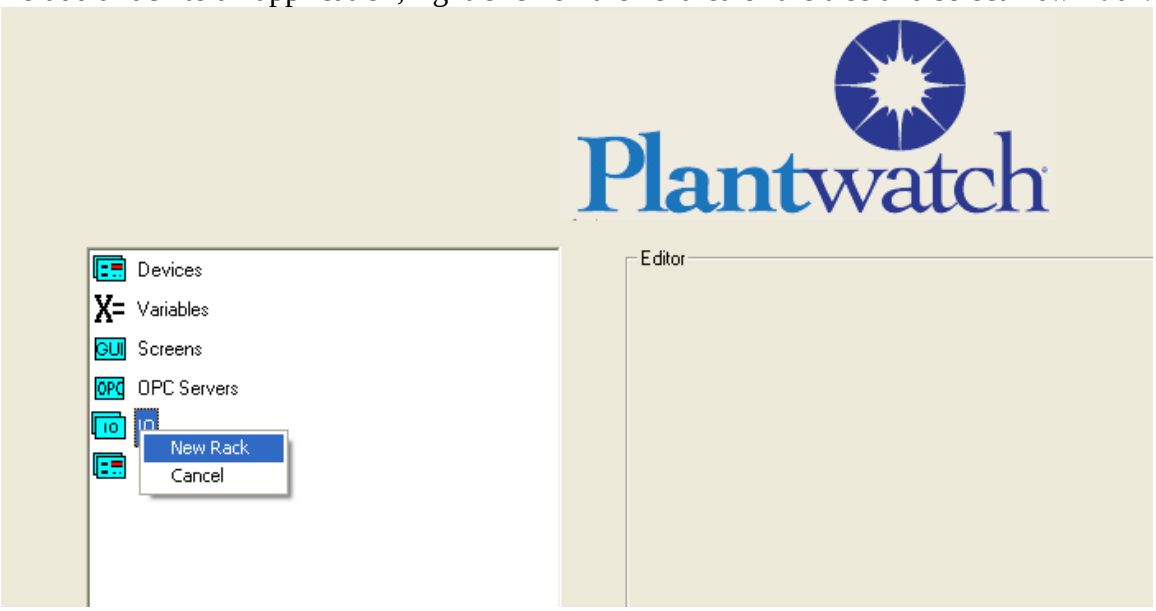
If it is a RF Read the value will be the string read from the RF tag.

Outgoing Data

PlantWatch controls the state of Outputs thru the logic charts. There is a logic cell type called Write to Output. Using the WriteToOutPut cells allows you to set the output to any valid value. See the Logic Chart section for more detail.

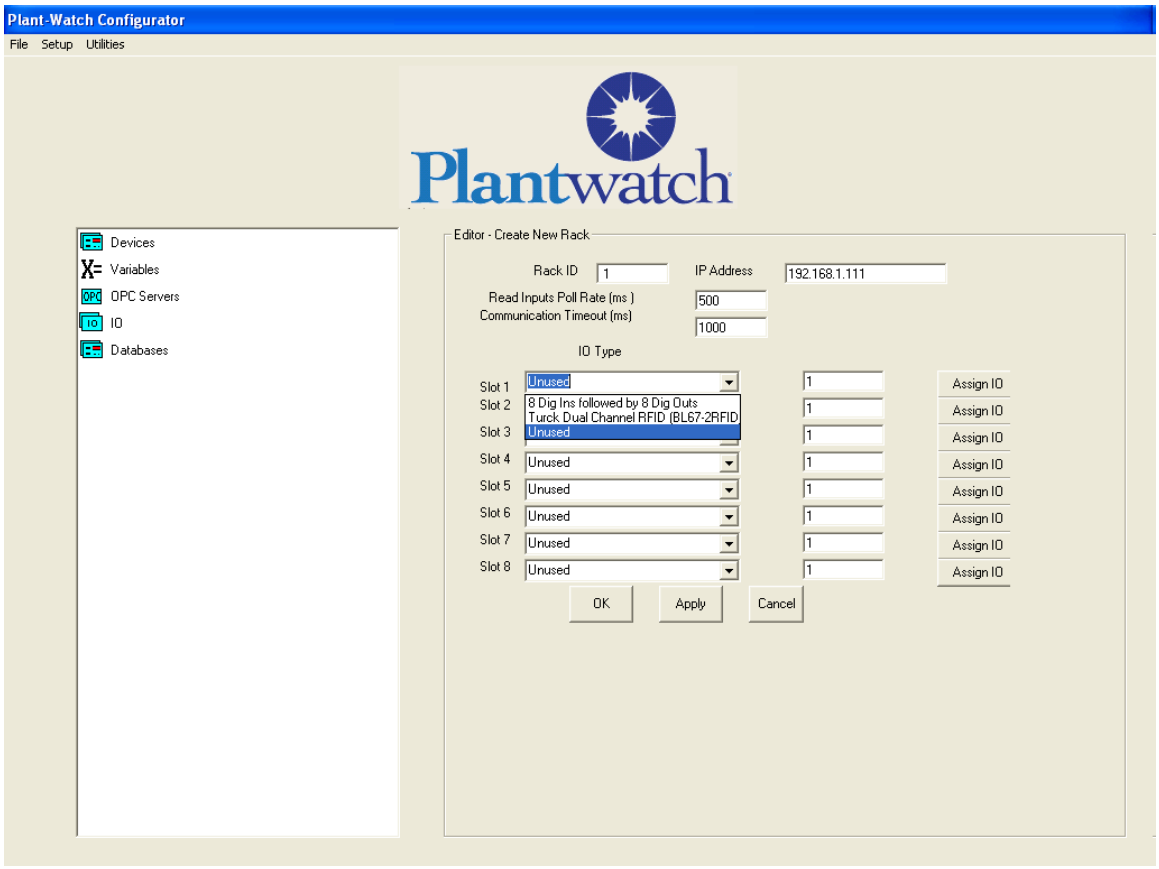
Configuration

To add a rack to an application, right click on the *IO* area of the tree and select *New Rack*.



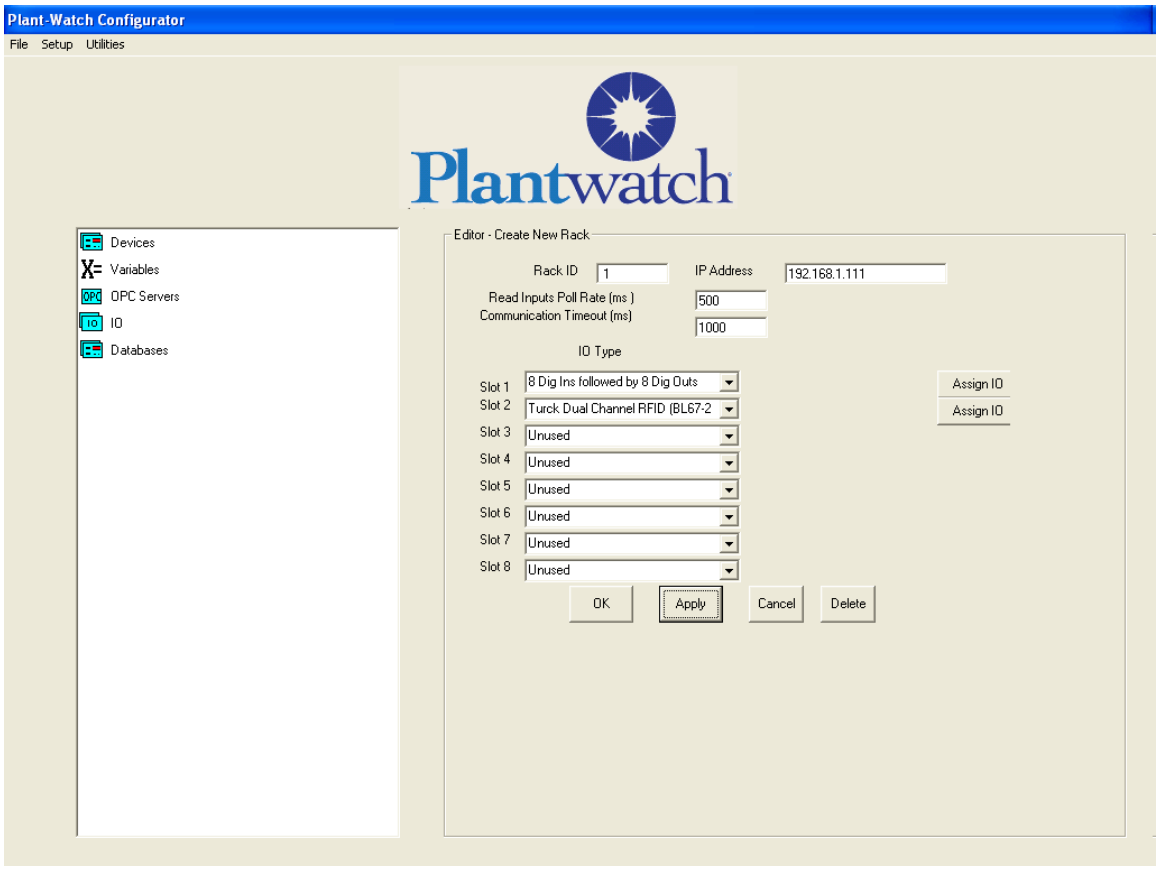
Adding a new rack

You will be presented with the rack configuration dialog. Here you will set the IP Address, Poll Rate and then select the type of card present in each slot.

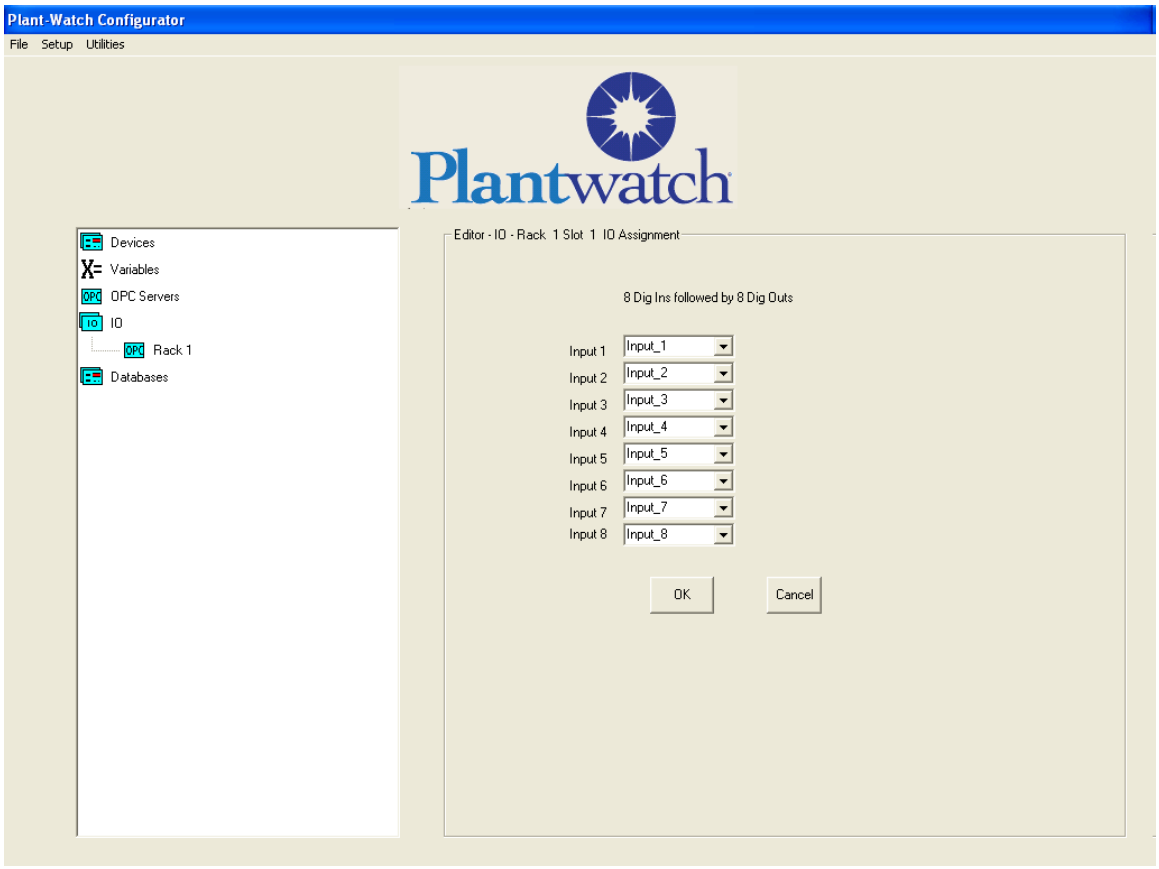


Rack and Slot Configuration

After configuring the *IP Address*, *Poll Rate* and *Slots*, click on *Apply* to update the *AssignIO* buttons.



For each input card there will be associated Assign IO button. Select this button to assign variables to each of the input points. Prior to assigning the variable you must first have created the variables.



Assigned Variables

After a *Variable* is assigned to an input point, its value will always reflect that of the input.

Part Number Setup

Within PlantWatch's logic charts, there are actions specific to traceability

Birth
Collect Data Point
Consume

Each of these cell types will collect data by inserting data into an associated SQL database. To validate that the data is correct, the definition of part numbers and part attributes must be setup in advance of collecting the data. This is done with the *Part Number Setup*

In *Part Number Setup* you will setup the part numbers, Attributes and Parent Child Relationships.

Parts and Sub Parts

Example

To clarify how the Part Number Setup works, we will use an example. For our example we have valid part numbers of

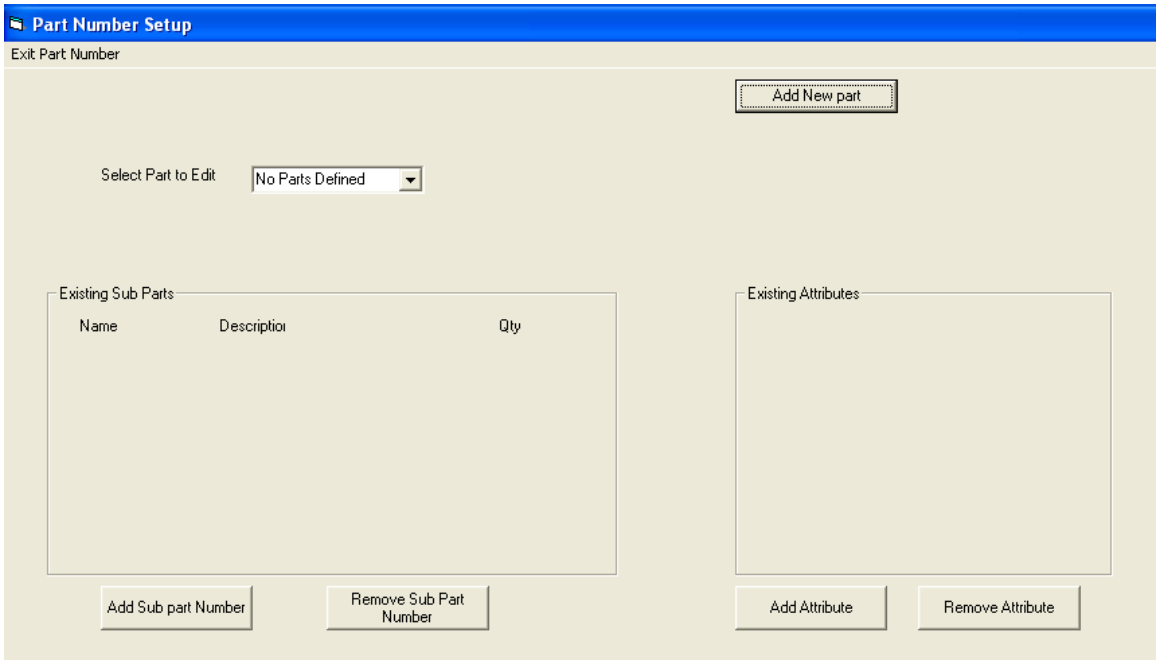
Rim
Tire
Wheel
Car

We will say that a wheel is made of a tire and a rim. Tire and Rim are referred to as *Sub Parts* in PlantWatch. This means that a Wheel can consume a Rim or Tire.

We will also say that a car consumes the wheel.

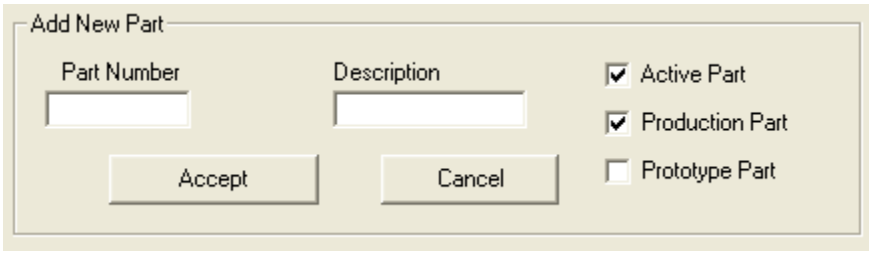
To access the part number setup click on *Part Number Setup* in the *Setup* menu.

To Get to Part Number Setup

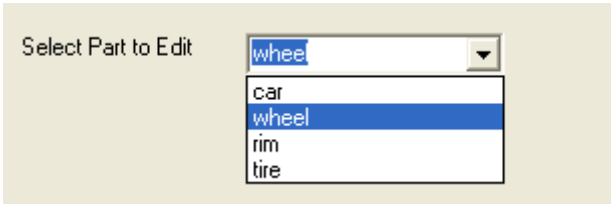


Part Number Setup with no parts defined

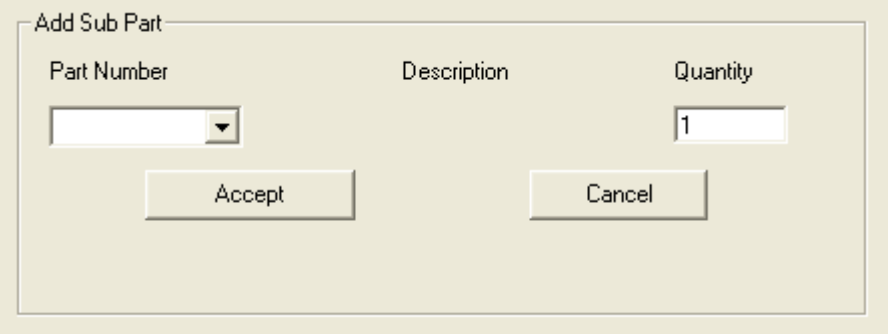
To create the configuration for our example, start off by selecting the *Add New Part* button. This will expose the *Add New Part* dialog



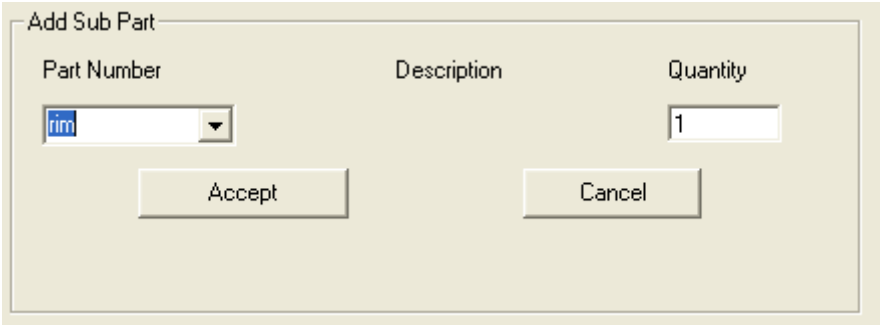
Add each of your part numbers by filling in the *Part Number* field and clicking on the *Accept* button. For our example we will add car, wheel, rim and tire.
After adding the part numbers they become available in the *Select Part to Edit* selection box.



To configure our wheel to be made of a rim and a tire we will first select it in the *Select Part To Edit* list box, and then we will select the *Add Sub Part Number* button.



Using the *Add sub Part* dialog, select rim from the *Part Number* list, and click on the *Accept* button to add it as a sub part.



Using the *Add sub Part* dialog, select *Tire* from the *Part Number* list, and click on the *Accept* button to add it as a sub part.

After adding the rim and tire as sub parts to wheel you can see them listed in the Existing Sub Parts.

Now we must setup the sub parts for the car part number.

To configure our car to be partially made of a wheel we will first select car in the *Select Part To Edit* list box, and then we will select the *Add Sub Part Number* button.

Using the *Add Sub Part* dialog, select wheel from the *Part Number* list, and click on the *Accept* button to add it as a sub part.

Based on our example configuration a car can consume a wheel, and a wheel can consume a tire and rim.

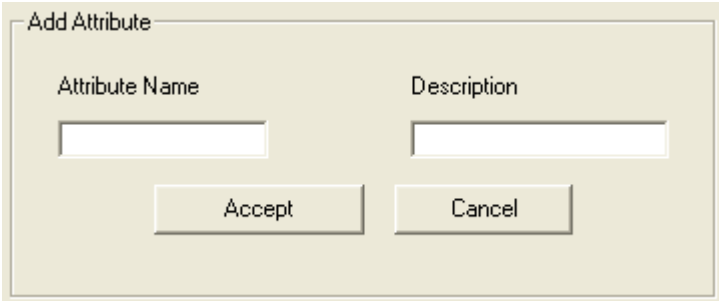
Part Attributes

Even though all parts of a specific design have similar characteristics, each instance of an item, if tested, will have a specific test result. For example the measured diameters of two engine cylinder bores would result in two numeric values. One value for each engine bore. In PlantWatch we call these types of values *Part Attributes.*

Once a Part Attribute is made, it is available to be written to in the SQL database.

For our example of a part called car we will say that it has a part attribute of MaxSpeed. After configuring the part attribute we will be able to collect data and store it as the part attribute of car called MaxSpeed.

To add a part attribute, first select the part number you will to add an attribute to. We will select car. Next we will select the *Add Attribute* button.



Enter in the *Attribute Name* and the click the *Accept Button.*

The Attribute is now created.

Appendix A

Database Browser Notes

Note that you can only have 100 DBBrowser cells total

The maximum number of child records is 256

The child table is evaluated until a row with no entries or a row that just has a condition entry. All Children records past that point are ignored.

During an insert, the tag field is used for the data source

During a delete, a child record with a column without a condition is ignored

During a delete, the tag field is ignored

In condition if you put %text% it is understood that text is the name of a variable. If it does not exist, we should tell the operator before we create a variable.

In *Condition* if you put are comparing to a *String* value you must put the string in single quotes. Example
= 'ABC xyz'

For insert if data type is *String*, numeric length is required.

For *Insert* or *Update*, data longer than field length will be truncated. There will be no padding for data shorter than field length.

Delete does not do any type conversion of the values in the where clause

With insert, you must have a selected column *Type*, all variable values will be inserted as that type.