Chapter 11
Programs and the Visual Programmer

Once you have created your HCA design with devices and groups, schedules, and schedule entries, you may want to take the next step and create programs to help control your home. The Home Control Assistant makes it easy for you to create programs by using two tools that work together: the New Program Wizard, and the Visual Programmer.

HCA programs are very capable. They provide a method of constructing simple sequenced programs: do this, do this, do that, done. The New Program Wizard makes it easy to begin a program, and the Visual Programmer provides a user interface that eliminates the need for a programming language with all its inherent difficulties. HCA also provides the ability to introduce conditionals into programs, thus allowing more complex programs to be created.

This chapter focuses on creating programs, the next chapter focuses on the Program Debugger.

Sections in this chapter are:

- Terminology
- Using the New Program Wizard
- Opening the Visual Programmer
- Areas of the Visual Programmer
  - Tool palette
  - Element list
  - Control buttons
  - Snippet Wizard
- Working with the programming canvas
  - About connecting elements
- Triggers
  - X10 Reception triggers
  - Insteon triggers
  - UPB triggers
  - Generic triggers
  - Magic Module triggers
  - Wireless triggers
  - Global Cache Sensor Triggers
  - Weather triggers
  - Variable change triggers
  - Expression triggers
  - Special condition triggers
  - How are triggers evaluated
- Setting properties for program elements
  - Elements
- Constructing programs
  - Test

  - Repeat
  - The Validate button
- Troubleshooting, or Getting programs to do what you want them to do
- Program properties Advanced tab—Examples

Terminology

Although the Visual Programmer was designed to be usable for non-programmers and does not use a lot of esoteric symbols and punctuation, there are still some terms used that you may want to become familiar with before you start using the Visual Programmer.

Start—When a program is *started* it begins running. Programs can be started in several ways. The first is from the HCA display by using the popup menu from right clicking either a program icon in the display pane or the program's name in the design pane. Programs can be also started when a trigger for it is received.

Run—Between the time a program is started and the time it finishes, it is said to be *running*.

Execute—While a program is running, each element is *executed*, that is, whatever the element is supposed to do, is done. If it is an element that turns on a light, when it is *executed* the light comes on.

Element—Each action that the program executes (does) is an element.

You draw programs by placing *elements* in the programming canvas, and link them together by drawing connecting lines. The program begins with the "Start Here" element and flows from element to element following the connecting lines in the direction of the arrows.

Test—A test is an element that allows the program to examine a condition and execute different elements based upon the outcome of that test.

Trigger—A condition that when it occurs starts a program running. This could be the receipt of a Insteon message, a UPB message, a weather condition, the change in value of a variable, etc

Repeat—An element that allows one or more elements to be executed (done) a number of times.

Variable—A *Variable* can hold values – numbers, strings, dates and times, or simple yes or no.

A program can use variable like pieces of note paper. Each variable has a name and a value, such as Yes or No.

In the way that you might make notes to yourself while doing a complex task, programs can use variables to record things. A program can set a variable to a value and that value remains as long as HCA is running. For example, one program can set a variable to Yes while it runs. Much later another program can test the value of that variable to see if it's Yes or No.

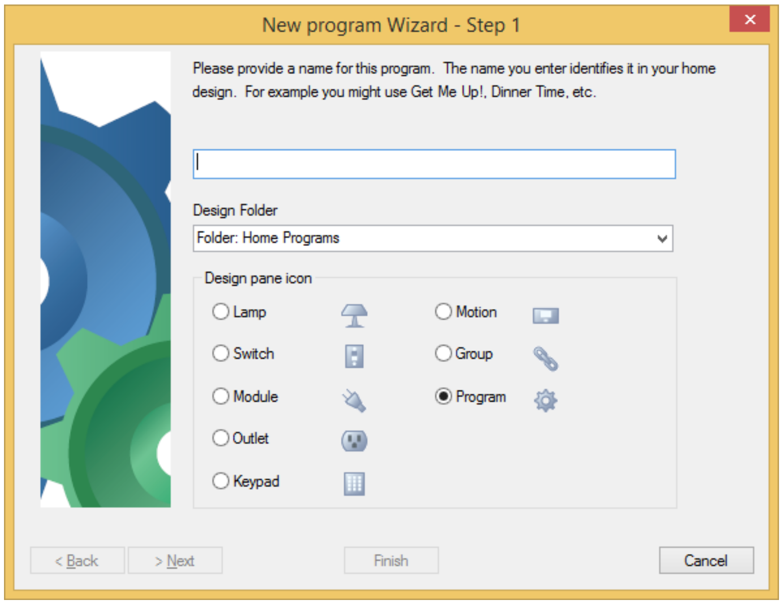
Now that you know the terminology, let's look at HCA programs, what you can do with the Visual Programmer, and how you do it.

Using the New Program Wizard

In the Home Control Assistant, you begin a new HCA program using the New Program Wizard. However, while the wizard creates the program, it doesn't create what the program does. To create the program, after completing the wizard, you use the Visual Programmer.

The first step in creating a new program begins with the New Program Wizard.

In the *Design* ribbon category, click *Program* in the *New* panel. This opens the New Program Wizard. This wizard is very similar to the New Device and New Group wizards. The wizard prompts you through a series of steps as it collects the program name, notes, icon, etc. You can click Back any time to check or change a previous step.


New program Wizard - Step 1

Please provide a name for this program. The name you enter identifies it in your home design. For example you might use Get Me Up!, Dinner Time, etc.

Design Folder: Folder: Home Programs

Design pane icon: Lamp, Switch, Module, Outlet, Keypad, Motion, Group, [✓ Program]

Go through the steps as the wizard prompts you. You will fill in several types of information. Click Next to go each successive step.

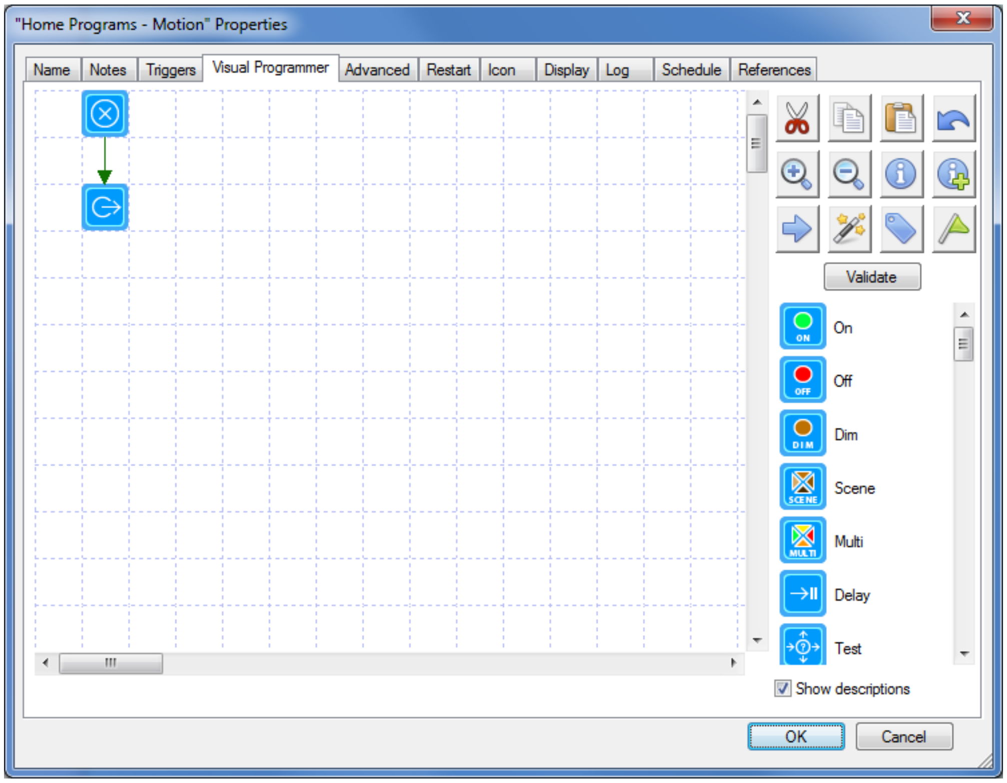
1. Type in a name and type in or select an existing room or folder name. Each program in the folder must have a unique name
2. Add notes if you like. Like devices and groups these notes can appear in a popup window when you move the mouse over a program icon.
3. Choose the icon you want to represent this program.
4. You may go back and check any steps now, or click Finish to complete the definition of the program.

At this point, you need to switch to the Visual Programmer, and complete your program using it.

Opening the Visual Programmer

The Visual Programmer is a tab on the properties dialog box for your program.

1. In the design pane or in the display pane select the program that you just created.
2. Right click and select Properties from the popup menu.
3. On the properties dialog box for your program (it will have the name of the program in the title bar), click the Visual Programmer tab.



There is one very important thing to note about this dialog. Look in the lower right-hand corner. Note the standard Windows mark for a dialog that can be expanded. To make the dialog bigger, left click on that mark and drag the mouse to expand the dialog. Doing this makes it much simpler to see large programs.

Areas of the Visual Programmer

The Visual Programmer tab contains several areas.

- The large area marked in a grid pattern is the *programming canvas*. This displays the programs that you create.
- At the upper right is a *Toolbox*, which provides several actions that you can take with the Visual Programmer.
- Directly below the toolbox is a button used to check the program to make sure it is all defined correctly.
- Below the validate button is a palette showing all the different elements that you can use to construct your program.

This section discusses each of these areas, except the programming canvas, which has its own section, following.

Toolbox

The toolbox contains buttons for several operations that you might use when creating your programs. These operations are listed below (going from left to right, and top to bottom).

Cut—removes the current selection from the programming canvas and places it on the clipboard.

Copy—duplicates the current selection and places it on the clipboard.

Paste—Inserts the clipboard contents onto the programming canvas.

Undo—Reverses your last change.

Zoom in—Makes program elements in the canvas appear larger.

Zoom out—Makes program elements in the canvas appear smaller.

Find—Find a program element based upon selected criterion

Find Next—Find next element that matches the find criterion

Connect program elements—Enters a mode to connect elements.

Snippet Wizard – A wizard the can create useful sequences of elements that can be pasted into your program.

Label elements—Place text next to each element that shows what it does.

Show variable usage—Displays a list of variables used in this program

The first eight of these tools are for activities that should be familiar to Windows users. The other operations – connect, snippet wizard, label, and variable usage are described below.

Element Palette

Below the toolbox is a palette containing all the different elements that you can use to construct the program.

What you see in the list may be different than the screen images presented here. As described in the *HCA Options* chapter you can configure which elements show in this list and in what order they appear.

There are two ways to view the Element Palette. The first method - shown in the screen image above – shows only icons for the elements. If you check the Show Descriptions checkbox below the Element Palette, the elements are show with descriptions.

If you don't use "Show descriptions" you can hover the mouse over an element in the palette and an info tip names the element.

The Element Palette contains many different elements. These are the common elements:

Element Name	What it does
Start Here	This is the element that starts a program running. Each program has only one Start Here element.
Add to log	Add a message to the log.
Auto Off	Configure auto off settings for a device
Camera	Execute an operation on a camera device
Change icon	Change the icon for this program seen on displays
Change schedule	Make a different schedule the current schedule.
Compute	Assign values to variables
Compute Test	Test a variable value
Day Night	Program selected keypads and switches for LED illumination level
Insteon	Read Insteon devices linking tables
Delay	Delay for a while. The time to delay is given in

	hh:mm:ss.
Dim	Control a device or group to Dim.
Email /SMS	Send email or SMS message
Exit	Close, stop, or end the program.
Get Status	Poll one or more devices that support status requests
HTTP	Perform a HTTP network operation
Hue	Perform an option with a HUE device
Make variable No	Make the value of a variable be “No.”
Make variable Yes	Make the value of a variable be “Yes.”
Multi	Control multiple devices in one element
Not variable	Reverse a variable. If its current value is Yes, make it No. If its current value is No, make it Yes.
Off	Control a device, group, or room to Off. Start a program with an Off command.
On	Control a device, group, or room to On. Start a program with an On command.
Ping	Send a message to a network device to see if it replies or not
Play sound	Play a sound file through the computer’s sound system.
Port I/O	Send or received with a serial or IP interface
Read Data	Read data from an external file
Repeat	Repeat a sequence of elements a specified number of times.
Request Input	Display a dialog to request user input
Resume	Remove the suspend condition from a device, program, or group.
Run	Start a Windows program
Scene	Send a command to activate or deactivate a scene
Script	Execute a text based script
Set Mode	Change Home Mode
Show Display	Change the display pane to show a named display
Show message	Show a message in the display pane for a few seconds or in its own window.
Speak	Uses a Text to Speech engine installed on your computer to say a phrase
Start program	Start another program and continue on with the current program when that program finishes.
Status export	Perform a status export
Stop program	Stops a running program
Suspend	Suspend a device, program, or group
Test	Test a condition, and execute different elements based upon the outcome of that test.
Update Tile	Update a tile in a tiled display

Variable Set	Set a variable to a value
Variable Test	Test the value of a variable
Wait Until	Wait until a time in the future. Time given as hh:mm or as sunset or sunrise.
Element Connect	Provides a way to join two elements without drawing a line between them.

These elements are for specific hardware that you may or may not have.

Element Name	**What it does**
Send IR	Sends a sequence of IR commands
Send X10	Sends X10 commands
Send ZWave	Sends ZWave commands
Thermostat	Controls a thermostat device
Thermostat test	Tests data retrieved from a thermostat
UPB Link	Send a UPB Link command
UPB Blink	Causes a UPB device to blink at a specified rate by sending the Blink command to the device
Weather Test	Tests data retrieved from a weather provider
X10 All lights on	Send an X10 All Lights On command to a house code.
X10 All lights off	Send an X10 All Lights Off command to a house code.
X10 All units off	Send an X10 All Units On command to a house code.

There are other elements besides those shown above for specific hardware that has been designated Legacy Hardware. To see these elements, select HCA – Properties from the menu and look on the Legacy tab.

Hint: It is suggested that you use the Visual Programmer tab of HCA Properties to configure the list of elements you see in the Element Palette. It makes no sense to show elements for hardware you don’t have or elements you plan never to use.

Validate button

Also, in this area of the dialog is the Validate button. This performs several checks on your program making sure that all element properties have been set and all elements are connected correctly.

Working with the programming canvas

Before discussing the details of the programming elements, let’s look at ways you can use the programming canvas to add, delete, select, and set the properties of elements. You can use the programming canvas for any of the following functions:

- To select an element or more than one element
- To add a new element or delete an element
- To move an element
- To change the properties of an element
- To cut or copy an element to the clipboard
- To paste an element from the clipboard
- To connect two elements together
- See what an element does
- Add a note or comment about an element

Following are procedures for accomplishing each of these tasks.

To select an element:

1. Place the mouse over the element and click.

   You can use the same method to select the arrow lines connecting elements.

To select more than one element:

1. Place the mouse over the first element and click.
2. Hold down the Ctrl key and move the mouse over another element and click.

Or

1. Left mouse button down someplace on the canvas – not on an element - and drag. Let up the mouse button to complete the rectangle. This creates a selection rectangle. Any element fully contained in the rectangle will be selected.

You can use either of these methods to select as many elements as you need.

To add a new element:

1. From the element list, select the element you want to add to the canvas and drag in on to the canvas and drop it where you want.

When you add a new element, the properties dialog for that element type automatically displays.

To delete an element:

1. Select an element or several elements – using either of the methods described above - and press the Delete key on the keyboard or right click and select Delete from the popup menu.

   Whatever you have selected is deleted. You can also delete connecting lines in this way by selecting them and pressing Delete.

To move an element:

1. Select the element or elements to be moved.
2. Click on one of the selected elements and drag the element(s) to the new location.
   Whatever you have selected: element, elements, or connecting lines, are moved. Connecting lines attached to the moved element(s) are stretched to maintain the connections.

To change the properties of an element:

1. Right click on the element and select Properties from the popup menu.

Or

1. Double click on the element.

The popup menu shows the *properties* choice only if the element has properties.

To cut or copy an element to the clipboard:

1. Select the element(s)
2. Use the tools palette Cut/Copy buttons.

or

1. Right click on one of the elements in the selection and choose Cut/Copy from the popup menu.

Hint: The operations to delete, move, cut, and copy all work whether you have one or several elements selected. The action that you choose happens to each element you have selected.

To paste an element from the clipboard:

1. Click the paste button from the tools palette. The element is pasted into the canvas on the first unused row - starting from the top and moving down.

or

2. Right click when over a blank cell on the programming canvas and select paste from the popup menu. This will paste the clipboard contents at the location.

If you have more than one element on the clipboard (that is, you selected multiple elements and then cut or copied), all the elements are pasted into the canvas. Any connecting lines between elements that were present when they were placed on the clipboard are retained when pasted.

To connect elements together:

There are two methods of connecting elements.

<u>Method 1</u>:

This method for connecting elements requires a steady hand with the mouse but is worth learning since it can be quick once you get used to it.

1. Select the *From* element.
2. Carefully move the mouse pointer to the edge of the element. If you get the pointer in just the right place, it changes to the small circle with a dot.
3. Left click and drag a connecting line to the *To* element. Release the mouse when over the target element.

<u>Method 2</u>:

This method is useful for hooking up a whole series of elements.

1. Click the line tool in the toolbox and the cursor changes to the "circle with dot".
2. Now click on an element (let's call it A) you want as the source for the line. Just click. Don't click and drag!
3. Now click on the element you want to have the line end in (let's call this B). The line is added from A to B. Note that the cursor is still the "circle with dot".
4. Now click on another element (called C). A new line is created from the B to C.
5. And you can keep going, connecting C to D, D to E, etc.
6. To end, just click anywhere not on an element.

It’s important to remember to click someplace on the canvas to end this style of connect. Until you do, you will not be able to, for example, press the Validate button, OK, button, change to another tab, etc.

Hint: There is another way to connect two elements without a direct line between them. Refer to the About Connecting Elements section.

See what an element does:

Once you have added an element to the canvas and set its properties, you can quickly see what it does.

1. Move the mouse pointer over the element.
   At the lower left corner of the display, directly below the programming canvas, you will see a description of what the element does when executed. If you have not yet set the properties for the element, the description reads *Element not ready*.

Or

1. Press the Label Elements tool in the tools palette. This places text next to each element that shows in a very brief form what the element does.

Hint: It is a good idea to leave some space between elements so the label text doesn't overlap other elements.

Add a note or comment about a program element:

Use the right mouse button when positioned over an element and select *Comment* from the popup menu. Enter any notes you have about this element into the dialog box that appears. If an element has any notes, these display in a popup window when the mouse is moved over the element.

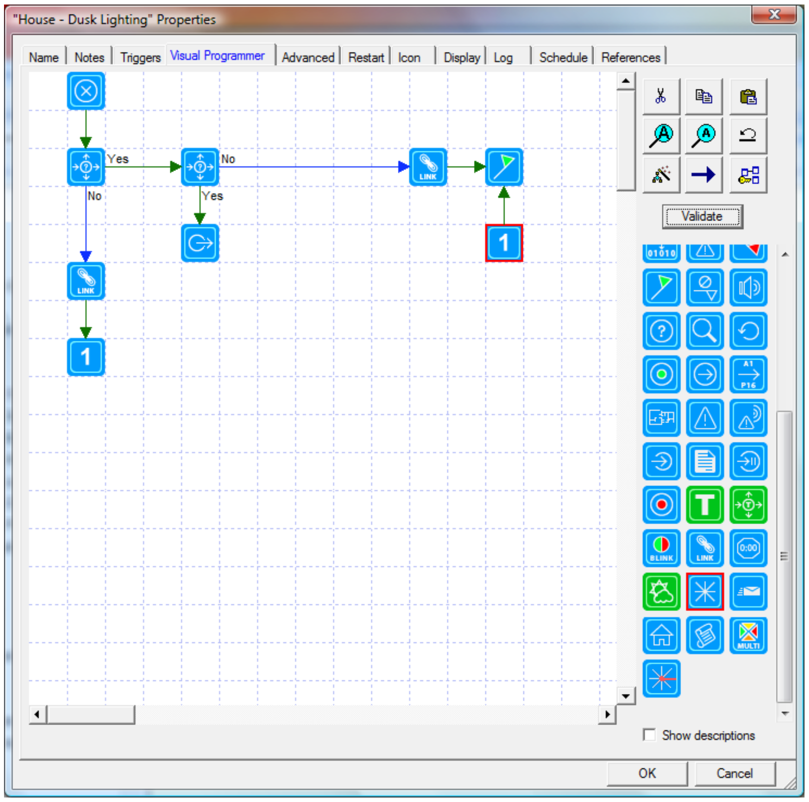
About connecting elements

You control the sequence in which your program executes its elements by connecting the elements. The program begins running with the *Start Her*e element, and continues from element to element by following the connecting lines in the direction of the arrows. If you have two elements with a connecting line between them, the *From* element is where the line starts, the *to* element is where it ends. The *To* element has the connecting line arrowhead pointing at it.

If you try to draw a connection that the Visual Programmer does not support, the connecting line will not appear when you complete the drag. Here are the connections that Visual Programmer considers invalid, and that it does not support:

- You can't connect to the "Start Here" element from any element. It is the beginning—no element can be in sequence before it.
- You can't connect to any other element from an Exit element. It's the end—no element can come after it in sequence.
- You can't have more than one connecting line from any element to another, except for the test and repeat elements.

If you create complex programs you may find that you have lines crossing elements and that can be hard to read. One way to solve this is to use the connector elements. These act to join two elements together without a direct line. For example:



In this example when execution completes the link element then it continues to the make variable yes element.

When you add a connector element you set the connector number from one to nine. In this way you can use connector elements to connect different parts of your program.

Snippet Wizard

The snippet wizard is started from the tool palette button, the one to the left of the arrow button. What it does is to create sequences of elements and places them on the clipboard. Once the wizard is done, you can paste these into your program where they are needed.

There are three different snippets that the wizard creates:

- Control lights over time. A sequence of elements to increase or decrease the illumination level of a device or a given time.
- Establish Scene. This captures the current illumination level of a set of devices you select. Elements are created to send commands to set those lights to those same illumination levels.
- Test for start conditions. Constructs a sequence of test elements to test for all the possible triggers for this program

Depending upon the snippet, the wizard may ask a series of questions or not. Once complete the snippet is on the clipboard. Select the cell where it should go, right click and select Paste from the popup menu.

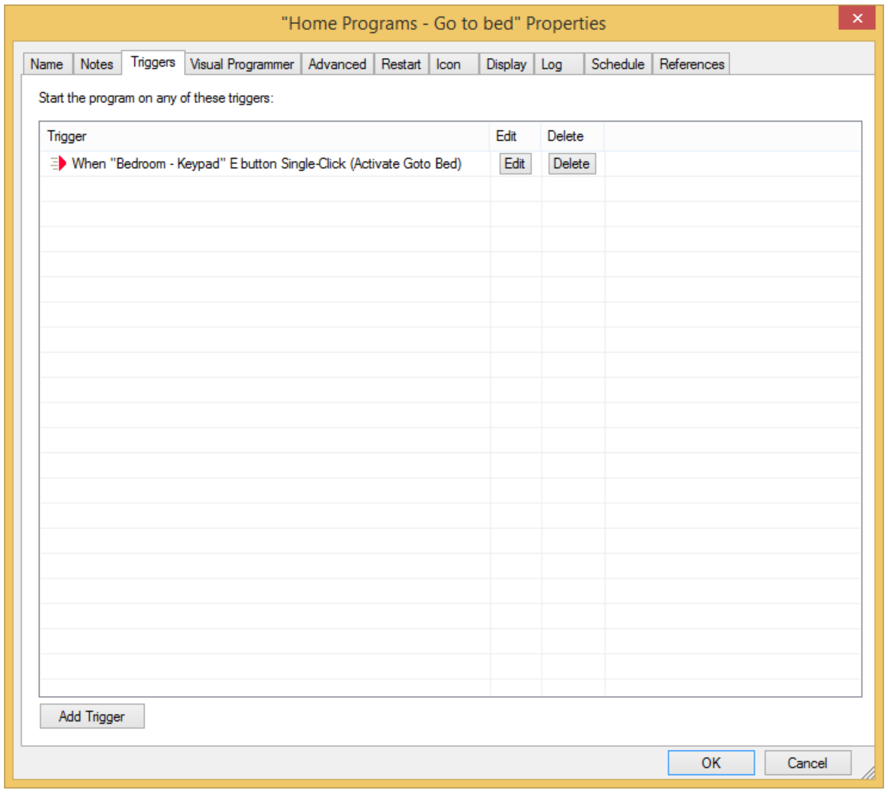
Program Triggers

Before discussing what programs can do, let's first examine how program can be started. Programs, unlike schedule entries, start when something happens in your home and usually not at the same time each day. There are several possible kinds of triggers that start programs. These are:

- Expression
- Variable Value change
- Generic trigger
- Global Cache Sensor reception
- Home mode change
- Insteon reception
- Port reception
- Special condition
- State change
- UPB Action
- UPB Powerline message
- Weather condition
- Wireless Component Message
- X10 reception
- Zwave reception

There may be other trigger types if legacy hardware support is enabled.

When you open the program properties and select the Triggers tab this page appears:


"Home Programs - Go to bed" Properties

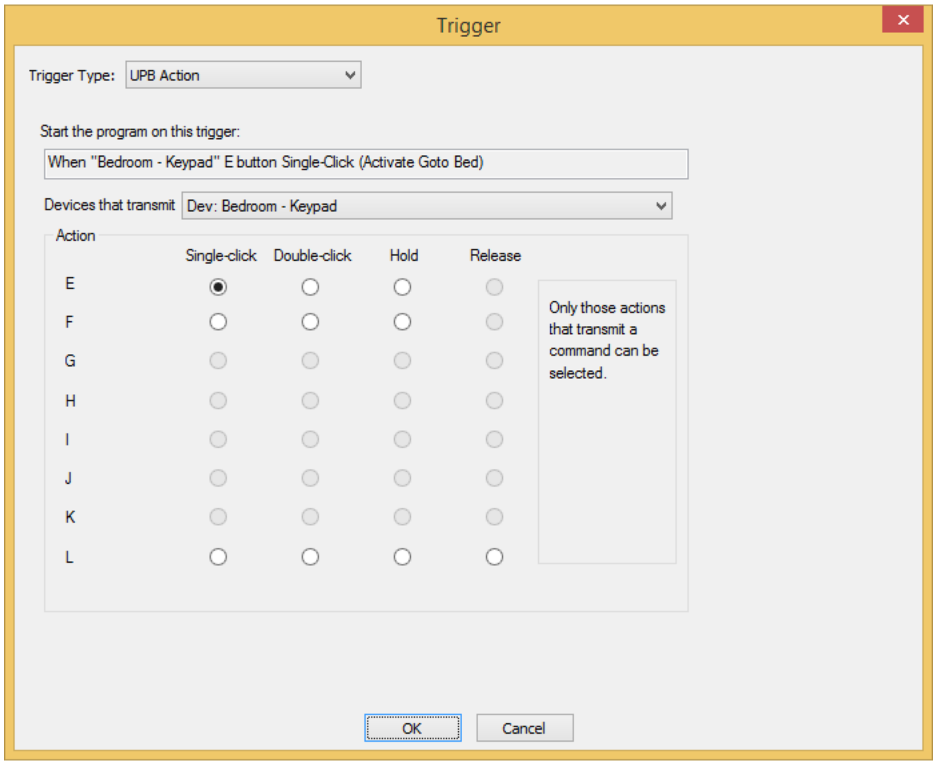
Start the program on any of these triggers:

Trigger	Edit	Delete
When "Bedroom - Keypad" E button Single-Click (Activate Goto Bed)	Edit	Delete

In the big list at the center of the tab is a list of all the triggers that can start the program. In this dialog you can perform three actions:

- To delete a trigger, press the Delete button on that row.
- To modify an existing trigger, double-click it or press the Edit button on that row.
- To add a new trigger, double-click on an empty row or press the *Add Trigger* button.

When adding a new trigger, a dialog opens where the configuration of the trigger is set. Each trigger type has configuration appropriate to its type. The example below shows a UPB Action trigger:



Trigger Type: UPB Action

Start the program on this trigger:

When "Bedroom - Keypad" E button Single-Click (Activate Goto Bed)

Devices that transmit Dev: Bedroom - Keypad

Action

	Single-click	Double-click	Hold	Release
E	●	○	○	○
F	○	○	○	○
G	○	○	○	○
H	○	○	○	○
I	○	○	○	○
J	○	○	○	○
K	○	○	○	○
L	○	○	○	○

Only those actions that transmit a command can be selected.

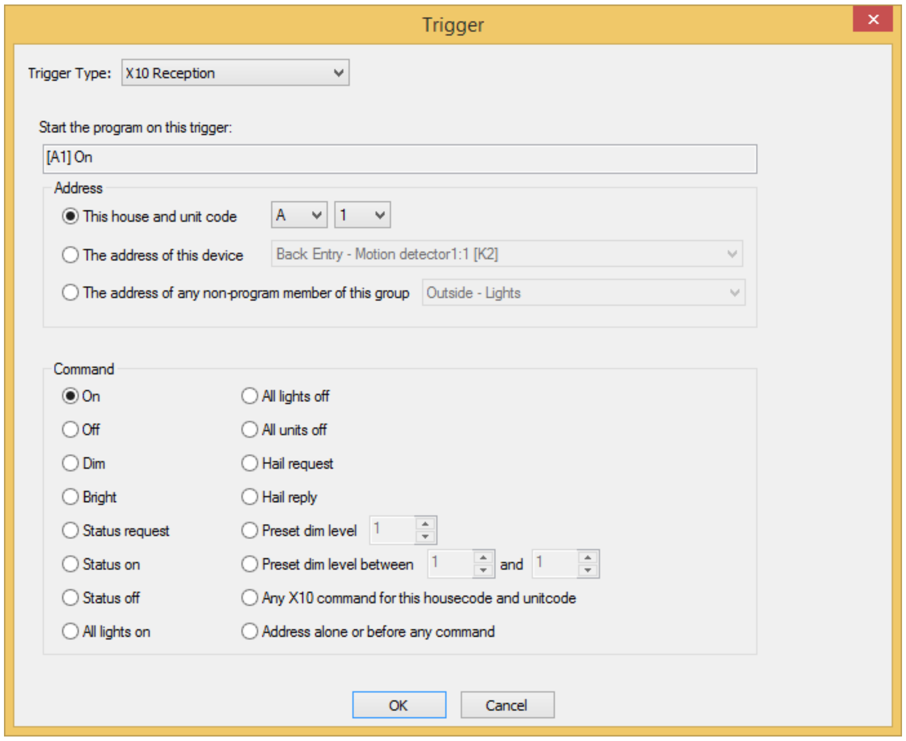
At the top of the dialog is a dropdown where you can select the type of trigger you want to add. In this example, a UPB Action trigger is being added.

Each trigger type has different properties and is shown in the next sections.

Hint: The Snippet Wizard can construct a series of Test elements to see what trigger started the program.

X10 Reception Trigger

This is the Add Trigger dialog when adding an X10 trigger.



Trigger

Trigger Type: X10 Reception

Start the program on this trigger:

[A1] On

Address

- [✓ This house and unit code] A 1
- The address of this device: Back Entry - Motion detector1:1 [K2]
- The address of any non-program member of this group: Outside - Lights

Command

[✓ On]	All lights off
Off	All units off
Dim	Hail request
Bright	Hail reply
Status request	Preset dim level 1
Status on	Preset dim level between 1 and 1
Status off	Any X10 command for this housecode and unitcode
All lights on	Address alone or before any command

OK Cancel

In the Address portion of the dialog is specified the house and unit code. This can be done in four possible ways:

- Select the house and unit code
- Select the house and unit code assigned to a device. If the device has multiple units, for example a multi-button keypad, select which unit of the device to use. The individual units are shown as the name of the device followed by a colon then a number.
- Select the house and unit code assigned to another device.
- Select a group

The first case is simplest. All you need do is to pick a house and unit code.

The next two options have a special behavior. If the X10 address of the device is changed, then this trigger for the program also changes.

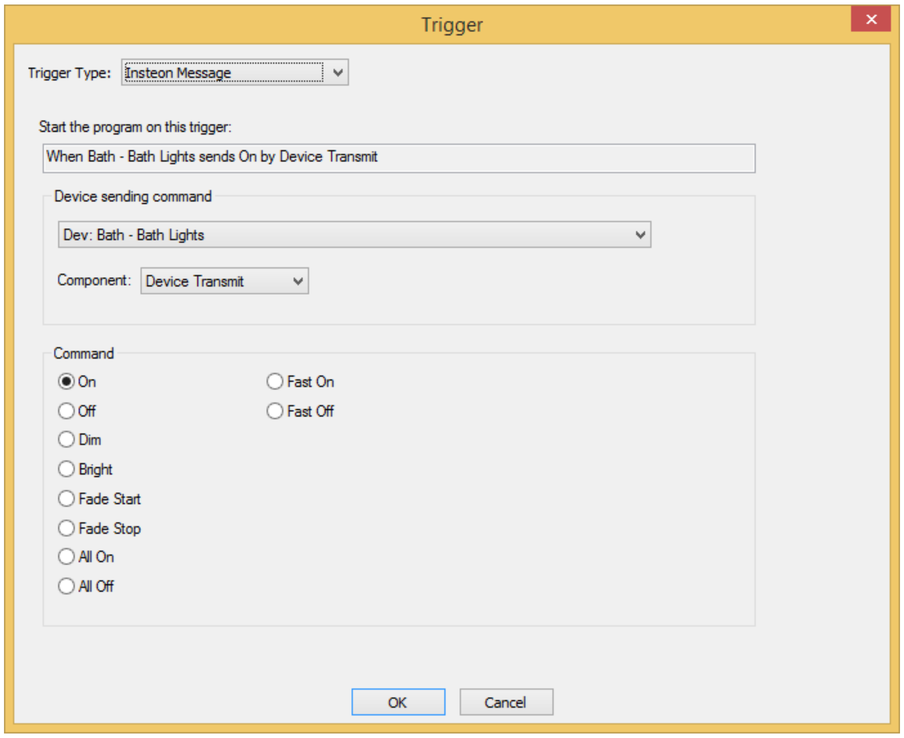
For example, assume you created a trigger and selected the "Use the address of this device" option. The device selected was "Keypad Entry:2" and "Keypad Entry" has a X10 address of L7. This would assign to this program a trigger of whose address is L8. If the keypad was later changed to D2, the trigger address automatically changes to D3.

The last case is a lot like assigning a trigger address using a device. In the case of selecting a group, the address of any of the device members of the group is the address of the trigger. For example, assume you have a group called Lamps. Also assume that in this group are three devices whose addresses are C1, D1, and E9. If you create a trigger and chose the group address option and select group Lamps, then the address part of the trigger are the addresses C1, D1, and E9. If you change the address of any of the group members, or add or remove members from the group, the addresses for this trigger changes also.

In the lower section of this dialog you specify the command that must follow the address to trigger the program. Listed here are all the X10 commands. You can also choose a specific preset dim level or a range of levels that trigger the program. The final option of the command list is, in effect, no command. If you choose this option the program triggers when the address selected is received regardless of what command, if any, follows.

Insteon Reception Trigger

This is the Add Trigger dialog when adding an Insteon trigger.



Trigger Type: Insteon Message

Start the program on this trigger:

When Bath - Bath Lights sends On by Device Transmit

Dev: Bath - Bath Lights

Component: Device Transmit

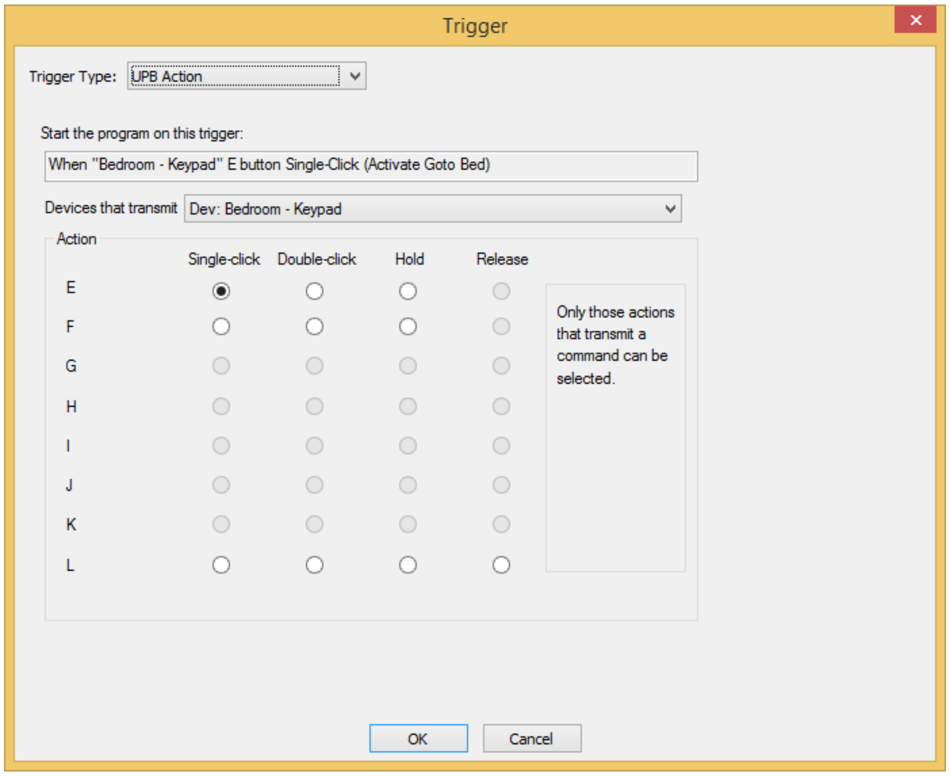
Command: [✓ On] Off Dim Bright Fade Start Fade Stop All On All Off Fast On Fast Off

This type of trigger is used to respond to an Insteon powerline message.

Insteon power line message triggers are described in the Insteon Appendix.

UPB Action Trigger

This is the New Trigger dialog when adding an UPB Action trigger.



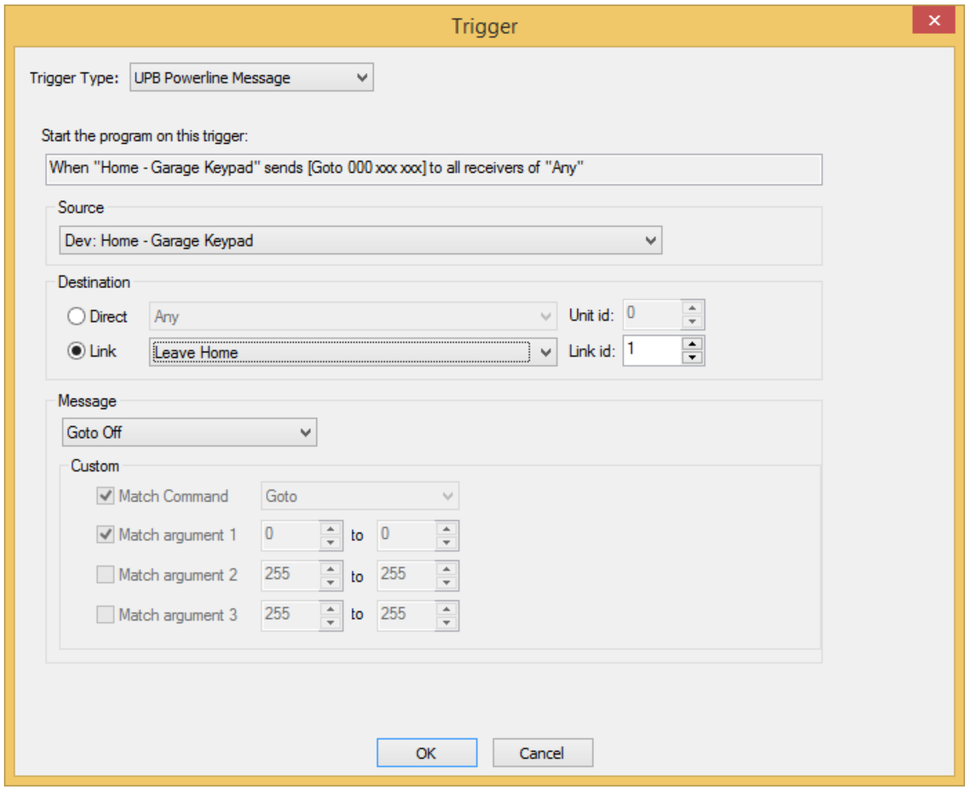
Trigger
Trigger Type: UPB Action
Start the program on this trigger:
When "Bedroom - Keypad" E button Single-Click (Activate Goto Bed)
Devices that transmit: Dev: Bedroom - Keypad
Action
Single-click Double-click Hold Release
E F G H I J K L
Only those actions that transmit a command can be selected.
OK
Cancel

This type of trigger is used to respond to an action taken at a UPB transmitter – a keypad, input module, or switch rocker.

UPB Action triggers are described in the *UPB* Appendix.

UPB Powerline Message Trigger

This is the New Trigger dialog when adding a UPB power line message trigger:



Trigger Type: UPB Powerline Message

Start the program on this trigger:

When "Home - Garage Keypad" sends [Goto 000 xxx xxx] to all receivers of "Any"

Source: Dev: Home - Garage Keypad

Destination

○ Direct: Any — Unit id: 0

● Link: Leave Home — Link id: 1

Message: Goto Off

Custom

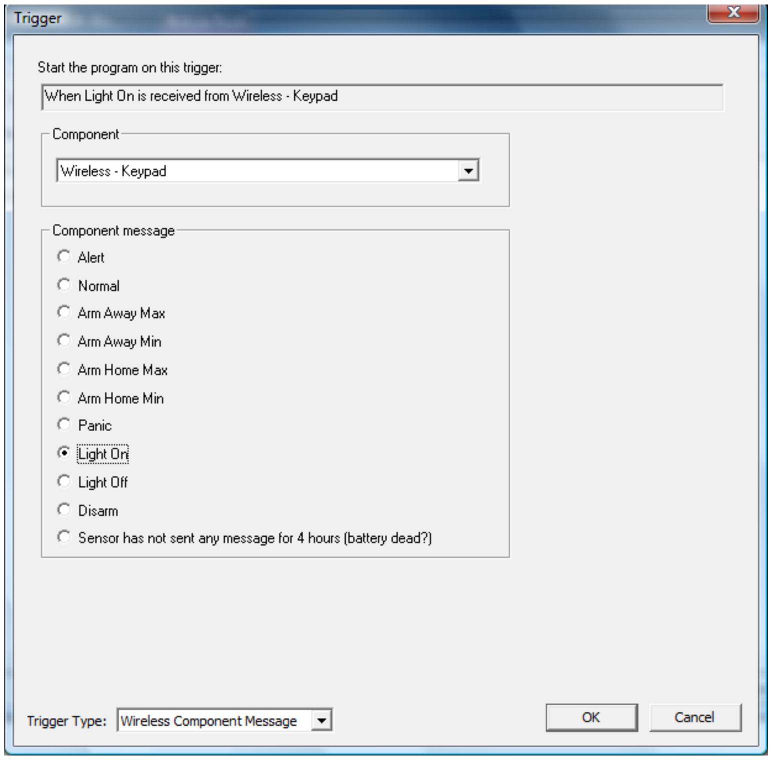
✓ Match Command	Goto		
✓ Match argument 1	0	to	0
Match argument 2	255	to	255
Match argument 3	255	to	255

This type of trigger is used to respond to any UPB power line message.

UPB power line message triggers are described in the UPB Appendix.

Wireless Component Message Trigger

This is the New Trigger dialog when adding a Wireless Component trigger:



Start the program on this trigger:
When Light On is received from Wireless - Keypad

Component: Wireless - Keypad

Component message:
- ( ) Alert
- ( ) Normal
- ( ) Arm Away Max
- ( ) Arm Away Min
- ( ) Arm Home Max
- ( ) Arm Home Min
- ( ) Panic
- [✓ Light On]
- ( ) Light Off
- ( ) Disarm
- ( ) Sensor has not sent any message for 4 hours (battery dead?)

Trigger Type: Wireless Component Message

In conjunction with a Wireless interface, messages from wireless door/window open/close sensors, non-X10 motion sensors, and security keypads, can be received and processed by HCA.

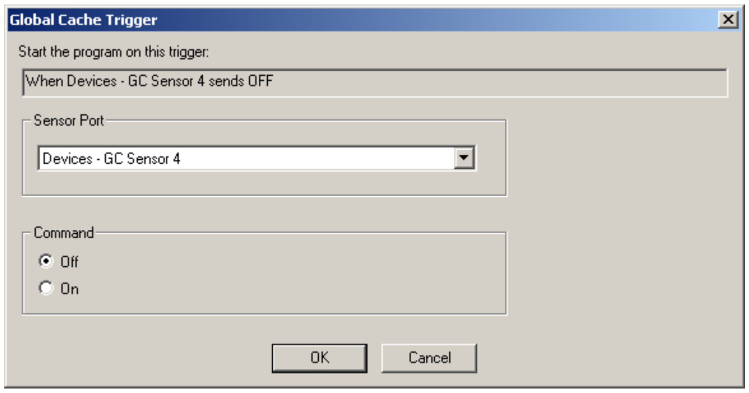
Once you have added wireless components to the design, programs can start when message are received from them.

Wireless Component Message triggers are fully described in the appendix on *Wireless Interfaces*.

Global Cache Triggers

A Global Cache trigger is used to start a program when a Global Cache port configured as a *Sensor Notify* detects a change.

This trigger type dialog appears as:



Start the program on this trigger:
When Devices - GC Sensor 4 sends OFF

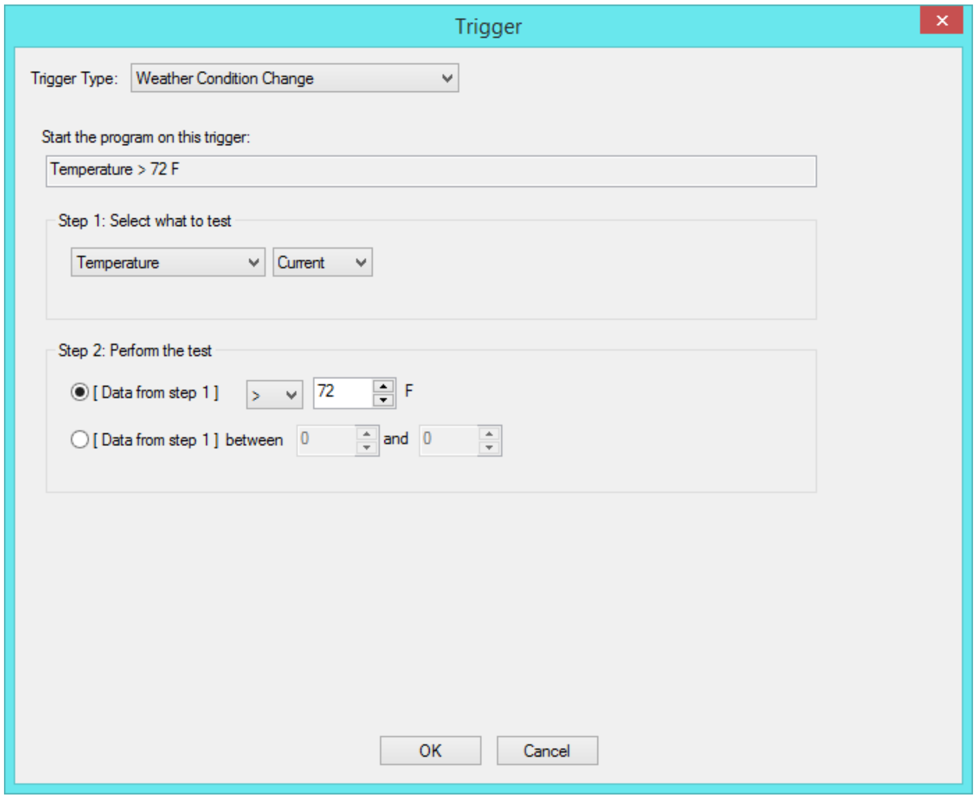
Sensor Port: Devices - GC Sensor 4

Command: [✓ Off] On

Global Cache triggers are described in the *IR* appendix.

Weather Condition Triggers

The Weather Condition trigger dialog appears as:



Trigger Type: Weather Condition Change

Start the program on this trigger:
Temperature > 72 F

Step 1: Select what to test
Temperature Current

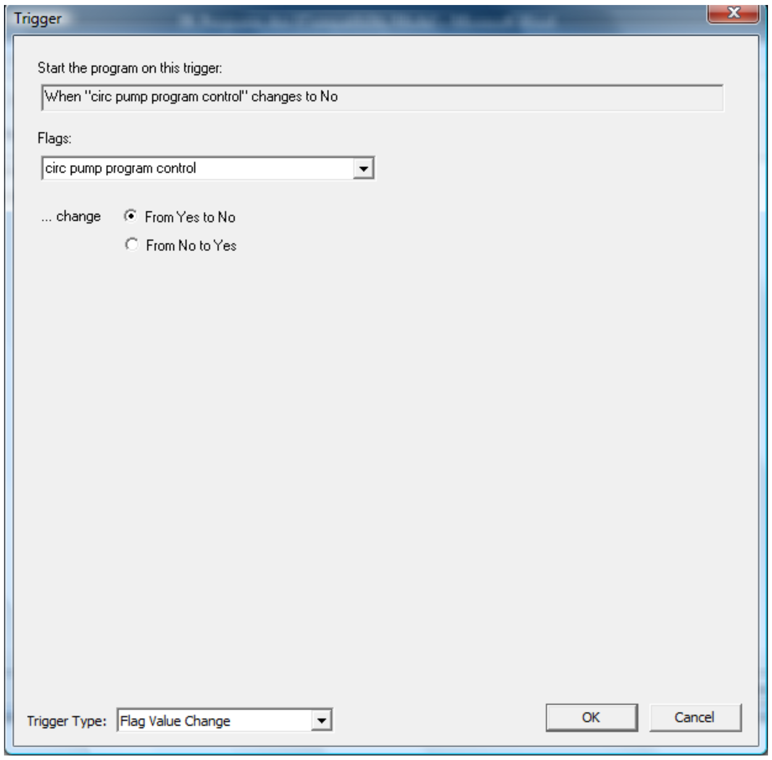
Step 2: Perform the test
[✓ [Data from step 1] > 72 F]
[Data from step 1] between 0 and 0

The Weather Condition trigger provides a means for you to create programs that start when data from a weather provider passes some test. For example, you can create a program that starts when the outside temperature goes over 80 degrees.

Weather Condition triggers are fully described in the appendix on Weather Providers.

Variable Value Change Triggers

The Variable Value Change trigger dialog appears as:



When "circ pump program control" changes to No

Flags: circ pump program control

... change (● From Yes to No) ○ From No to Yes

Trigger Type: Flag Value Change

Variables can hold Yes and No values and can be created with the Make Variable Yes, Make Variable No, and the Compute elements. This type of trigger doesn't test what the variable value is, just what it changes to. For example, suppose you have these three programs:

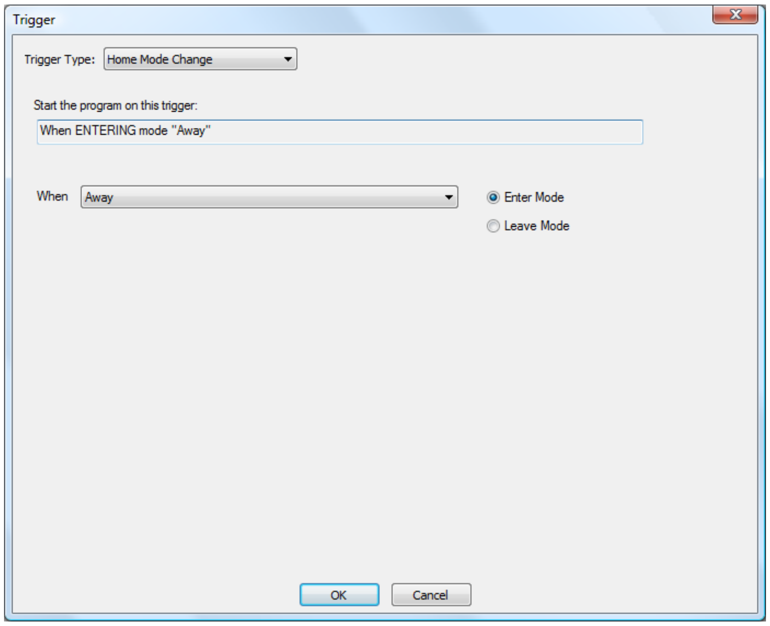
- Program A. Sets Variable "It's Dark" to No.
- Program B. Sets Variable "It's Dark" to Yes.
- Program C. Has a trigger that says: When "It's dark" changes to No

When program A runs nothing happens. When program B runs, the change of the Variable to NO causes Program C to start.

Why wouldn't program B just start program C directly instead of relying on the variable change trigger? Well it could but it just may be more convenient to do it this way.

Home Mode Triggers

This is the Add Trigger dialog when adding a home mode change trigger:



Start the program on this trigger:
When ENTERING mode "Away"

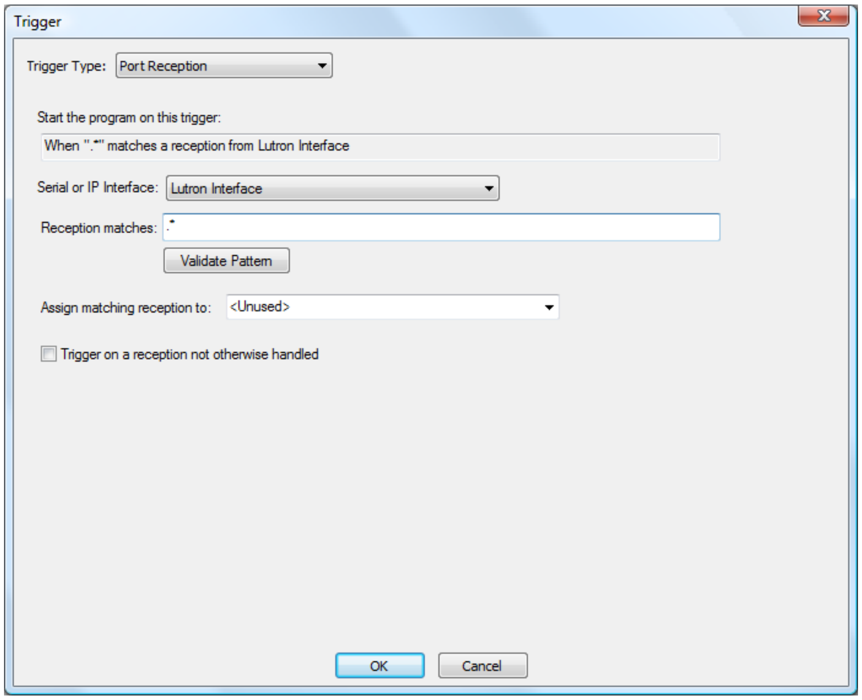
When: Away

[✓ Enter Mode] Leave Mode

This type of trigger happens when the current home mode changes. Any mechanism that resulted in the mode change, a program changed the mode or it was changed by user action, can cause programs with this type of trigger to start.

Port Reception Trigger

This is the Add Trigger dialog when adding a port reception trigger:



Trigger Type: Port Reception

Start the program on this trigger:
When ".*" matches a reception from Lutron Interface

Serial or IP Interface: Lutron Interface

Reception matches: .*

Assign matching reception to: <Unused>

☐ Trigger on a reception not otherwise handled

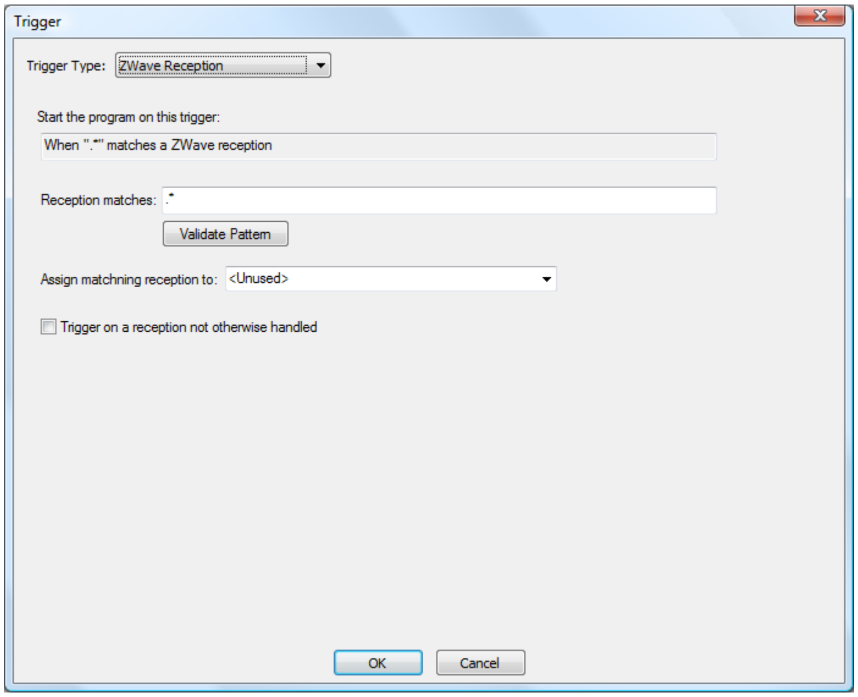
This type of trigger happens when a reception from a Generic Serial Interface or Generic IP interface occurs.

If the regular expression in the trigger matches the reception then the program starts. A program can also be designated as the program that starts if no other program triggers on the reception. This could be because no other trigger on any program matches the reception or because no other program has any triggers for this interface.

If the trigger does match then the reception message is optionally assigned to the variable entered or selected in the dropdown.

ZWave Reception Trigger

This is the Add Trigger dialog when adding a zwave reception trigger:



Trigger Type: ZWave Reception

Start the program on this trigger:

When ".*" matches a ZWave reception

Reception matches: .*

Assign matchning reception to: <Unused>

☐ Trigger on a reception not otherwise handled

This type of trigger happens when a reception from a ZWave interface occurs.

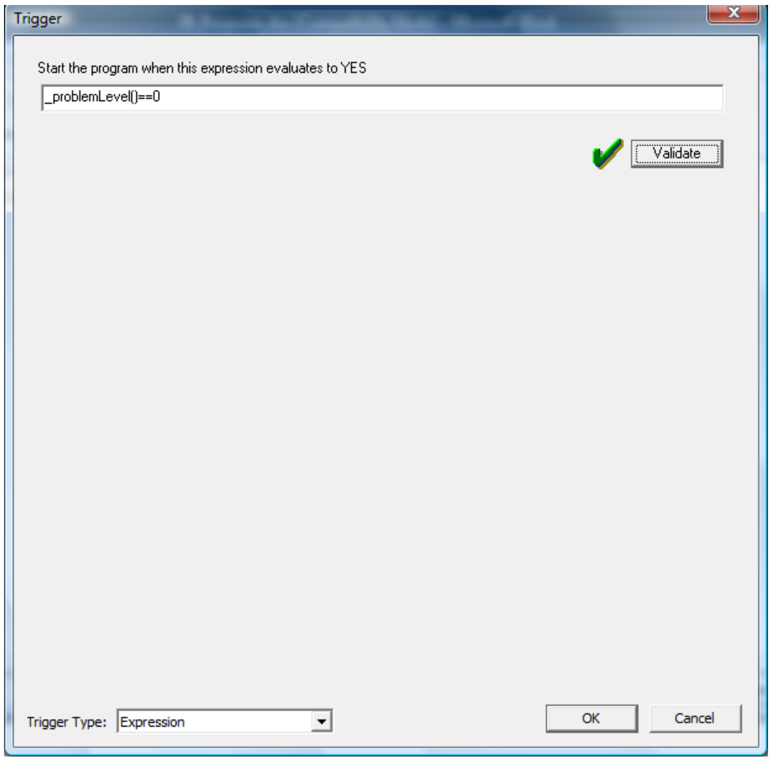
It is configured similarly as the Port Reception trigger except that you don't special an interface since it uses the Zwave interface and there can be at most one of those in use.

If the regular expression in the trigger matches the reception then the program starts. A program can also be designated as the program that starts if no other program triggers on the reception. This could be because no other trigger on any program matches the reception or because no other program has any triggers for this interface.

If the trigger does match then the reception message is optionally assigned to the variable entered or selected in the dropdown.

Expression Triggers

This is the Add Trigger dialog when adding an expression trigger:



Start the program when this expression evaluates to YES

_problemLevel()==0

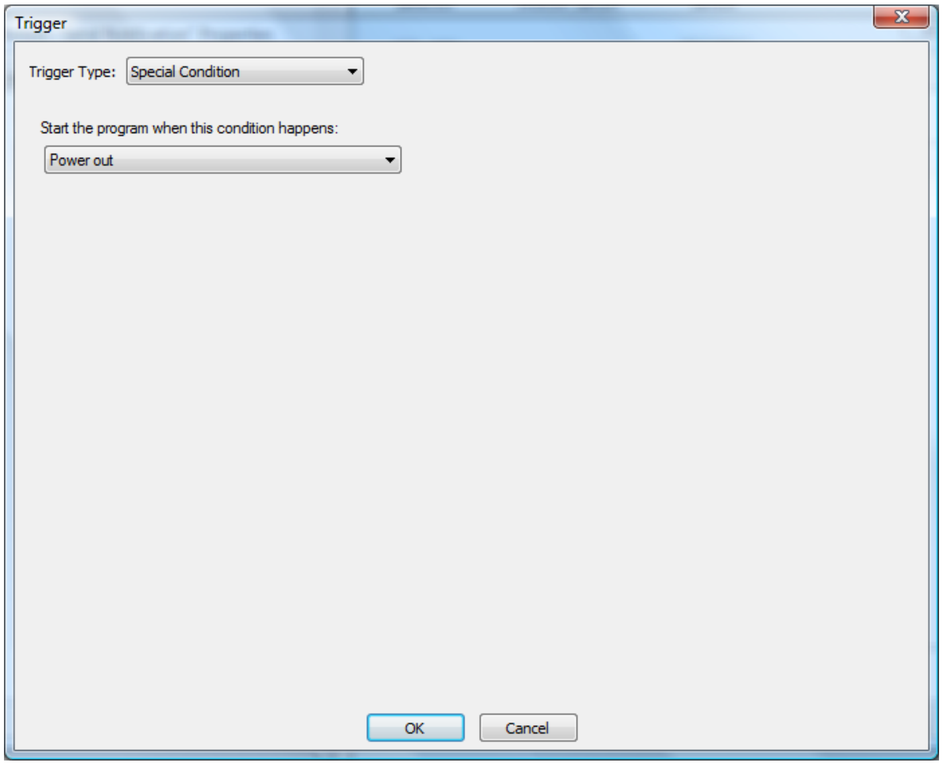
Trigger Type: Expression

The expression trigger is the most general type of trigger. What you enter into this dialog uses the same expression language used in the ComputeTest element.

Whenever this expression evaluates to Yes, the program starts.

Special Condition Trigger

The Special Condition trigger dialog appears as:



Trigger Type: Special Condition

Start the program when this condition happens:

Power out

In this dialog you create trigger for special conditions that occur. These are:

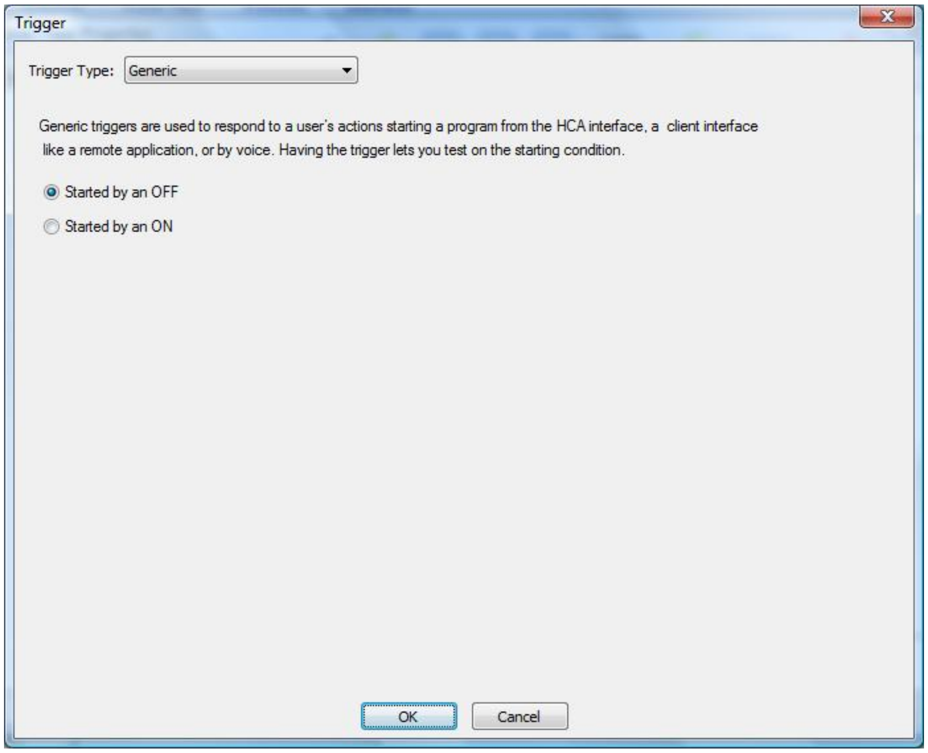
- HCA started normally
- HCA started from a power failure
- Power Out
- Power restored
- Client connection made

The first two conditions give you the means to have a program run when HCA first starts. The second two conditions are only useful when your computer is on a UPS backup system. When HCA detects that the house power is off, and programs with a "Power Out" trigger start. When Power returns, any programs with a "Power restored" trigger start.

Hint: The Power Out and Power Restored triggers are for legacy designs. The better method to handle these conditions is using the Alert mechanism of the Troubleshooter. See chapter 11 for information on the Alert Manager.

Generic Trigger

The Generic trigger dialog appears as:



Generic triggers are used to respond to a user's actions starting a program from the HCA interface, a client interface like a remote application, or by voice. Having the trigger lets you test on the starting condition.

[✓ Started by an OFF]

Started by an ON

You may have wondered why when you right-click on a program in the development UI you sometimes get a "Start" menu pick and sometimes "On" and "Off" menu picks. It has to do with if the program has any triggers defined and if they are "On" or "Off" type triggers. If it doesn't have "On" or "Off" type triggers then the popup menu has "Start". Same reason for a client when in the control page you see "Start" or "On" and "Off".

Also, the question comes up of what does an ON and OFF mean for a program. Both start the program – OFF doesn't mean stop. In the case of "On" it starts the program running with an "On" trigger and the same for "Off" – it starts the program with an OFF trigger. The key fact is that you can then test in the program for "Started by ON" or "Started by OFF" and do different things.

Testing for "Started by ON" and "Started by OFF" is very useful for when a program is started from a client.

Generic triggers provide the ability to create triggers that help you get the action you want when a program is started from the user interface of a client.

Trigger Evaluation

There are a few important additional points to consider when using Weather triggers, Variable triggers, and Expression triggers.

The properties of the trigger define when the program starts. For example, the trigger "When the outside temperature is over 80" starts the program when the weather provider tells HCA that the outside temperature has risen to 80. But when does the program start again? The temperature may stay above 80 for several hours. Does the program start each time a weather observation is made?

No. The program only starts the first time the temperature is reported over 80. The temperature must drop back below 80 and then rise above 80 before the program starts again.

Think of the trigger being in two possible states: *Ready to trigger* and *triggered*. If the temperature is under 80, then the trigger is *ready*. Once it goes over 80 the program starts because the trigger has *triggered*. It can't trigger again until it first goes back to the *Ready* state.

This is the same logic used by the variable triggers and expression trigger. The expression must evaluate to NO to be *ready*, then evaluate to YES to *trigger* and then back to NO to become *Ready* again.

The last important point is that HCA evaluates expression and weather triggers about every 60 seconds. This means that the expression trigger is only looked at once a minute. If the condition the expression is based upon changes more rapidly than once a minute, HCA may miss some of those changes.

Parameterized programs

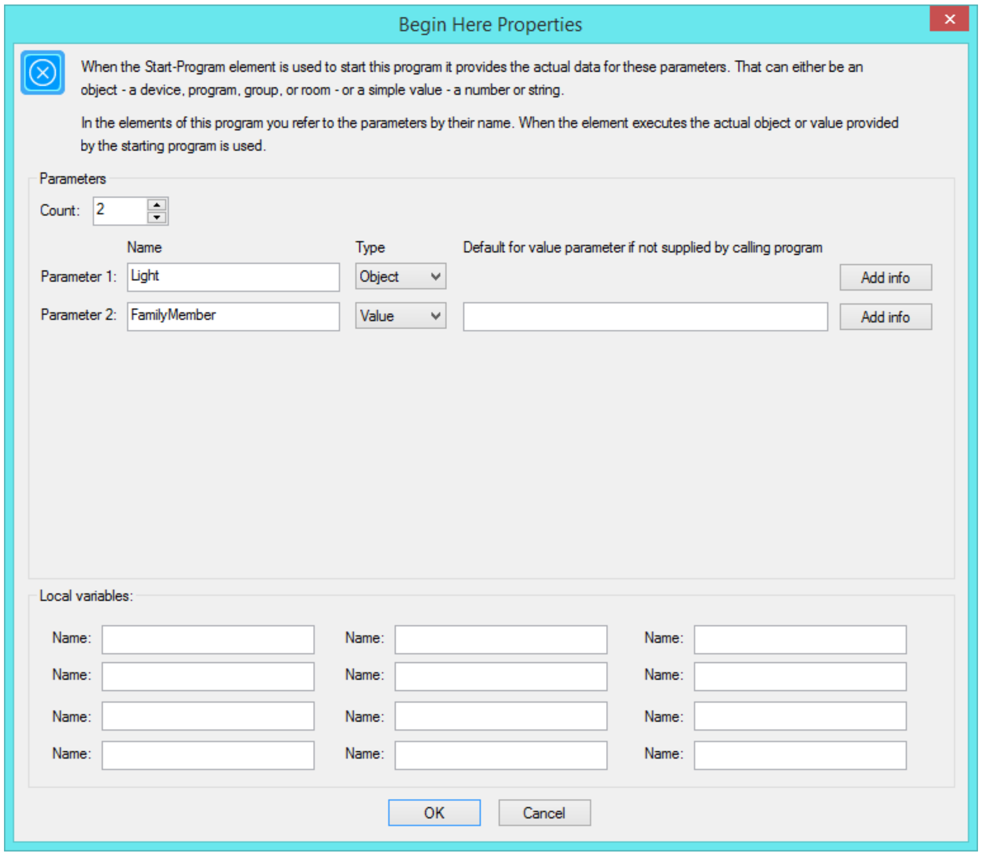
In previous versions of HCA one program could start another program but, unlike a more traditional programming language, there was no way to pass "parameters" to the started program.

This has changed in HCA 13 with the ability for a program to accept arguments – objects and values – from other programs.

Suppose Program A wants to start program B and tell B what device to work on and also pass another piece of data that is a string.

In traditional programming language, Program B is a "subroutine" of A.

In HCA 13, after program B is created, on the "Advanced Options" tab you must first tick the "This program supports and/or local variables" box. One that is done that then the properties of the Begin-Here element can be opened.


Begin Here Properties

When the Start-Program element is used to start this program it provides the actual data for these parameters. That can either be an object - a device, program, group, or room - or a simple value - a number or string.

In the elements of this program you refer to the parameters by their name. When the element executes the actual object or value provided by the starting program is used.

Parameters

Count: 2

	Name	Type	Default for value parameter if not supplied by calling program	
Parameter 1:	Light	Object		Add info
Parameter 2:	FamilyMember	Value		Add info

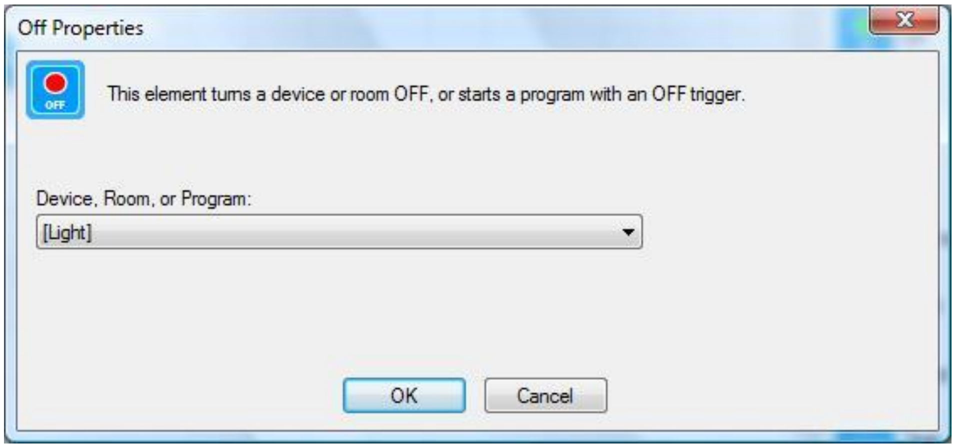
Local variables:

In the Begin-Here properties you select the number of parameters and given them names and what they are used for. Parameters can either be an object – a device, program, group, room - or a value – a string numbers, date-time, etc.

For value parameters you can also enter a default value that is used if the caller of the program doesn't provide an argument to that parameter.

Also, you can provide some text that describes the use or the parameter. Press the "Add Info" button to do that. The button is labeled as "Edit Info" if you already have added that text. This text will appear when this program is selected for use in the Start-Program element.

Once the parameters are defined then they can be used. For example, here is an ON element in program B:


Off Properties

This element turns a device or room OFF, or starts a program with an OFF trigger.

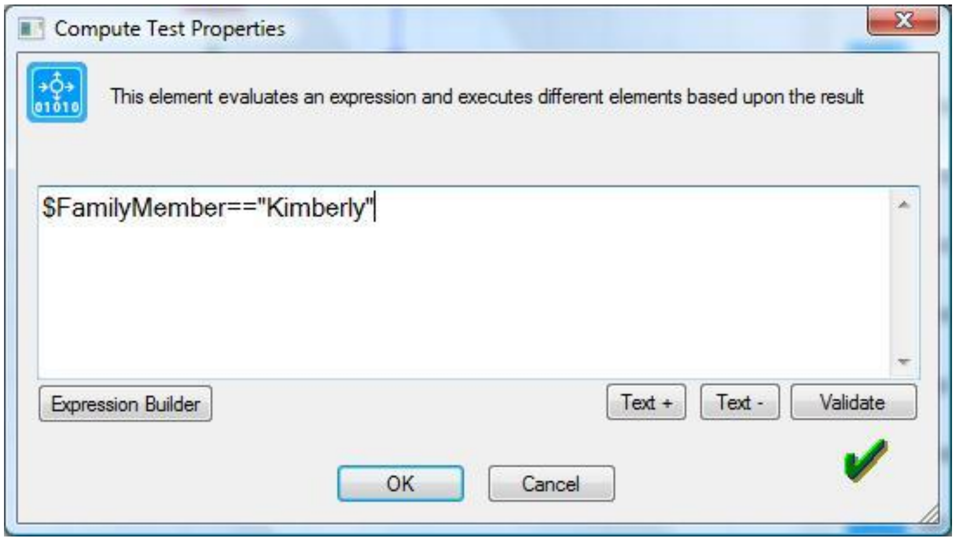
Device, Room, or Program:
[Light]

Note how the selection, in addition to all the usual "on-able" things are the parameter names. It added to the dropdown those parameters specified in the Begin-Here element that are used for objects. In this example, in the drop down you will not see "[FamilyMember]" since it is a parameter for a value.

The elements where you can select an object parameter are the ones you would expect

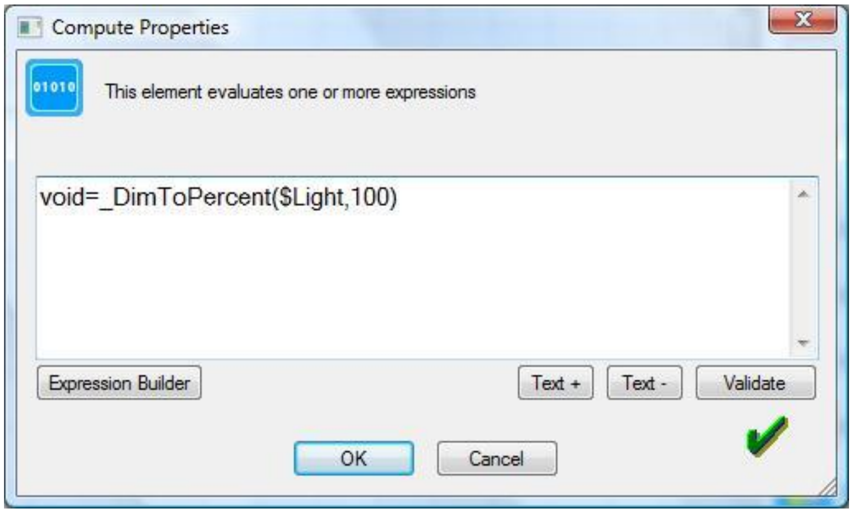
- On
- Off
- Dim
- Multi
- Thermostat
- Test (IsOn, IsOff, IsDim, IsSuspended)
- Thermostat-Test
- UPB-Blink
- Suspend
- Resume
- Stop
- ShowDisplay
- Auto-Off

In a compute or compute-test element you can also use the value parameters as well as the object parameters. Here are two elements from this program. The first is seeing which Family Member is being targeted and the second controls the light to a level.



This element evaluates an expression and executes different elements based upon the result

```
$FamilyMember=="Kimberly"
```



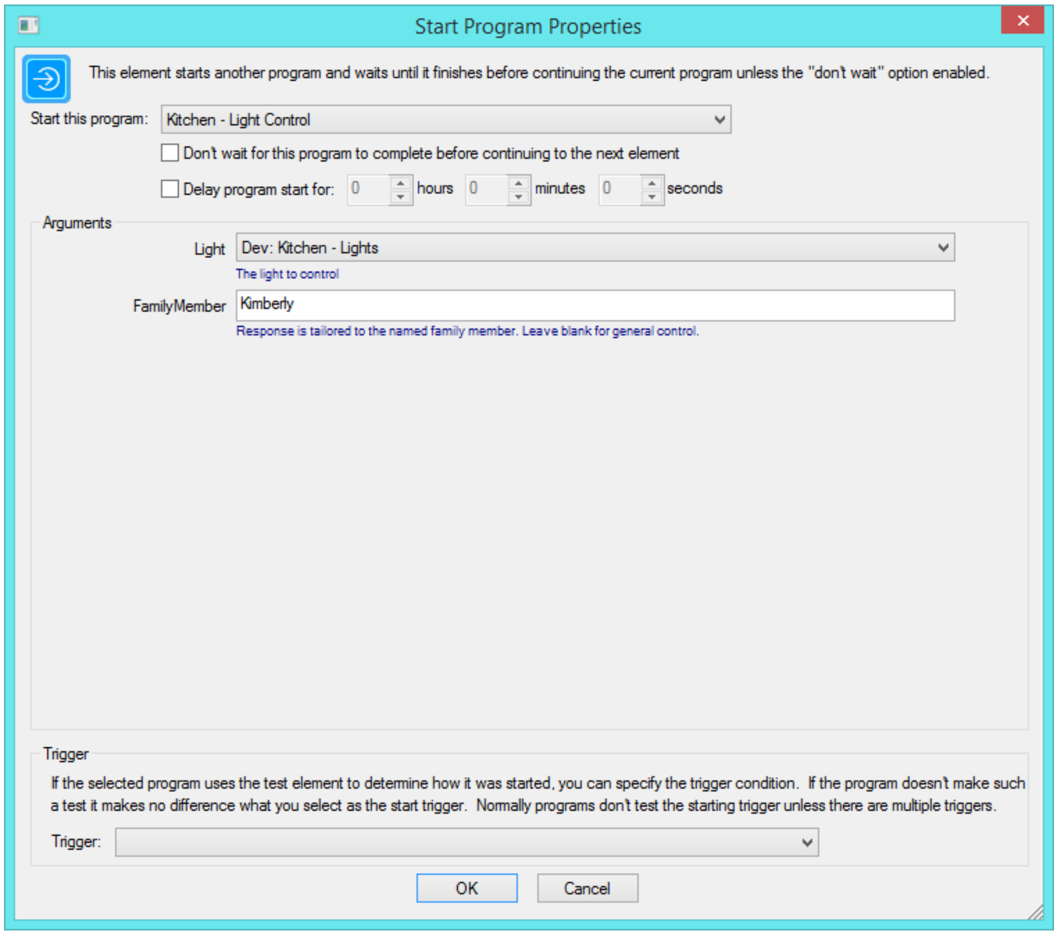
This element evaluates one or more expressions

```
void=_DimToPercent($Light,100)
```

Note that you refer to program parameters by their name prefixed with a $. This lets the parser disambiguate variable names from parameter names.

Now let's turn to the calling program. Here is the Start-Program element in program A.

Note the text in blue in the image below. This is the parameter info text that was added in the Begin Here element of the program where the parameters: are defined.



This element starts another program and waits until it finishes before continuing the current program unless the "don't wait" option enabled.

Start this program: Kitchen - Light Control

☐ Don't wait for this program to complete before continuing to the next element

☐ Delay program start for: 0 hours 0 minutes 0 seconds

Arguments

Light: Dev: Kitchen - Lights
The light to control

FamilyMember: Kimberly
Response is tailored to the named family member. Leave blank for general control.

Trigger

If the selected program uses the test element to determine how it was started, you can specify the trigger condition. If the program doesn't make such a test it makes no difference what you select as the start trigger. Normally programs don't test the starting trigger unless there are multiple triggers.

Trigger:

Since the Begin-Here element of the started program has defined what kind of argument it expects – object or value – it presents the UI you would expect. Object parameters get a dropdown listing all the devices, programs, groups, etc. Value parameters get a simple edit control.

Note: The text entered in the edit control for a value parameter may contain expressions. Those expressions are embedded in %'s like other HCA elements that take expressions embedded in text. For example, if there was variable named "count" containing the value 23, then if this was entered as a value parameter argument:

Count is %count + 1%

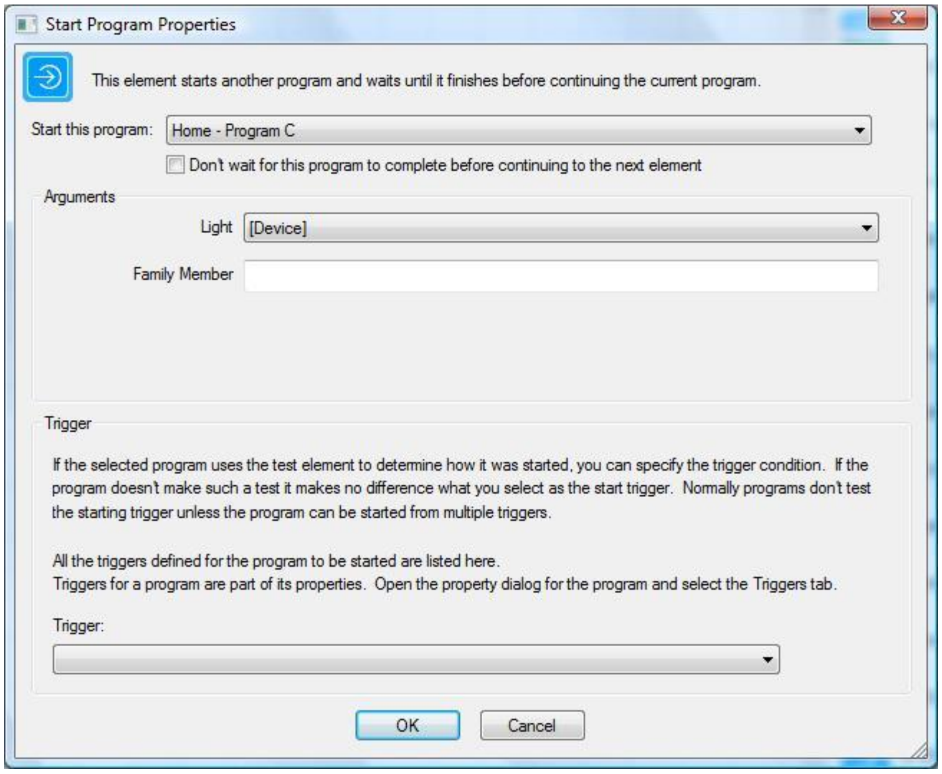
The started program would receive a parameter of text "Count is 24"

Some "Advanced Level" Info for the most sophisticated users

Suppose you want program A to Start-Program on program B and program B to Start-Program on program C. Also suppose that when program B starts program C you want to pass the value of one of its (that is, Program B) parameters to program C.

As an example, program B has a parameter called "Device". Program C has a parameter called "Light". In the start-program element in program A starting program B you select "Home-Lamp" for "Device".

Now in program B in the Start-Program element starting program C you can do this:


Start Program Properties

This element starts another program and waits until it finishes before continuing the current program.

Start this program: Home - Program C

Don't wait for this program to complete before continuing to the next element

Arguments

Light [Device]

Family Member

Trigger

If the selected program uses the test element to determine how it was started, you can specify the trigger condition. If the program doesn't make such a test it makes no difference what you select as the start trigger. Normally programs don't test the starting trigger unless the program can be started from multiple triggers.

All the triggers defined for the program to be started are listed here.
Triggers for a program are part of its properties. Open the property dialog for the program and select the Triggers tab.

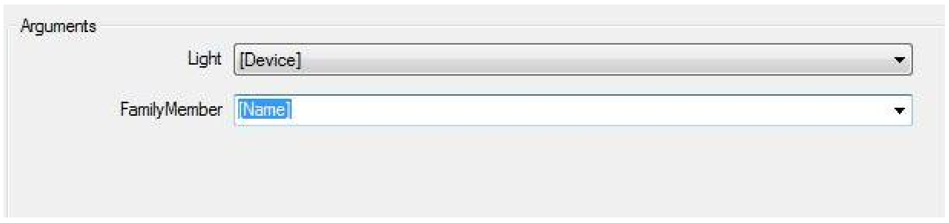
Trigger:

Note that in the selection for the "Light" parameter to program C this example has selected the name of one of program B's parameters. Suppose that when program B was started by program A, program A passed "Home-Light" to program B. Now program B passes "Home-Light" to program C.

Things get a bit more complicated for value parameters due to the UI. When you are filling in the Start-Program properties for a value parameter you want to be able to enter in text and also to be able to select one of the program parameters.

HCA uses one of the Windows UI controls HCA tries to avoid: The dropdown. This is a combination of an edit control and a droplist. The only other place it is used in HCA is in the Variable elements where you can either select an existing variable or type in a new one.

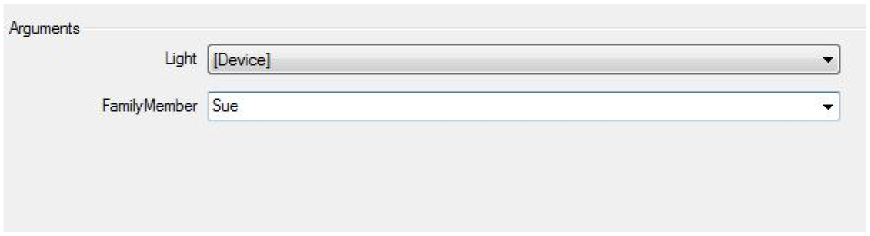
Here I am selecting one of the program parameters for "Family Member"


Arguments

Light [Device]

FamilyMember [Name]

Here I am just entering a value for "Family Member"


Arguments

Light [Device]

FamilyMember Sue

Using the “dropdown” you can either select something from the list or type in the value you want.

This only is used in the case where one parameterized program uses start-program on another parameterized program.

As an example of the utility of this feature, in a sample home we have duplicated several programs each to work with devices and motion sensors in a room. They are all the same program except they work on different devices. Now we can make one common program and use it where needed. This can also be exported and given it to someone else and they could make use of it as is.

Why would you use this feature?

If you are comfortable with traditional programming languages the concept of "subroutines" will be familiar. Using this new feature you can create programs that perform an action without the need to duplicate the programs. For example, a program that implements the interaction between a switch, a keypad, and a motion sensor in a room, can be generalized in such a way – by passing in the switch, motion sensor, and keypad objects – so it can be used for more than one room. Previously you would have to duplicate the program and change the elements that operate upon the specific keypad, motion sensor, and switch in the room.

Parameterized Triggers

In addition to the above changes, triggers can also be parameterized. First some background.

What we have now is that if program A starts program B using the Start-Program VP element then program A can pass parameters to program B.

But there is another way to start a program: a trigger also starts a program. If program A has triggers and takes parameters then why not let the trigger pass parameters to the program?

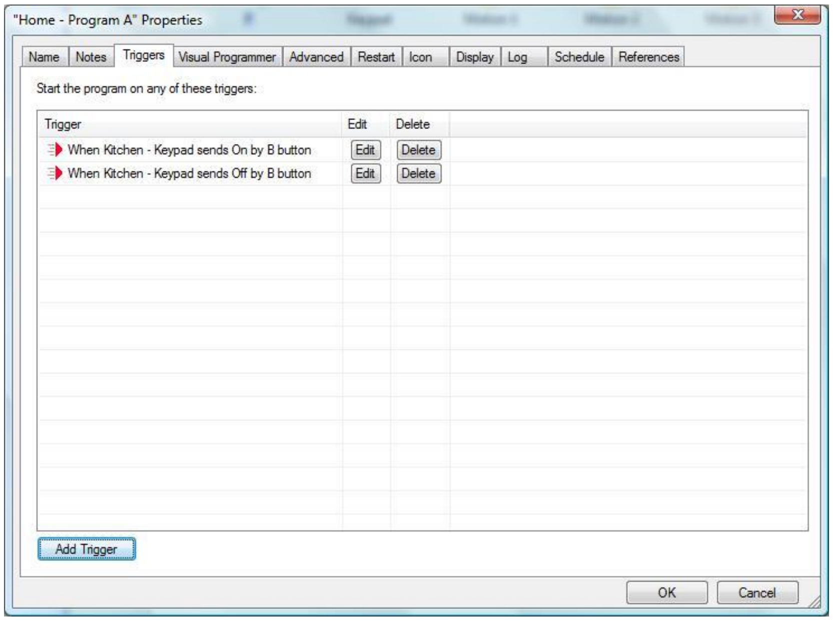
Simplest case: Suppose that program A takes a single object parameter.

Supposed program A is triggered based upon one or more devices. For example, three different switches. The triggering device should be available to the program as an argument. That way the program could operate upon the triggering device without having to test for the starting trigger. And if additional triggers are added then it all just works without having to change the program other than adding the trigger.

This is of course only good for triggers that have an associated object. The State-Change, Insteon, and UPB triggers have an associated object. A weather trigger for example doesn’t.

A more complex case: Suppose that associated with each trigger are the arguments to pass to the program for its parameters. It would be part of the trigger configuration. You would specify in the trigger what values and objects to associate with each of the program parameters.

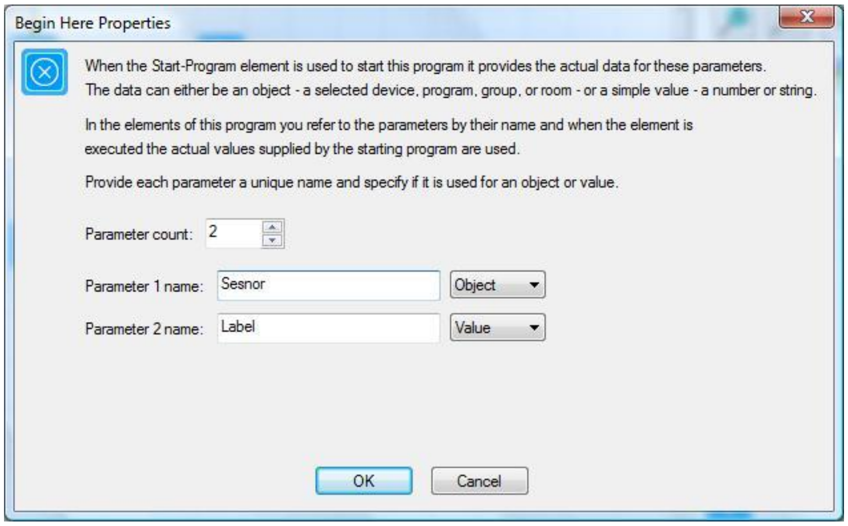
To implement this, the trigger tab in an object’s properties has been changed. Here is what the trigger tab looks like now for programs without parameters.



Start the program on any of these triggers:

Trigger	Edit	Delete
When Kitchen - Keypad sends On by B button	Edit	Delete
When Kitchen - Keypad sends Off by B button	Edit	Delete

Now suppose that the program takes parameters. For example:



When the Start-Program element is used to start this program it provides the actual data for these parameters. The data can either be an object - a selected device, program, group, or room - or a simple value - a number or string.

In the elements of this program you refer to the parameters by their name and when the element is executed the actual values supplied by the starting program are used.

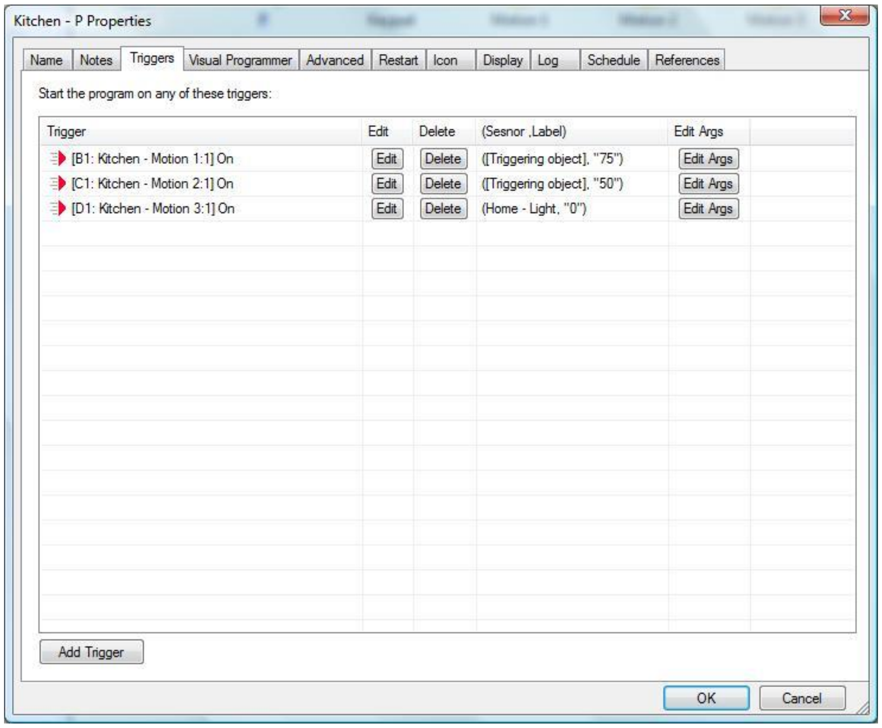
Provide each parameter a unique name and specify if it is used for an object or value.

Parameter count: 2

Parameter 1 name: Sesnor — Object

Parameter 2 name: Label — Value

Now when you are on the triggers tab for this program, this is what you would see:

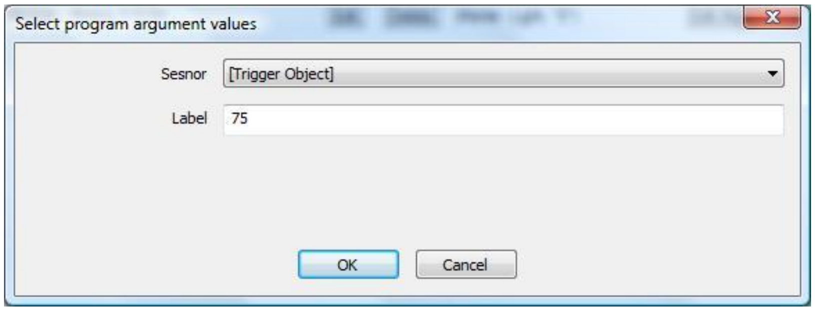


Start the program on any of these triggers:

Trigger	Edit	Delete	(Sesnor ,Label)	Edit Args
[B1: Kitchen - Motion 1:1] On	Edit	Delete	([Triggering object], "75")	Edit Args
[C1: Kitchen - Motion 2:1] On	Edit	Delete	([Triggering object], "50")	Edit Args
[D1: Kitchen - Motion 3:1] On	Edit	Delete	(Home - Light, "0")	Edit Args

There is now a column for the parameters headed with the parameter names. For each trigger the row shows the arguments for each parameter. For an object parameter you can select any of the HCA objects – programs, devices, etc – or [nothing] or [triggering object]. This last case is where the object, if there is one, that is associated with the trigger is passed to the program. For example the keypad object that the user pressed a button on.

This is the dialog when you press the “Edit Args” button.



Sesnor: [Trigger Object]

Label: 75

Why would you use this feature?

Same reasons as you would use parameterized programs in general. The ability to pair a trigger with an object or a value can make it simpler in creating the program. If for example you were to have a program that triggers on messages from many different devices you could use the triggering device in the program elements that operate upon that device.

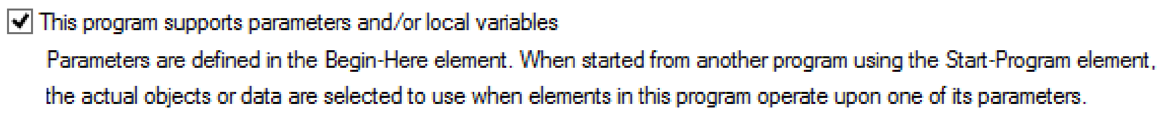
Local and Global Variables

Several Visual Programmer elements in some way operate on variables. Some directly modify a variable like Variable-Set, Make-Variable-Yes and other use variables to hold results. For example, the HTTP element.

These variables can be local or global. A local variable is a variable that exists only during the execution of the program. When the program terminates all the variables are deleted. The advantage of this is that if more than one instance of the program is executing simultaneously then each has its own variables and the two can co-exist without changing data in the other. This can also be a disadvantage in that no other program can see the variables or their values.

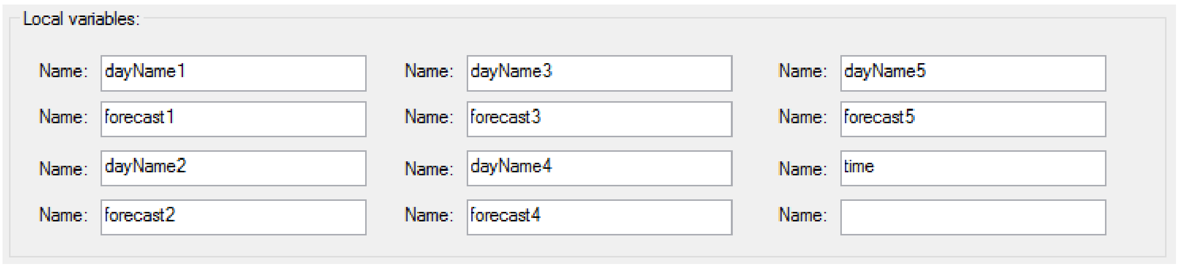
A global variable is a variable that exists even after the program terminates. Global variables are a great place to record data that you want more than one program to operate upon. Or for one program to save a value that another program later can get access to.

In a method like parameterized programs, you first must enable local variable support for a it on the "Advanced" tab.


[✓] This program supports parameters and/or local variables

Parameters are defined in the Begin-Here element. When started from another program using the Start-Program element, the actual objects or data are selected to use when elements in this program operate upon one of its parameters.

Then in the Begin-Here element local variable names are specified.

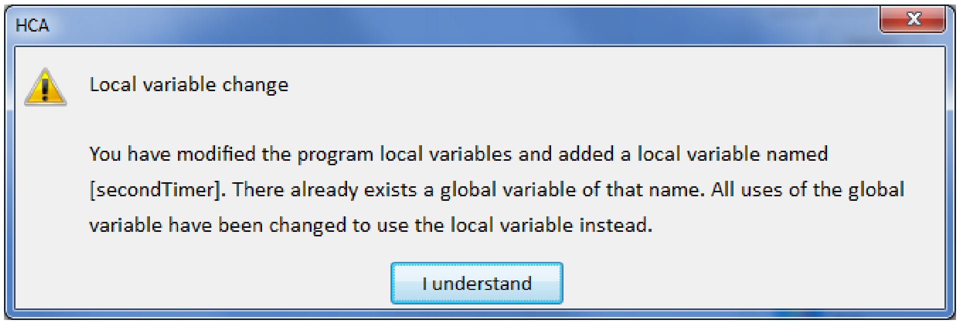

Local variables:

Name:	dayName1	Name:	dayName3	Name:	dayName5
Name:	forecast1	Name:	forecast3	Name:	forecast5
Name:	dayName2	Name:	dayName4	Name:	time
Name:	forecast2	Name:	forecast4	Name:	

As you can see from this section of the Begin-Here element properties, a program can have a maximum of 12 local variables.

Changing global to local

What happens if you enter the name of a variable into an element and then later add it to the list of local variables? When you first create a new variable by entering the name into the properties of an element it is created as a global variable if it isn't one listed in the local variable list in Begin-Here. If you later then add it to the Begin-Here list, it is changed from global to local:


HCA

Local variable change

You have modified the program local variables and added a local variable named [secondTimer]. There already exists a global variable of that name. All uses of the global variable have been changed to use the local variable instead.

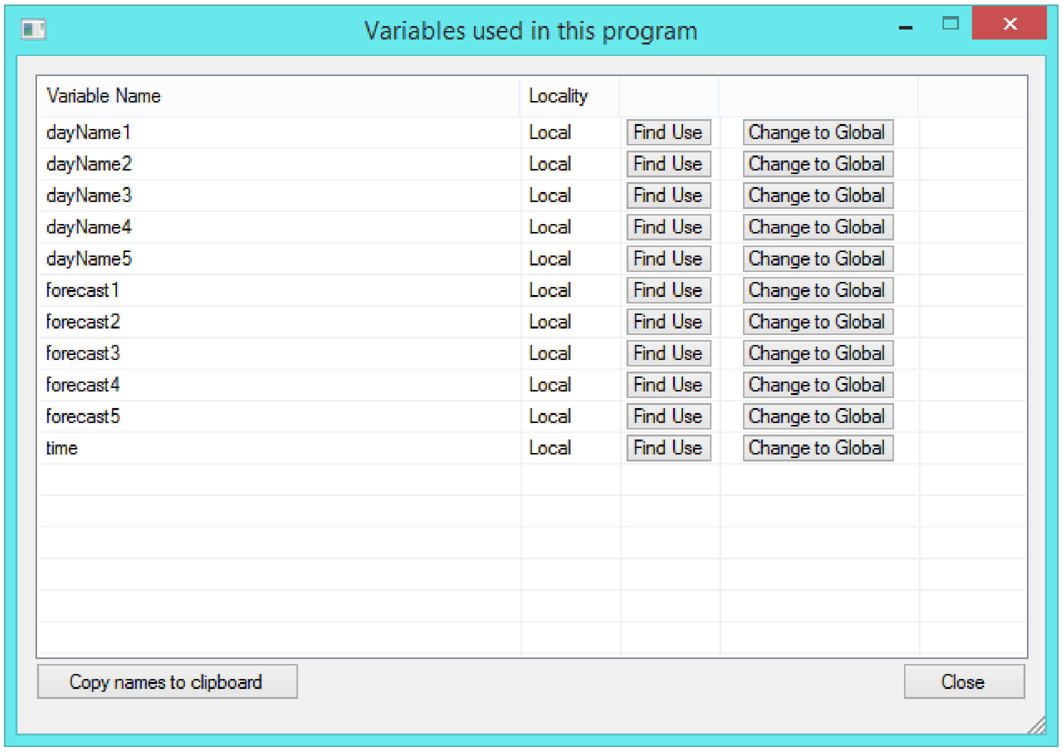
The global variable already created isn't removed and you should use the Variable Inventory to remove it if not needed.

A similar action happens if you remove a local variable. In that case any uses of the variable name are changed to use a global variable of that name.

Variable Use

In programs that use many variables it can become confusing as to what you are using and if they are local or global. There is a tool that can help and is started from the “variable” tool in the tools panel.

This tool opens a window that shows all variable usage in the program and if the variable is local or global.


Variables used in this program

Variable Name	Locality		
dayName1	Local	Find Use	Change to Global
dayName2	Local	Find Use	Change to Global
dayName3	Local	Find Use	Change to Global
dayName4	Local	Find Use	Change to Global
dayName5	Local	Find Use	Change to Global
forecast1	Local	Find Use	Change to Global
forecast2	Local	Find Use	Change to Global
forecast3	Local	Find Use	Change to Global
forecast4	Local	Find Use	Change to Global
forecast5	Local	Find Use	Change to Global
time	Local	Find Use	Change to Global

Note: You can leave the variables Window open on the screen while you work in the Visual Programmer. As you add or change elements in the program, the variable window updates to show you the current variable usage.

Also, in this window, for each variable you can change it from local to global or global to local. If you make that change a popup tells you the effect of that action.

If you press the “Find Use” button associated with a variable, an element that uses the variable is selected.

Tip: In HCA Options is a setting on the Visual Programmer tab that causes any new variables created as you are editing the program to become local variables automatically.

☑ When editing elements, new variables in expressions are created as local variables if the "This program supports parameters and/or local variables" option enabled

Setting properties for program elements

Now that you know how programs start and how to draw them using the program canvas, let’s look at the elements of those programs, and see what they can do. You control what an element does by setting its properties, using the properties dialog box specific to that element.

Hint: Most programs will not use all the different types of elements that HCA provides. The sample file provided when you installed HCA contains several simple and several complex programs. A good way to learn about the Visual Programmer is to take a look at what these programs do.

web tip: Refer to the technical notes section on the web site for more information on the MyHome sample file and its programs.

There are two ways that you can open the properties dialog box for an element. The first way is automatic: as you add an element to the program canvas, its dialog box opens so that you can set the properties for the element.

The second way is to use the right mouse button:

- Click an element and choose Properties from the popup menu or double click on the element. The dialog box for the element opens so that you can enter or change its properties.

Elements

This section lists the various elements used in programs and provides details about how you can set and use their properties. The properties dialog boxes for the elements are very similar and very easy to use. They have different components, many of which will be familiar to you.

Note all elements are fully explained here. Some are very specific to some automation hardware. These are fully explained in the appendix for the hardware they apply to.

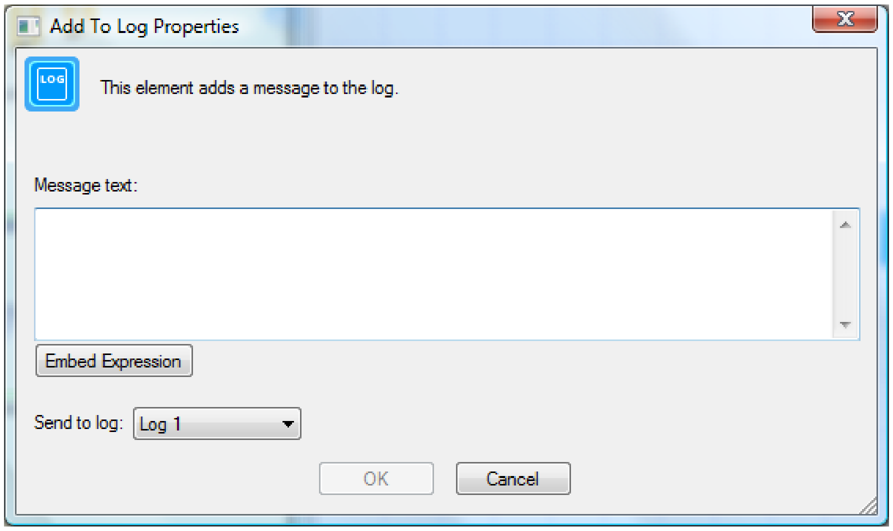
Start Here

This element provides a marker for where a program starts running. It properties only if the option is enabled on the Advanced Options tab for parameters and/.or local variables as described above.

Add to log

This element adds an entry to the log. To see that entry, open a log viewer from the Control category.

You might use this element if you are attempting to determine why a program is not doing what you want it to do, or if you want to make a note whenever the program is started.


Add To Log Properties

This element adds a message to the log.

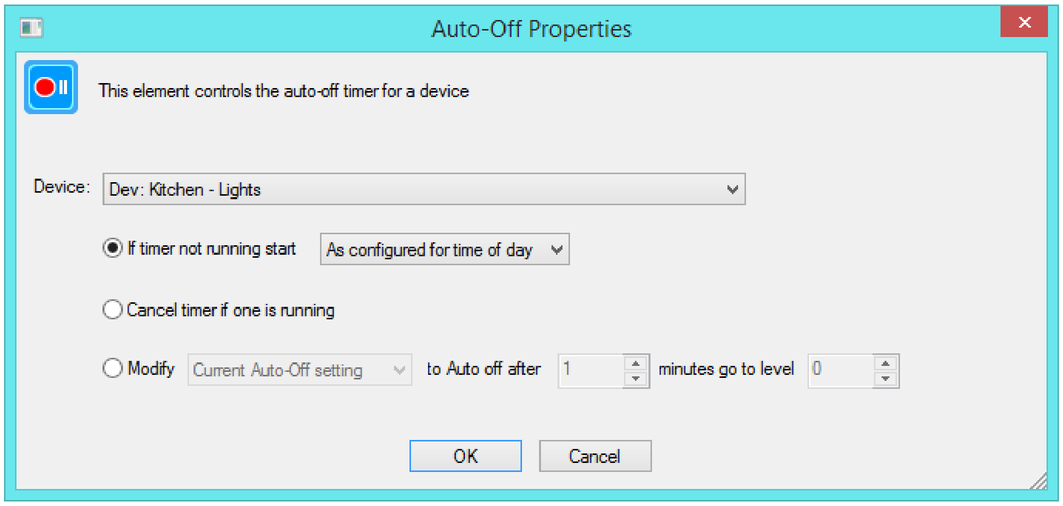
Message text:

Send to log: Log 1

Hint: You can embed HCA expressions in the text between %%. See the chapter on expressions and the expression builder. For example, The time is %_now()%

Auto Off

This element configures the auto off settings for a device. You may want to change the auto off settings for a device, for example, different times of the day.


Auto-Off Properties

This element controls the auto-off timer for a device

Device: Dev: Kitchen - Lights

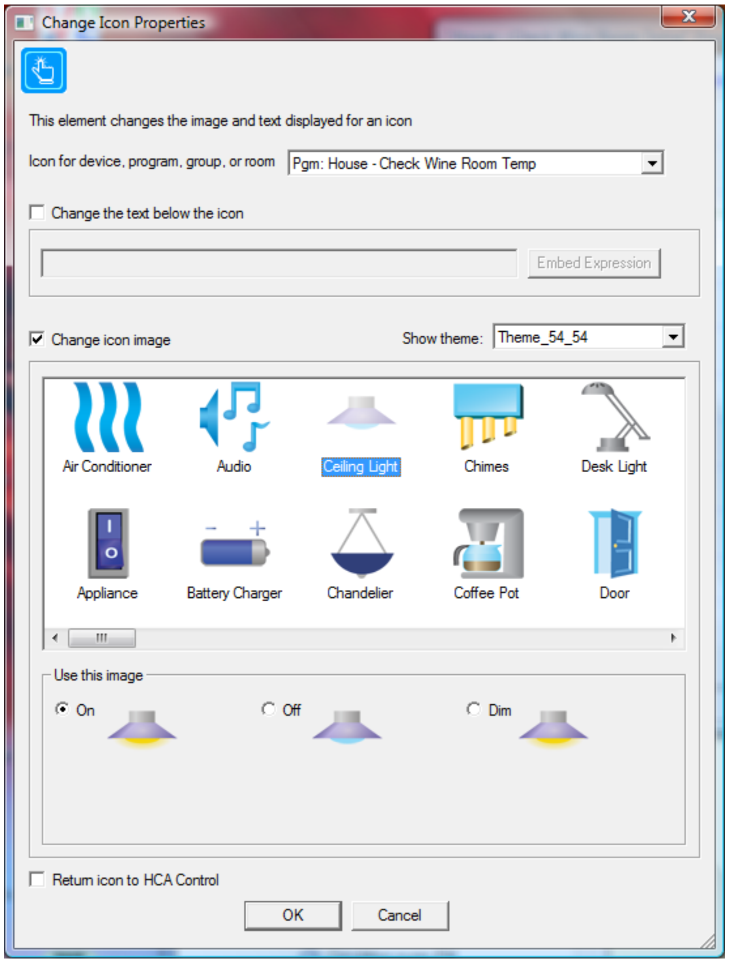
● If timer not running start As configured for time of day

○ Cancel timer if one is running

○ Modify Current Auto-Off setting to Auto off after 1 minutes go to level 0

Change icon

The change icon element is one of the more complex elements. It allows you change the icon for any device, program, or group and/or the text below that icon.



Change Icon Properties

This element changes the image and text displayed for an icon

Icon for device, program, group, or room: Pgm: House - Check Wine Room Temp

☐ Change the text below the icon

☑ Change icon image   Show theme: Theme_54_54

Use this image: (●) On ( ) Off ( ) Dim

☐ Return icon to HCA Control

With this dialog box, you can either select which icon to use and its representation (on, off, dim), the text below the icon, or both the icon and the text.

1. Select the device, program, or group to change.
2. Select the icon you want to use
3. Click the button for the icon representation that you want—On, Off, or Dim.
4. Click OK when done.

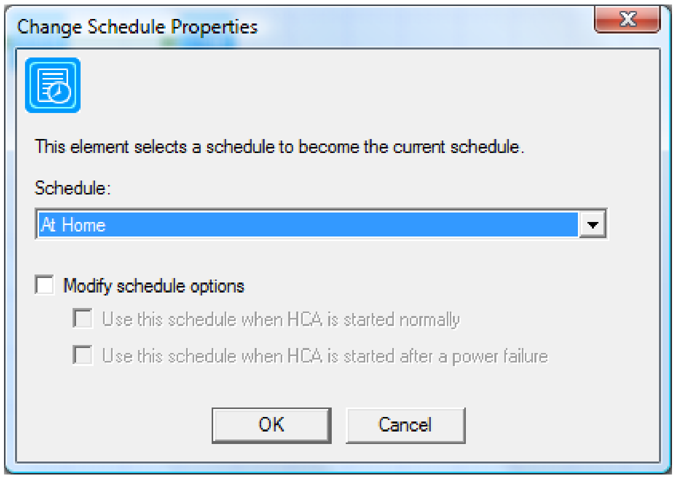
This is a good element to use when you have special needs for a program. For example, a program could modify a door sensor device icon and the text below to show if a garage door is open or closed.

Hint: You can embed HCA expressions in the text. See the chapter on expressions and the expression builder.

You can draw these additional icons yourself with a paint program and add them to an HCA icon theme. Refer to the Icon Theme chapter for more information.

Change schedule

This element changes the current schedule. With this element, you can cause HCA to stop monitoring one schedule and start monitoring another.


Change Schedule Properties

This element selects a schedule to become the current schedule.

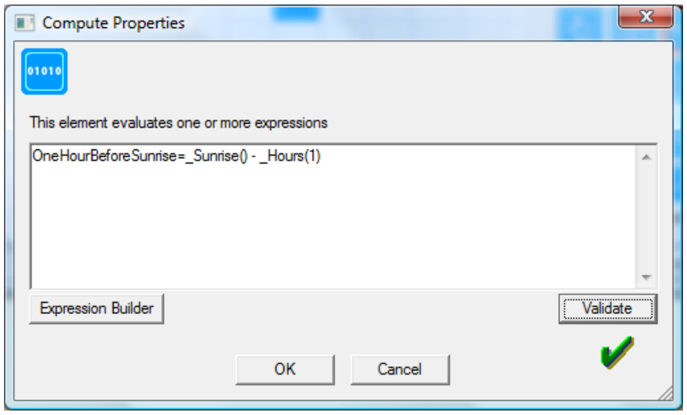
Schedule: At Home

- [ ] Modify schedule options
  - [ ] Use this schedule when HCA is started normally
  - [ ] Use this schedule when HCA is started after a power failure

From the Schedule box, select the schedule you want to be the current schedule.

Compute and Compute Test

These two elements assign values to variables and test those values. The Compute element properties are:


Compute Properties

This element evaluates one or more expressions

```
OneHourBeforeSunrise=_Sunrise() - _Hours(1)
```

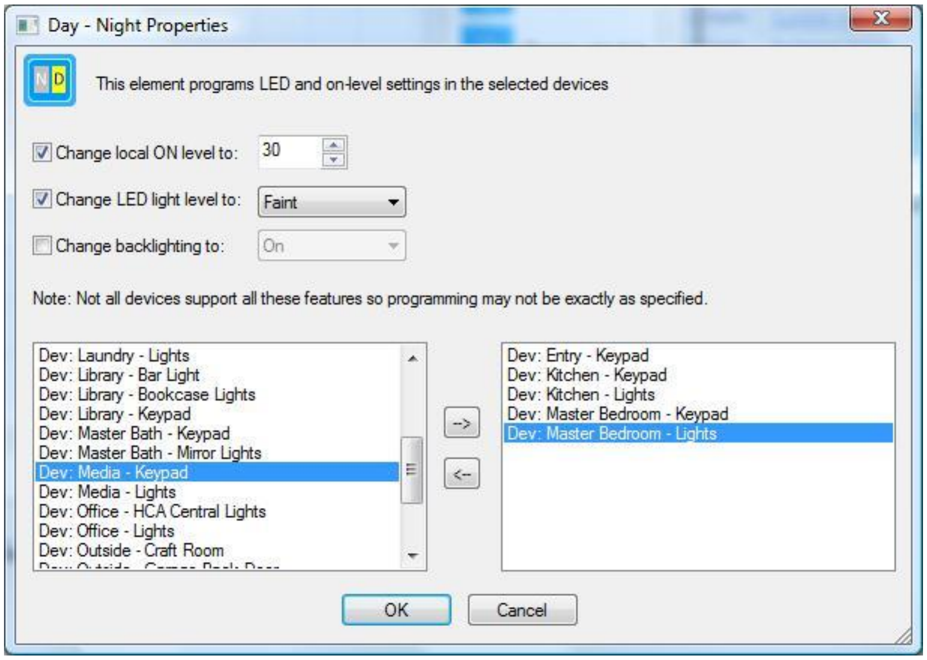
The Compute Test element is similar to the Compute element except that the expression entered is evaluated and if true, execution follows the Yes path, otherwise the No path.

These two elements are used by more sophisticated programs and are not necessary for most programs.

What you place in the edit control is complex and powerful. It is described in the chapter on expressions and the Expression Builder.

Day Night

The *Day Night* element reprograms devices that have adjustable LEDs – keypad buttons or indicator LEDs – to a different level, and/or changes the local on-level for switches. Not all devices support these features. By creating programs that run at night you can "dim down" annoying lighting and then increase it during the day.


Day - Night Properties

This element programs LED and on-level settings in the selected devices

☑ Change local ON level to: 30

☑ Change LED light level to: Faint

☐ Change backlighting to: On

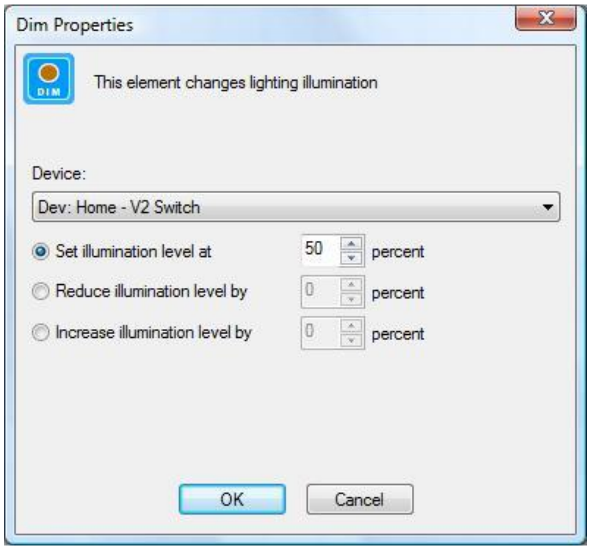
Note: Not all devices support all these features so programming may not be exactly as specified.

Dev: Laundry - Lights
Dev: Library - Bar Light
Dev: Library - Bookcase Lights
Dev: Library - Keypad
Dev: Master Bath - Keypad
Dev: Master Bath - Mirror Lights
Dev: Media - Keypad
Dev: Media - Lights
Dev: Office - HCA Central Lights
Dev: Office - Lights
Dev: Outside - Craft Room

Dev: Entry - Keypad
Dev: Kitchen - Keypad
Dev: Kitchen - Lights
Dev: Master Bedroom - Keypad
Dev: Master Bedroom - Lights

Dim

This operation dims a device or group. It is a little more complex than the On and Off elements.



This element changes lighting illumination

Device:

Dev: Home - V2 Switch

[✓ Set illumination level at] 50 percent

Reduce illumination level by 0 percent

Increase illumination level by 0 percent

First select the device you want to dim the set the illumination level.

There are three ways to dim a device (or group):

- The first way is to set the illumination level—100% is full bright and 0% is full dim.
- The second way reduces the illumination level by a percentage.
- The third way increases the illumination level by a percentage.

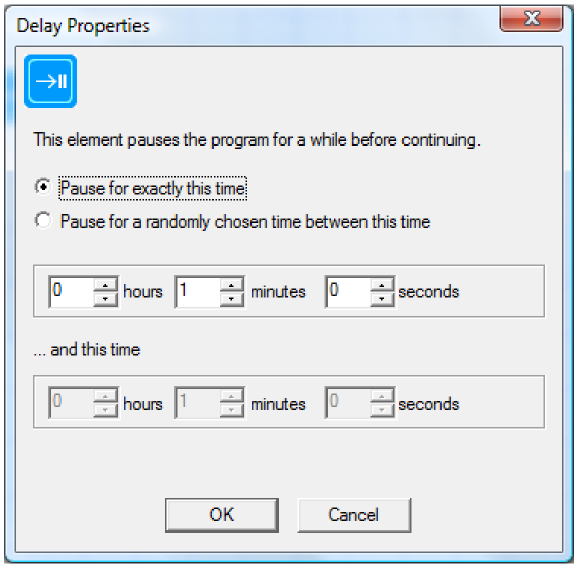
Using a percentage works, regardless of how many dim levels the device supports.

The percentage you select converts to a number of dim levels using this formula:

$$\text{Dim levels} = \frac{\text{(supported dim levels)} \times \text{(percentage you entered)}}{100}$$

Delay

This element causes a program that is running to pause for a while. You set the length of the pause using the element properties.



Delay Properties

This element pauses the program for a while before continuing.

[✓ Pause for exactly this time]

Pause for a randomly chosen time between this time

0 hours 1 minutes 0 seconds

... and this time

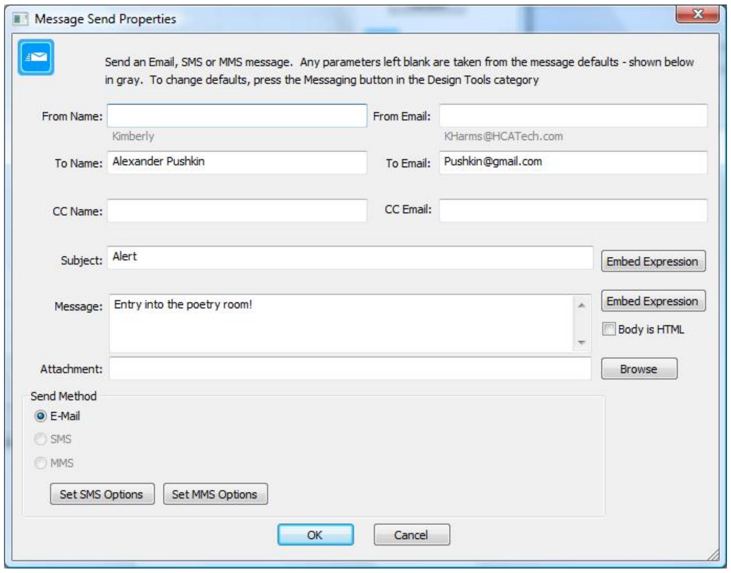
0 hours 1 minutes 0 seconds

Use the hours, minutes, and seconds controls to set the hours, minutes, and seconds of delay. If you choose the second option, then HCA computes a different delay each time the element is executed.

Hint: Even if you choose an exact delay time, your program may not continue precisely after the delay time you set. But it should continue within a few seconds of the time you specified. Note that this is the only place in HCA where you enter a time to the second.

Email / SMS

The send email element is used to send email, SMS, or MMS messages.



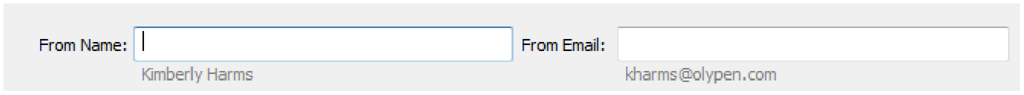
Message Send Properties

Send an Email, SMS or MMS message. Any parameters left blank are taken from the message defaults - shown below in gray. To change defaults, press the Messaging button in the Design Tools category

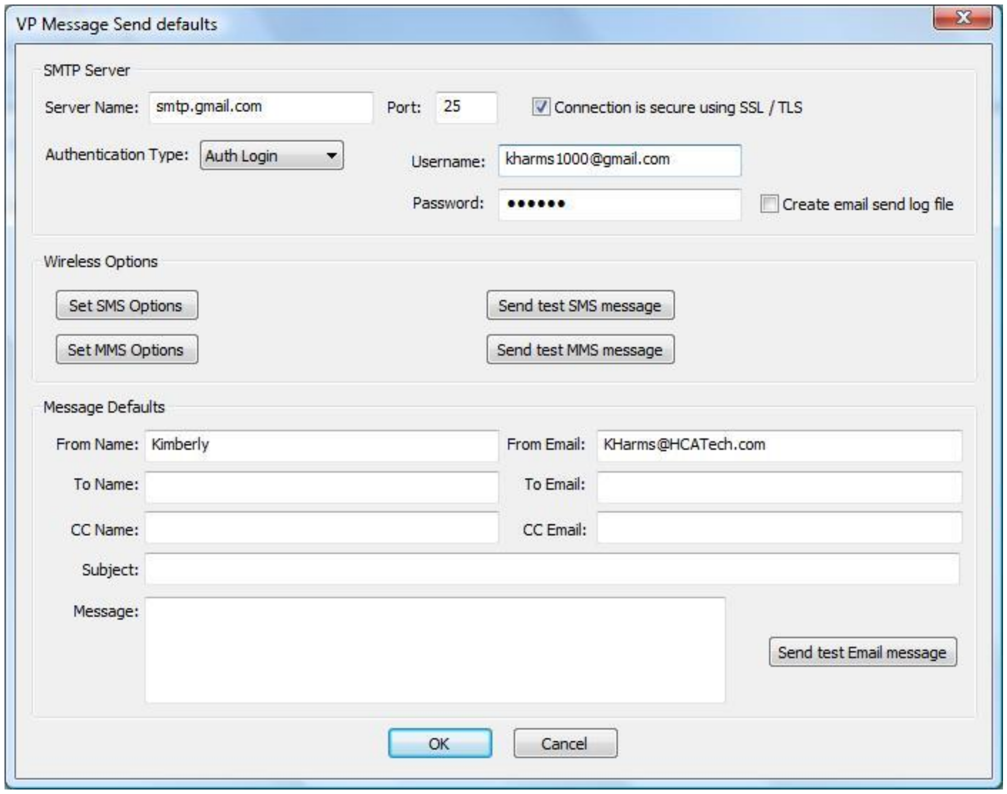
From Name: Kimberly
From Email: KHarms@HCATech.com
To Name: Alexander Pushkin
To Email: Pushkin@gmail.com
CC Name:
CC Email:
Subject: Alert
Message: Entry into the poetry room!
Body is HTML: ☐
Attachment:

Send Method
[✓ E-Mail]
SMS
MMS

This dialog captures the parameters of the message. Any fields you have defined in the message defaults are shown in gray below the field. If you don't enter anything into these fields, then the defaults are used when the message is sent.


From Name: Kimberly Harms
From Email: kharms@olypen.com

Before you use this element you have to configure messaging. Press the *Messaging Setup* button from the *Design* category.


VP Message Send defaults

SMTP Server
Server Name: smtp.gmail.com
Port: 25
[✓] Connection is secure using SSL / TLS
Authentication Type: Auth Login
Username: kharms1000@gmail.com
Password: ••••••
[ ] Create email send log file

Wireless Options

Message Defaults
From Name: Kimberly
From Email: KHarms@HCATech.com
To Name:
To Email:
CC Name:
CC Email:
Subject:
Message:

In addition to the email server parameters you can also see that the remainder of the dialog is very similar to the *Email* element properties dialog. When the *Email* element executes any parameter not specified in its properties is taken from the messaging defaults. In this way you can enter the "from" information – and perhaps the "to" as well - so it doesn't have to be duplicated in each *Email* element added to a program.

Hint: To enter multiple recipients separate them with ';'in both the "To name" and "To email" edit boxes.

Exit

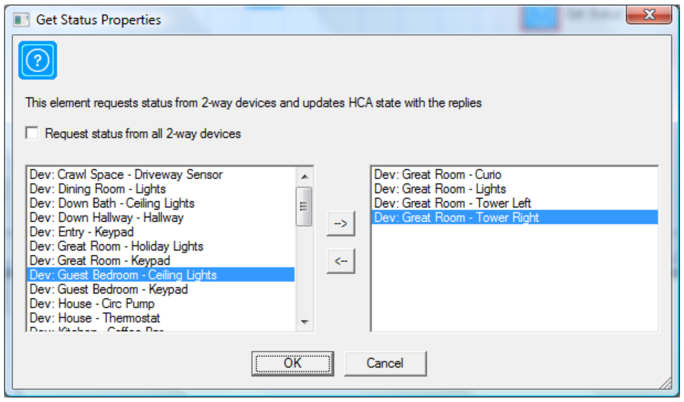
When a running program executes this element, the program ends. The *Exit* element has no properties.

You may want to use this element in conjunction with the *Test* and *Repeat* elements (described later).

Each path of a program does not need to end in an exit element.

Get Status

The *Get Status* element initiates a poll of one or more devices that respond to status requests.



This element requests status from 2-way devices and updates HCA state with the replies

☐ Request status from all 2-way devices

Dev: Crawl Space - Driveway Sensor
Dev: Dining Room - Lights
Dev: Down Bath - Ceiling Lights
Dev: Down Hallway - Hallway
Dev: Entry - Keypad
Dev: Great Room - Holiday Lights
Dev: Great Room - Keypad
Dev: Guest Bedroom - Ceiling Lights
Dev: Guest Bedroom - Keypad
Dev: House - Circ Pump
Dev: House - Thermostat

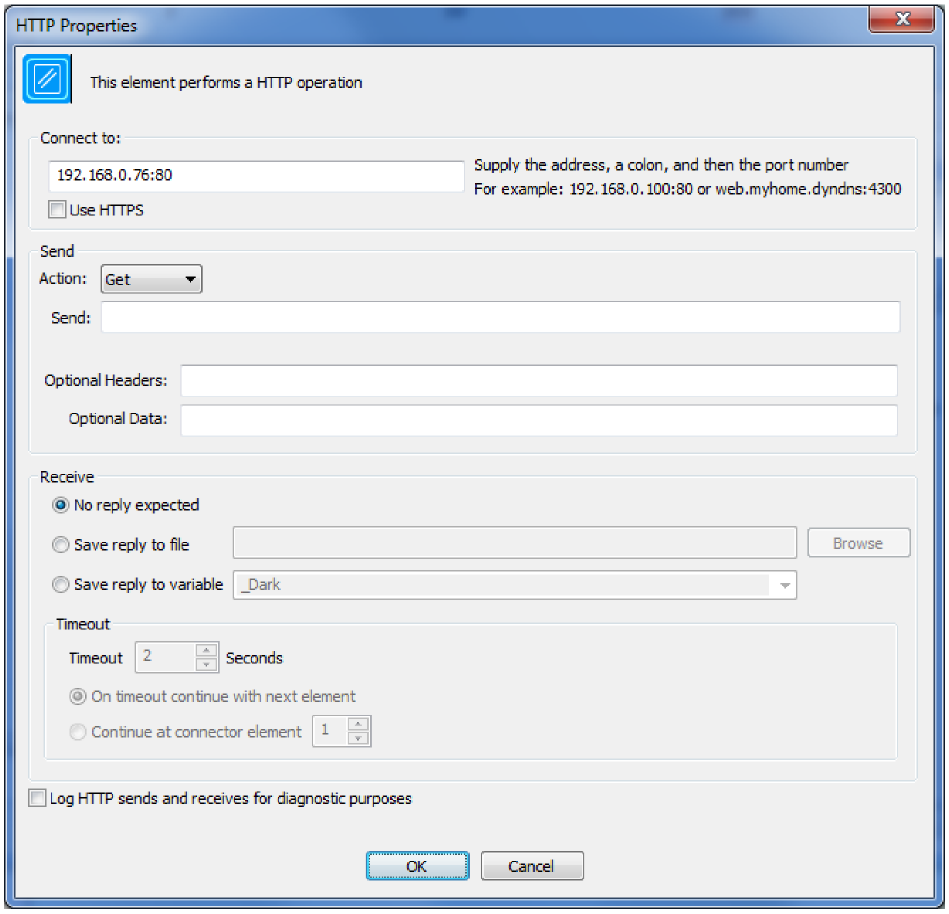
Dev: Great Room - Curio
Dev: Great Room - Lights
Dev: Great Room - Tower Left
Dev: Great Room - Tower Right

You can either select the devices you want to poll or you can just say to get status from every device that responds to status requests.

One important point about this element is that these status requests are not performed when the *Get Status* element is executed. The status requests are queued for later transmission. There is no way to predict when they will be sent.

HTTP

The New HTTP element lets you make a HTTP Get, Put, and Post operation to a web server. What is sent, including optional headers and data is specified in the element properties.


HTTP Properties
This element performs a HTTP operation
Connect to:
192.168.0.76:80
Supply the address, a colon, and then the port number
For example: 192.168.0.100:80 or web.myhome.dyndns:4300
Use HTTPS
Send
Action: Get
Send:
Optional Headers:
Optional Data:
Receive
No reply expected
Save reply to file
Browse
Save reply to variable _Dark
Timeout
Timeout 2 Seconds
On timeout continue with next element
Continue at connector element 1
Log HTTP sends and receives for diagnostic purposes
OK
Cancel

The “Destination” is anything that resolves to a IP address and is usually but doesn’t have to be a 4-part IP address.

The reply can be ignored or captured to a variable or a file. If a reply is expected then a timeout in seconds can be specified. If the operation doesn’t receive the reply in the specified time the next element to execute is specified. The element doesn’t check for the result from the operation. If there is a reply within the time limit then execution continues at the next element.

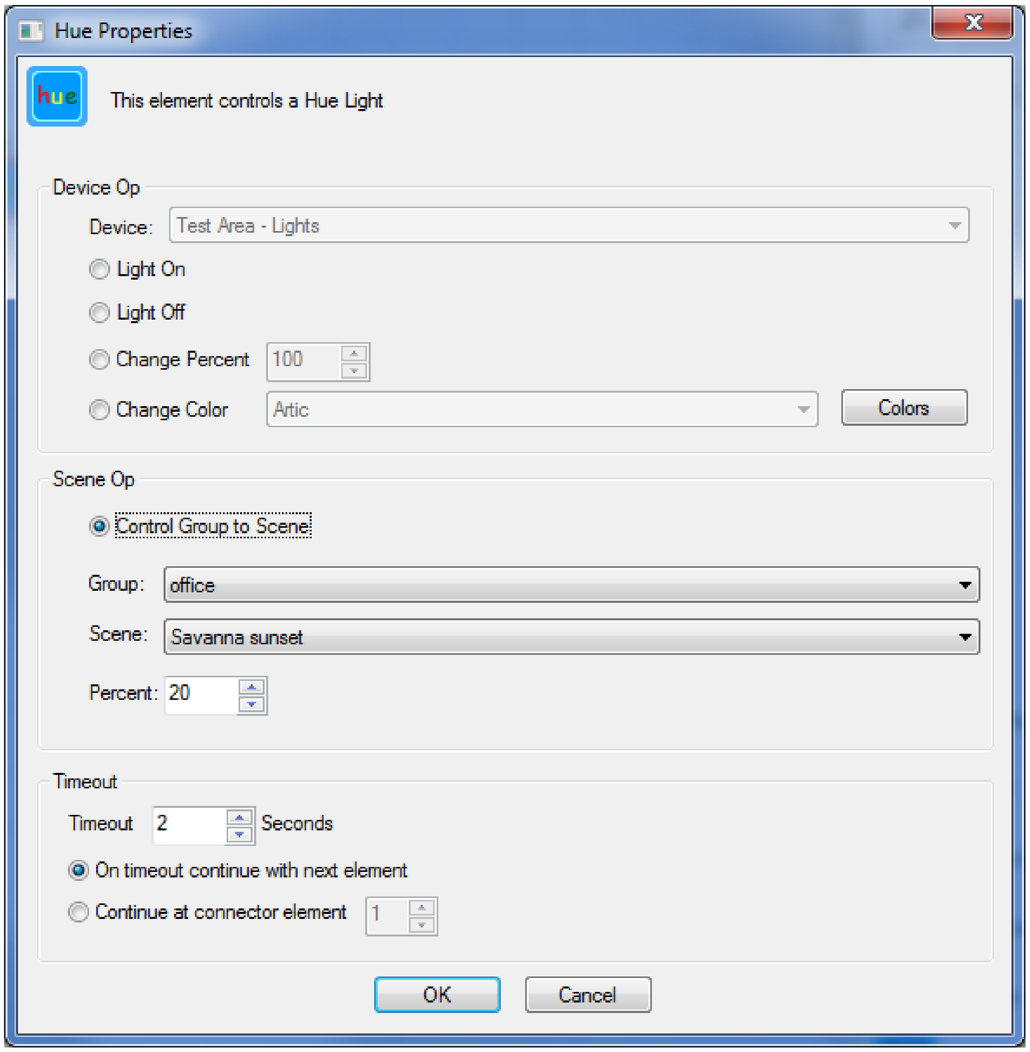
Hue

The Hue element lets you control Phillips Hue lights.

As you can see there are device and scene operations. A device can be controlled on, off, to a level, or to a color.

Color in the Hue world is complex and here are recommendations on how to handle it: If possible, use the Hue app to create scenes and use them in HCA. It is easy to create and modify scenes and quick for HCA to activate them.

If you want to work with color directly then HCA gives you the ability to define, name, and use colors. Press the “Colors” button in the Hue VP element.



Hue Properties

This element controls a Hue Light

Device Op

Device: Test Area - Lights

Light On

Light Off

Change Percent 100

Change Color Artic

Colors

Scene Op

[✓ Control Group to Scene]

Group: office

Scene: Savanna sunset

Percent: 20

Timeout

Timeout 2 Seconds

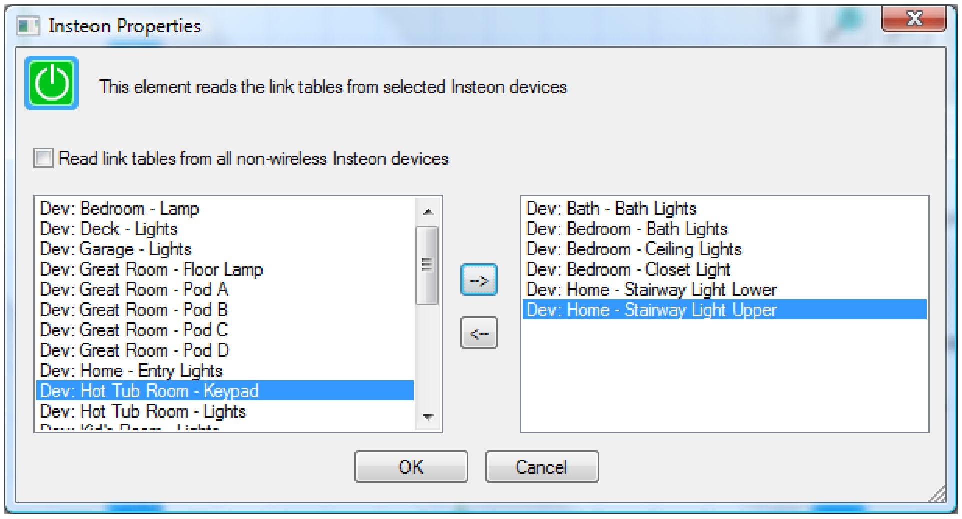
[✓ On timeout continue with next element]

Continue at connector element 1

OK Cancel

Insteon

The Insteon element reads the linking tables from one or more selected devices and saves that state within HCA. As described in the Insteon Appendix, these linking tables enable HCA to kep track of the state of your devices and also to correctly respond to messages from them.


Insteon Properties

This element reads the link tables from selected Insteon devices

☐ Read link tables from all non-wireless Insteon devices

Dev: Bedroom - Lamp
Dev: Deck - Lights
Dev: Garage - Lights
Dev: Great Room - Floor Lamp
Dev: Great Room - Pod A
Dev: Great Room - Pod B
Dev: Great Room - Pod C
Dev: Great Room - Pod D
Dev: Home - Entry Lights
Dev: Hot Tub Room - Keypad
Dev: Hot Tub Room - Lights

Dev: Bath - Bath Lights
Dev: Bedroom - Bath Lights
Dev: Bedroom - Ceiling Lights
Dev: Bedroom - Closet Light
Dev: Home - Stairway Light Lower
Dev: Home - Stairway Light Upper

This element is typically used a in a program that you schedule to run periodically as a way of keeping your HCA design up to date with your hardware.

Make variable No / Make variable Yes / Not variable

HCA provides variables as a very simple way for programs to keep track of things. Unlike other objects in HCA, variables are not created by using a wizard. Variables are created when you first use them. Variables are very important for some aspects of programs, and variables have a few simple rules:

- You can have any number of variables you want in any program.
- Variables can take on many values but these elements only work with Yes and No values.
- Each variable has its own name.

Variables are most useful with the *Test* element since it can check the value of a variables and do different things based upon the outcome of that test.

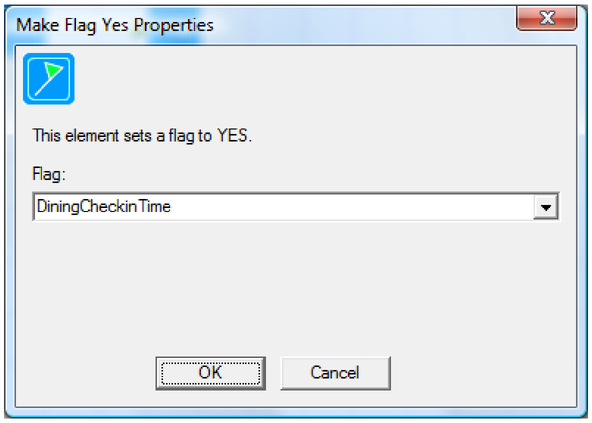
There are three elements that work with variables:

Make variable No—Makes the variable contain a No value

Make variable Yes—Makes the variable contain a Yes value

Not variable —This reverses the value of the variable. If the variable contains a Yes before the *Not* element is executed, it contains a No after. If the variable contains a No before the *Not* element is executed, the *Not* changes it to a Yes.

These three elements have similar properties.



This element sets a flag to YES.

Flag: DiningCheckinTime

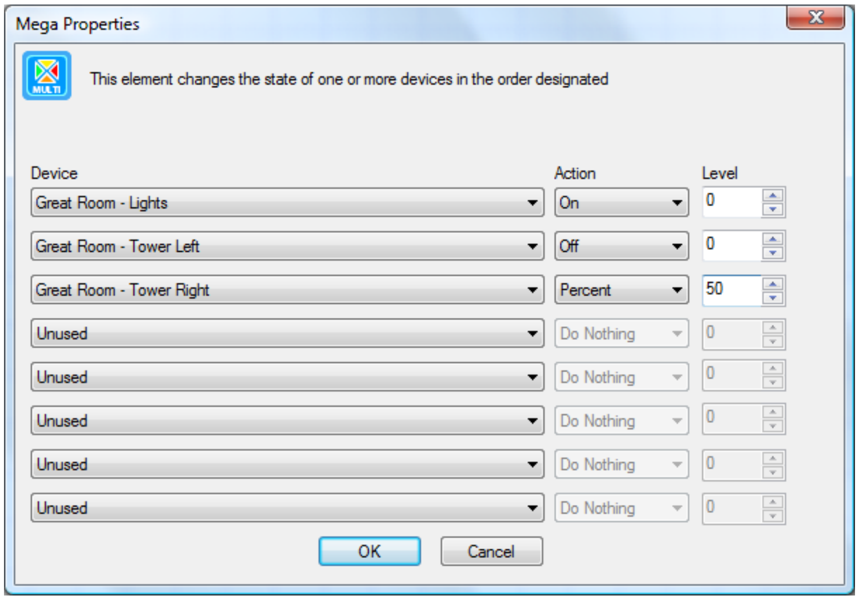
When setting the properties, select a variable you already have created, or type the name for a new one.

Once you create a new variable it appears in the variable inventory.

The properties for a variable are its name, its current value (Yes or No), and the value that HCA should assign to the variable when HCA first loads your design. To inspect the variable'sproperties open the Variable Inventory from the *Control* category.

Multi

This element allows you to perform what the On, Off, and Dim elements do on a set of devices. Used in this way rather than a series of the other elements may make constructing programs simpler.

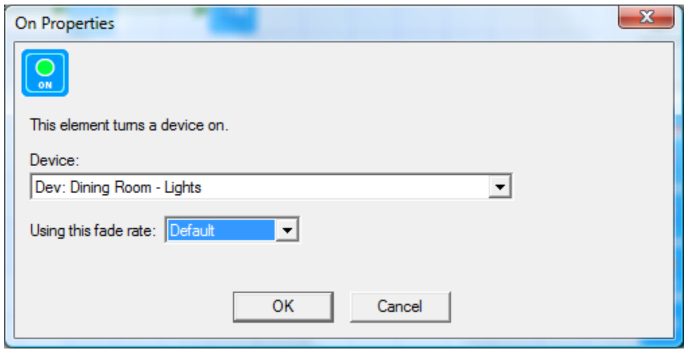


This element changes the state of one or more devices in the order designated

Device	Action	Level
Great Room - Lights	On	0
Great Room - Tower Left	Off	0
Great Room - Tower Right	Percent	50
Unused	Do Nothing	0
Unused	Do Nothing	0
Unused	Do Nothing	0
Unused	Do Nothing	0
Unused	Do Nothing	0

On

This element sends an ON command to a device or group, or starts another program with an ON command.



This element turns a device on.

Device: Dev: Dining Room - Lights

Using this fade rate: Default

Select from the dropdown the device or program to control with an ON.

Some kinds of devices show additional parameters for you to select in the ON element properties.

You can also use this element to start another program running. If you do, that program runs concurrently with the current program. The current program continues to the next element as soon as the other program starts.

Off

This element sends an OFF command to a device or group, or starts another program with an OFF command. Its properties dialog box looks very similar to the On properties.

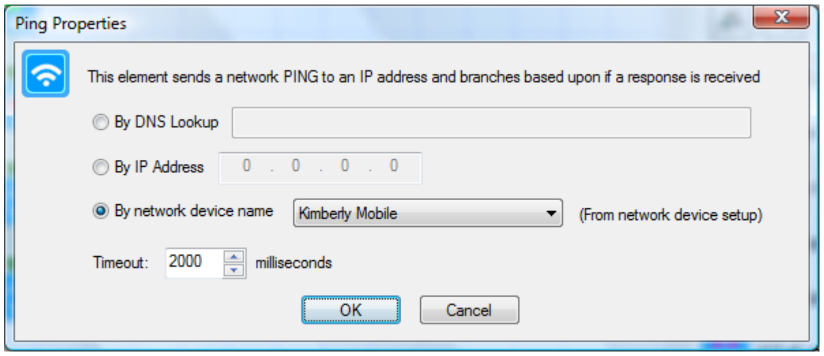
From the list, select the device or group you want to turn off.

Some kinds of devices will show additional parameters for you to select in the OFF element properties.

You can also use this element to start another program running. If you do, that program runs concurrently with the current program. The current program continues to the next element as soon as the other program starts.

PING

The Ping element is part of the "Network Devices" component and uses several of the configuration settings defined there. To configure those settings, use the *Network devices* button in the *Design* ribbon category.



This element sends a network PING to an IP address and branches based upon if a response is received

By DNS Lookup

By IP Address 0 . 0 . 0 . 0

[✓ By network device name] Kimberly Mobile (From network device setup)

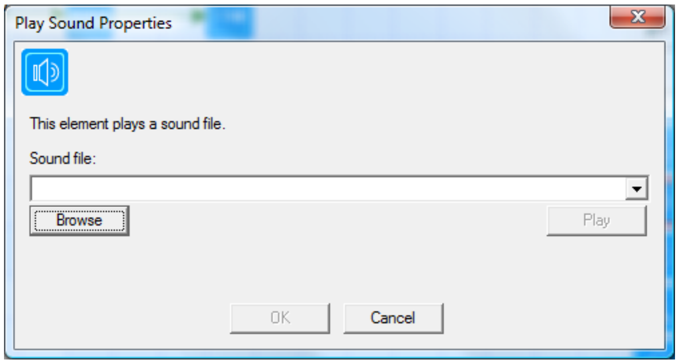
Timeout: 2000 milliseconds

A device is specified by entering a DNS lookup name, an IP address, or selected from one of the network devices configured in the Network Devices dialog. When the element is executed, a network PING is sent to the device and if it responds in the time allotted then, like the Test element, the "yes" path is taken from the element otherwise the "no" path is taken.

Play sound

The *play sound* element plays a WAV file using whatever sound system your computer contains. .

Hint: In client-server mode, each client has configuration to say if it plays sound or not. The WAV file must be present on each client if they are to play the file.



Play Sound Properties

This element plays a sound file.

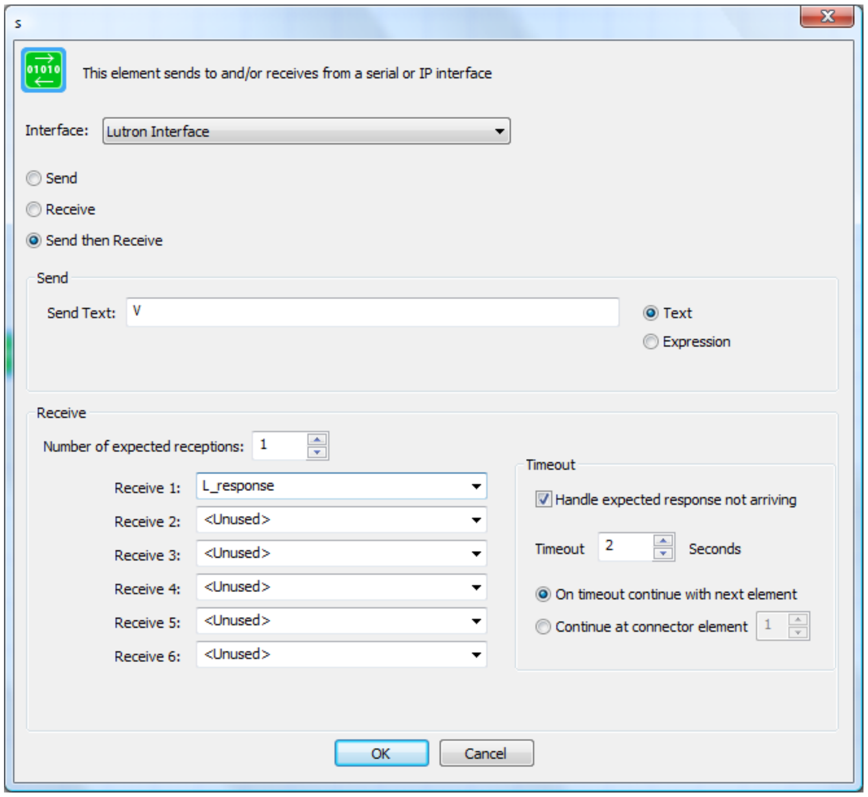
Sound file:

If you create your design file (the .HCA file) on one computer, and then move it to another computer, check your programs for *play sound* elements. If you have *play sound* elements, you must make sure that you move the WAV files and place them in the same directories as they were on the machine you are moving from.

Hint: Use the Design Inspector (from the HCA menu) to verify that all the paths to WAV files in your programs do indeed locate their WAV files.

Port I/O

The Port IO element works with the generic serial and generic network interfaces configured in HCA Options on the hardware tab. This element can send, receive, or send and then receive from the port.


s

This element sends to and/or receives from a serial or IP interface

Interface: Lutron Interface

Send
Receive
[✓ Send then Receive]

Send

Send Text: V

[✓ Text]
Expression

Receive

Number of expected receptions: 1

Receive 1:	L_response
Receive 2:	<Unused>
Receive 3:	<Unused>
Receive 4:	<Unused>
Receive 5:	<Unused>
Receive 6:	<Unused>

Timeout

[✓ Handle expected response not arriving]

Timeout 2 Seconds

[✓ On timeout continue with next element]
Continue at connector element 1

Select from the interface dropdown the name of the interface to use, then choose the action option for send, received, or send then receive.

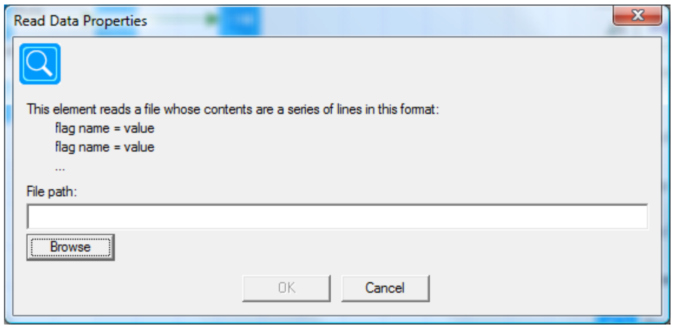
If sending, the text to be sent is entered. This can be the actual text or the result of an expression evaluation if the *Expression* option is used.

If receiving, enter the number of expected responses and then, for each response, select the variable that reception is assigned to, or the name of a new variable that the response is assigned to.

As with any communication, you can handle the case of what happens if an expected reception doesn't arrive. The *Timeout* group contains the options for this. Enter the max number of seconds to wait for the expected receptions. If that timeout limit is reached then execution can either continue at the next element or at an element pointed to be the numbered connector.

Read Data

The *Read Data* element is a way to get data, perhaps produced by another Windows program, into HCA. The file it reads consists of a number of lines where each line is of the form variableName = expression. In effect, each line is treated as if it was the text in a Compute element.

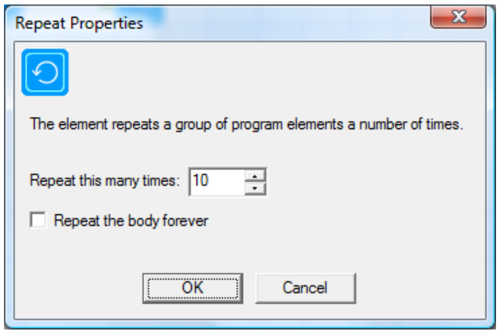

Read Data Properties

This element reads a file whose contents are a series of lines in this format:
flag name = value
flag name = value
...

File path:

Repeat

This element allows a program to repeatedly execute one or more elements.


Repeat Properties

The element repeats a group of program elements a number of times.

Repeat this many times: 10

☐ Repeat the body forever

Set the number of times that you want the elements to repeat or tick the box to have the elements repeated continually.

If you choose to repeat continuously, the program never stops until the program is stopped(by selecting Stop from the popup menu, another program using the Stop element, or by HCA itself terminating (maybe due to a power failure).

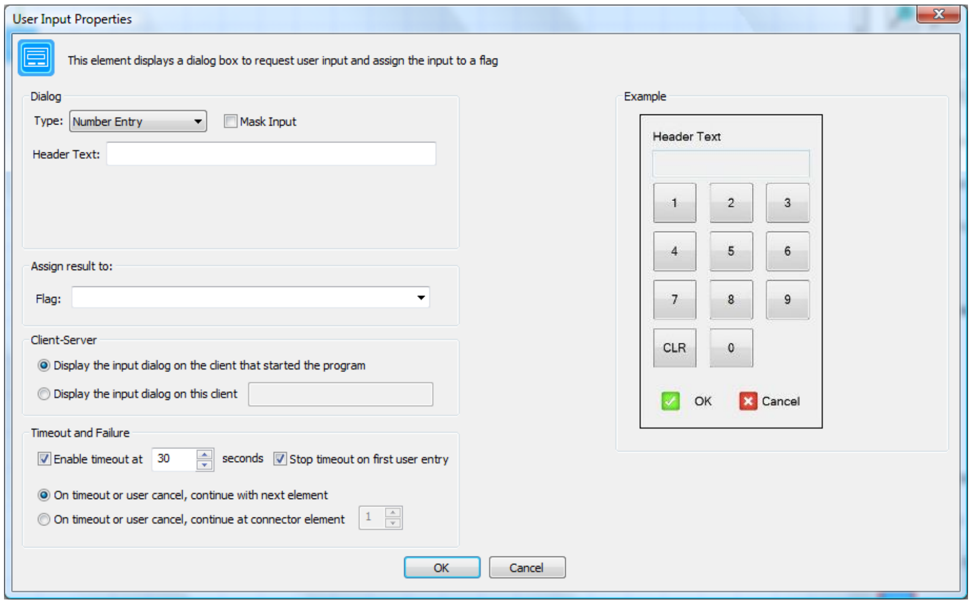
If in the body of the repeat – those elements that are repeated – if an exit element is executed this can also terminate the repeat. This is configurable in *HCA Options.*

There is more information on how to construct programs using the Repeat element later on in this chapter.

Request Input

The *Request Input* element presents a dialog box to the user and waits for input. Here are several different dialog types possible. For example, using the *Request Input* dialog a program can create a user interface for modifying a schedules time, or request a numeric entry code for a control system.

The element properties dialog shows an example of the general shape and content of the displayed dialog.



User Input Properties

This element displays a dialog box to request user input and assign the input to a flag

Dialog

Type: Number Entry [ ] Mask Input

Header Text:

Assign result to:

Flag:

Client-Server

[●] Display the input dialog on the client that started the program

[○] Display the input dialog on this client

Timeout and Failure

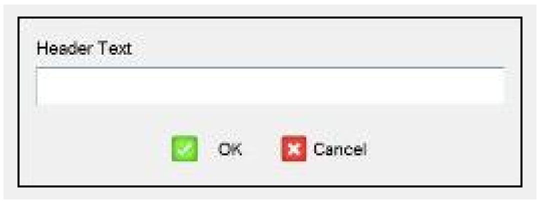
[✓] Enable timeout at 30 seconds [✓] Stop timeout on first user entry

[●] On timeout or user cancel, continue with next element

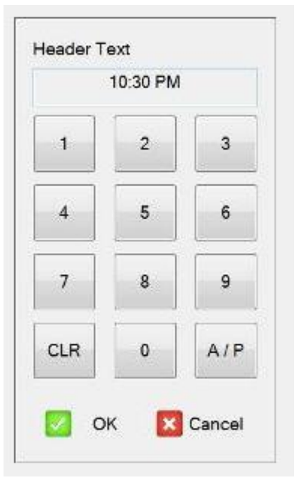
[○] On timeout or user cancel, continue at connector element 1

In the above screen image the numeric entry dialog is shown. The other dialog types are:

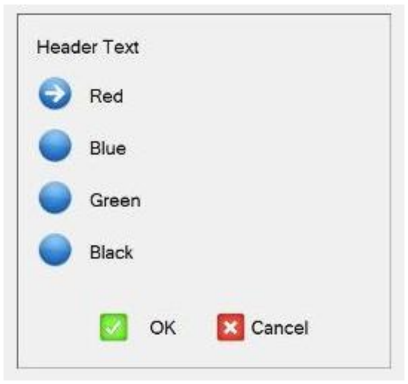
Text Entry:



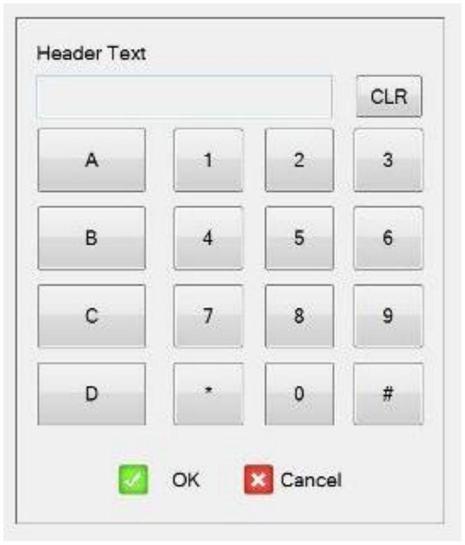
Time Entry:



Choice:



Security Console:



Each different dialog type has different configuration where the various text and button labels can be modified for your application.

The other parts of these element properties are common for all dialog types. These are:

<u>Assign Results To</u>
Select the name of a variable or enter a new variable name to assign the input – what the user entered – to.

<u>Client Server</u>
If HCA is operating in client-server mode, the dialog appears on either a designated client – using the client name which is set in *HCA Options* on each client – or on the client that started the program that contained the *User Input* element.

<u>Timeout</u>
The user input dialog can be limited in the length of time that it appears on the screen. If a timeout occurs then the next element executed in the program can be the next element or an element pointed to by the numbered connector element.

Resume and Suspend

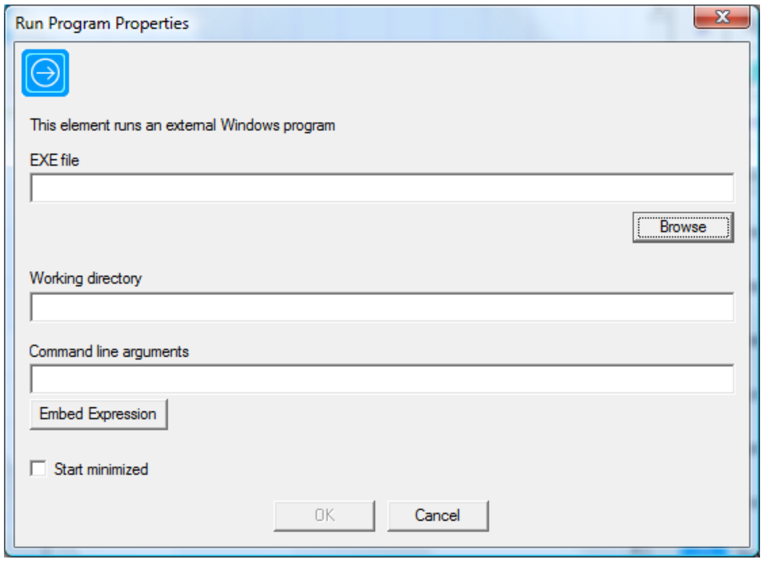
The suspend element is very similar to the suspend menu selection for devices, programs, and groups. In the properties of this element the device, program, or group is chosen and the type of suspension is selected. A item can be suspended from:

- Schedule control. Any schedule entries for this item don't happen while the item is suspended.
- Program control. Any On, Off, or Dim elements of any program don't control the item while it is suspended.
- Remote control. Any command received for a program or group doesn't start the program or control the group.

The Resume element is used to undo the effects of a Suspend.

Run

When you use the Run element you can start another Windows program. Don't confuse that with starting a HCA program. A Windows program is started from the Windows Start button or from a command line.



This element runs an external Windows program

EXE file

Working directory

Command line arguments

☐ Start minimized

The properties for the Run element are:

- The path to the executable file. Typically these end in .EXE
- The directory that the program is started in. The working directory is the directory where a program will first find any files it opens.
- The command line to pass to the program.
- An option to start the program in the minimized state.

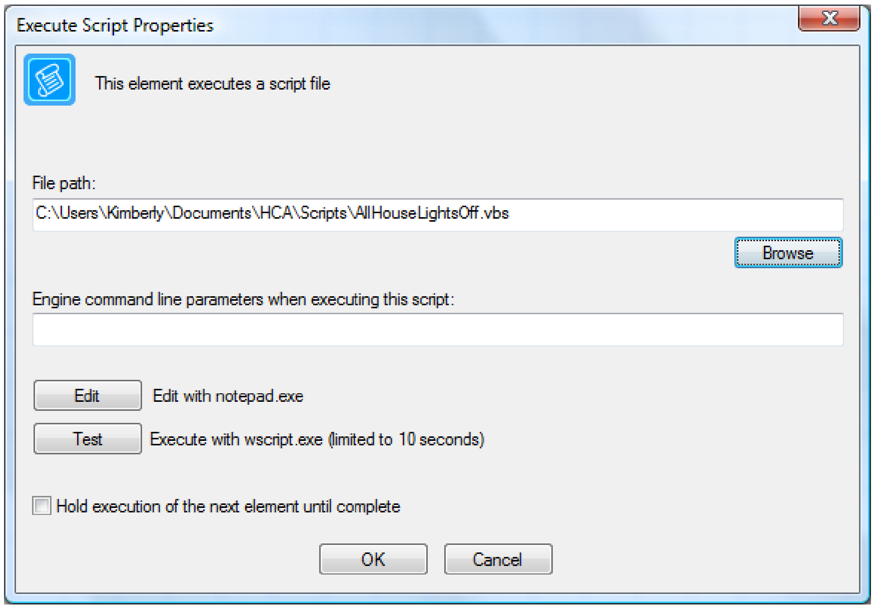
This element is very similar to starting a program from the Run command you get from the Start button.

Hint: You can embed HCA expressions in the text. See the chapter on expressions and the expression builder.

Script

The run script element starts a text-based script using the currently selected script engine.

This is a large topic and is fully described in the Script chapter.



This element executes a script file

File path:
C:\Users\Kimberly\Documents\HCA\Scripts\AllHouseLightsOff.vbs

Engine command line parameters when executing this script:

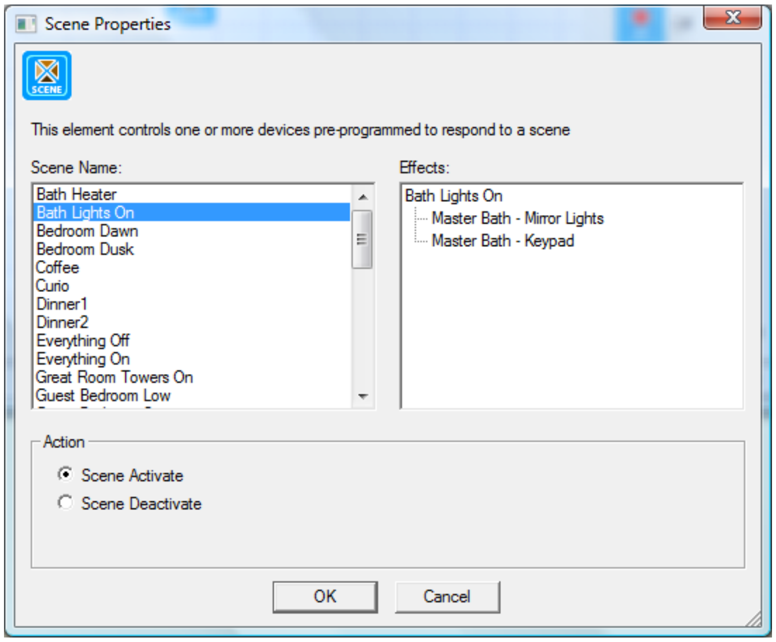
Edit with notepad.exe

Execute with wscript.exe (limited to 10 seconds)

☐ Hold execution of the next element until complete

Scene

The scene element is used with devices that can be programmed to react to commands to activate and deactivate scenes.



This element controls one or more devices pre-programmed to respond to a scene

Scene Name:
Bath Heater
Bath Lights On
Bedroom Dawn
Bedroom Dusk
Coffee
Curio
Dinner1
Dinner2
Everything Off
Everything On
Great Room Towers On
Guest Bedroom Low

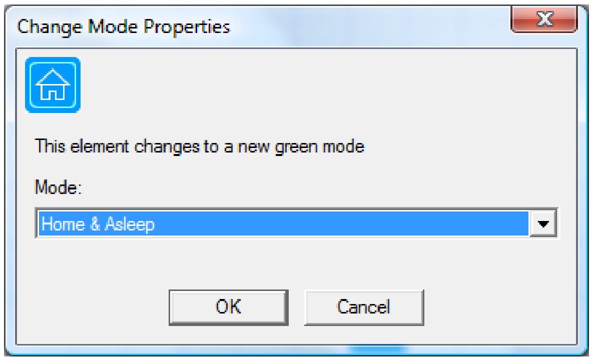
Effects:
Bath Lights On
Master Bath - Mirror Lights
Master Bath - Keypad

Action
[✓ Scene Activate]
Scene Deactivate

Scene names are listed in the left column and the effects of that scene in the right column. Depending upon the type of scene – Insteon, UPB, or XP – various options may display at the bottom of the dialog.

Set Mode

The Set Mode element changes the current home mode tone of the home modes configured in the Home Properties.



This element changes to a new green mode

Mode: Home & Asleep

Show Display

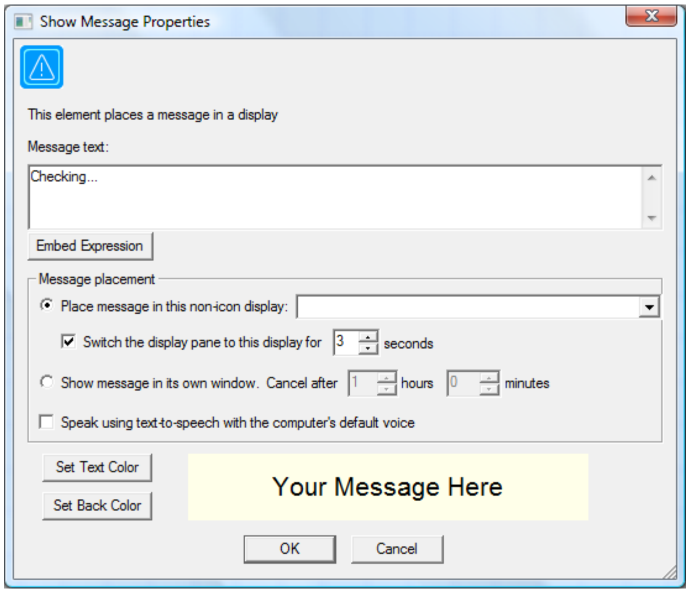
This element causes the display pane to show the display selected.

Show Message

The *Show message* element creates a message that is either displayed in the HCA display pane or in its own window. If the message is shown in the display pane it persists for a few seconds before being removed.

If the message is displayed in its own window, that window persists until you close it or it expires. You can set the expiration time in the HCA properties on the display tab.

The properties for the Show message element allow you to enter the text to display, the destination of the message (display pane or own window) and the color for the text and background.


Show Message Properties

This element places a message in a display

Message text:

Checking...

Message placement

(●) Place message in this non-icon display:

[✓] Switch the display pane to this display for 3 seconds

( ) Show message in its own window. Cancel after 1 hours 0 minutes

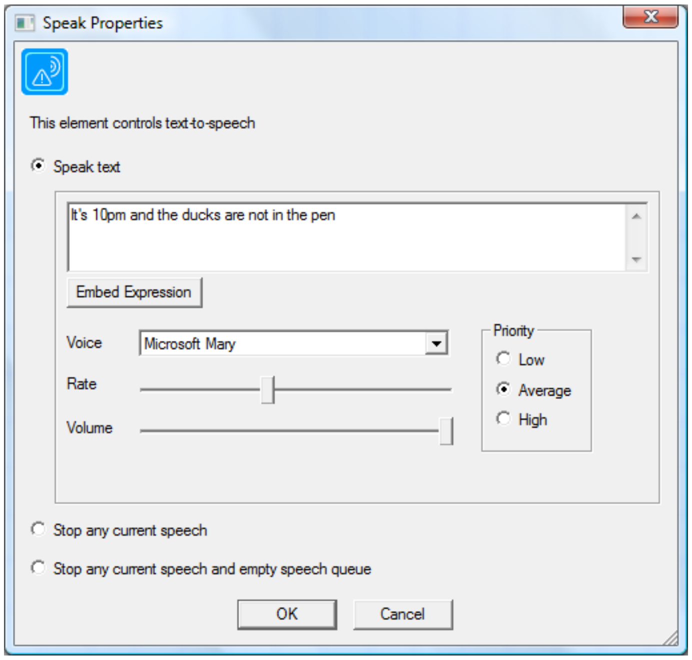
[ ] Speak using text-to-speech with the computer's default voice

The text can contain expressions embedded in %%. These are evaluated before the final text is assembled and displayed.

Speak

The *Speak* program element works with a Text-to-speech engine installed on your computer to convert a piece of text into spoken words played through the computer sound system. HCA doesn’t contain a text-to-speech engine and you must acquire one and install it before HCA can use it. Check the Windows Control Panel Speech applet for information and installation on text-to-speech.

The properties for the Speech element are:


Speak Properties

This element controls text-to-speech

(●) Speak text

It's 10pm and the ducks are not in the pen

Voice: Microsoft Mary

Priority: ( ) Low [✓ Average] ( ) High

( ) Stop any current speech

( ) Stop any current speech and empty speech queue

There are three options:

1. Speak some text. This text can contain embedded expressions that HCA evaluates and replaces with the result.
2. Stop any current speech.
3. Stop any current speech and empty the speech queue.

It is important to remember that the Speech element adds the message to the text-to-speech engine queue and moves on. As each item in the queue is completed, then the next item in the queue is spoken. The last two options give you the ability to cancel the current text and to empty the speech queue.

Start Program

The *Start program* element starts another program running. (Refer to the description of the *On* and *Off* elements). Like the *On* or *Off* elements, the *Start program* element starts another program, but it does ***not*** continue until the started program completes.

1. Select the program that you want this element to start.
2. Select the type of command that starts the program. This is only important if the program being started tests for the command that starts it. (Refer to the description of the *Test* element.).

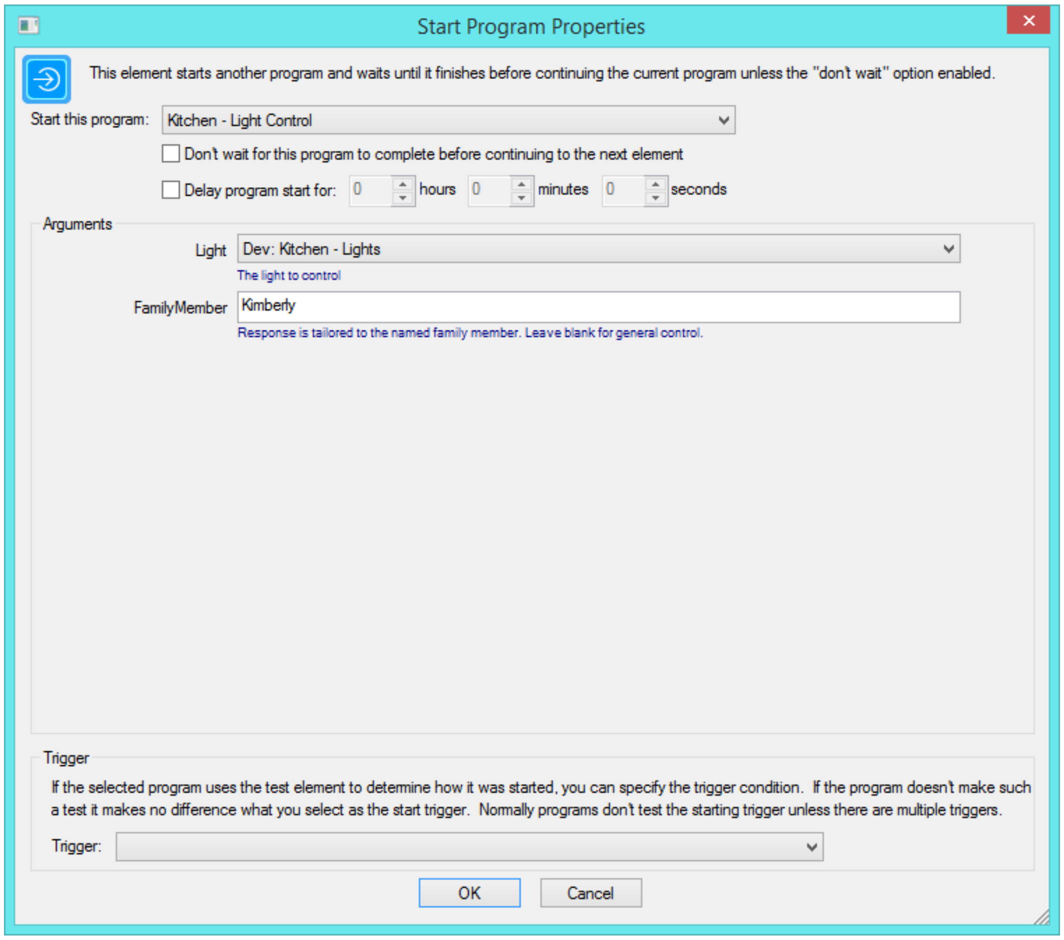
One program can cause another program to start in one of two ways: using this element or using the ON or OFF elements.

In the case of the Start-Program element, the named program acts as a "subroutine" of the calling program. For example, if program A contained a Start-Program element naming program B, then when the Start-Program element is executed program B starts and program A stops until program B completes.

In the case of the ON and OFF elements, the named program runs as a co-routine to the starting program. That is, they both run concurrently. For example, if program A contains an ON element naming program B, then when the ON element executes program B starts and program A immediately continues on at the next element after the ON element.

There is a major advantage of the Start-Program element instead of the ON and OFF elements: With Start-Program you get to specify a trigger to use when starring the named program and with parameterized programs – described later in this chapter - you also got to supply arguments for those parameters. Neither can be done with the ON or OFF element. Also, to choose a program as the target of the ON or OFF element the program had to have an ON or OFF trigger.

There is an additional property of the Start-Program element that allows you to overcome these limitations.



This element starts another program and waits until it finishes before continuing the current program unless the "don't wait" option enabled.

Start this program: Kitchen - Light Control

☐ Don't wait for this program to complete before continuing to the next element

☐ Delay program start for: 0 hours 0 minutes 0 seconds

Arguments

Light: Dev: Kitchen - Lights

The light to control

FamilyMember: Kimberly

Response is tailored to the named family member. Leave blank for general control.

Trigger

If the selected program uses the test element to determine how it was started, you can specify the trigger condition. If the program doesn't make such a test it makes no difference what you select as the start trigger. Normally programs don't test the starting trigger unless there are multiple triggers.

Trigger:

If this option is enabled, then the program is started as a co-routine as described above.

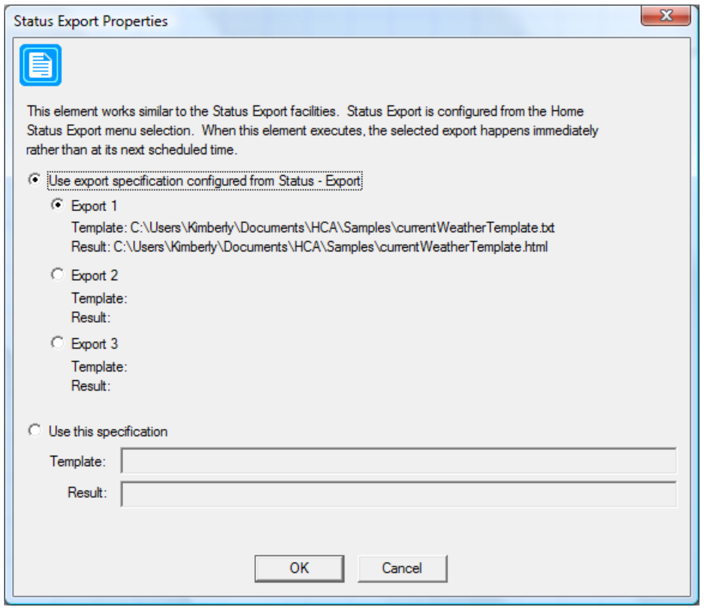
The major advantage over the ON and OFF elements is that the started program can be supplied with parameters and/or a trigger. And since the ON and OFF elements require that a program have an ON or OFF trigger, the started program no longer needs those triggers to execute as a co-routine when started by the start-program element.

In addition to this option, there is an additional option that will start the program but only a time period has elapsed. This can be very useful for performing some action, and then waiting for some time to perform a second action. Rather than use a Delay element to do the waiting, you can break the second action out into its own program and then start that program.

In the element is the time to delay before the start happens.

Status Export

The Status Export element starts a status export using one of the three configured status exports. The properties of this element are:



This element works similar to the Status Export facilities. Status Export is configured from the Home Status Export menu selection. When this element executes, the selected export happens immediately rather than at its next scheduled time.

(●) Use export specification configured from Status - Export

(●) Export 1
Template: C:\Users\Kimberly\Documents\HCA\Samples\currentWeatherTemplate.txt
Result: C:\Users\Kimberly\Documents\HCA\Samples\currentWeatherTemplate.html

( ) Export 2
Template:
Result:

( ) Export 3
Template:
Result:

( ) Use this specification
Template:
Result:

The action of the Status Export and how it is configured is described in the *Status Export* chapter.

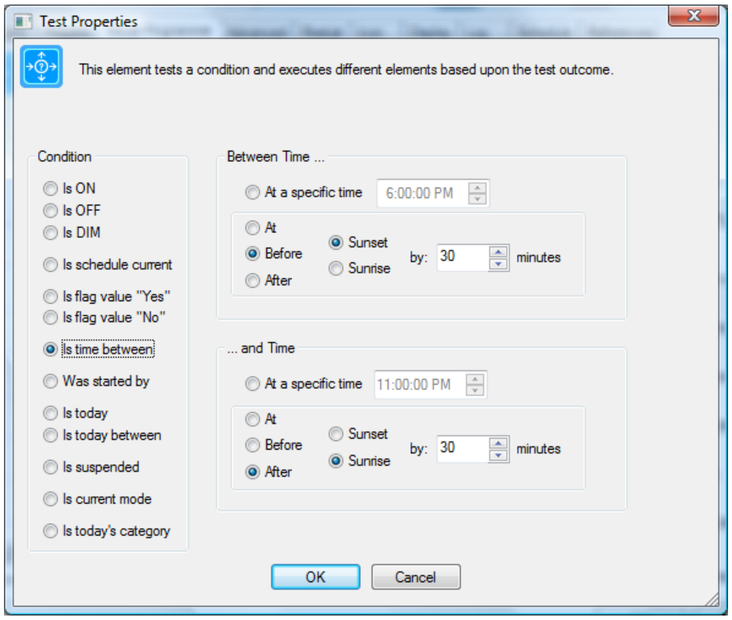
Stop Program

This element stops a running program. If the selected program is not running when this element is executed, it has no effect. If multiple copies of the program are running (see the section on advanced program properties), all copies of the program are stopped.

Hint: You can’t stop the program that contains the stop element. To do that use an *Exit* element instead.

Test

The *test* element is the most complex element HCA contains. Without the test element, each program would start with the “Start here” element and continue from element to element following the connecting lines until the last element has been executed. With the test element, your program can execute different elements based upon conditions that are present when the program runs.



Test Properties

This element tests a condition and executes different elements based upon the test outcome.

Condition

- Is ON
- Is OFF
- Is DIM
- Is schedule current
- Is flag value "Yes"
- Is flag value "No"
- [✓ Is time between]
- Was started by
- Is today
- Is today between
- Is suspended
- Is current mode
- Is today's category

Between Time ...

At a specific time 6:00:00 PM

At [✓ Before] After — [✓ Sunset] Sunrise — by: 30 minutes

... and Time

At a specific time 11:00:00 PM

At Before [✓ After] — Sunset [✓ Sunrise] — by: 30 minutes

This element dialog box has two parts. First click the button to set the condition you want to test for. Second, select the object that you want to test.

Here is a list of conditions that this element can be used to test:

Test choice	**Tests for**
Is On	Is a device/group on? Is a program running?
Is Off	Is a device/group off? Is a program not running?
Is Dim	Is a device/group dim?
Is Schedule current	Is a schedule the current schedule?
Is Variable Value Yes	Is the variable value Yes?
Is Variable Value No	Is the variable value No?
Is Time between	Is the current time between two selected times?
Was Started by	What trigger caused this program to start?
Is Today	Is today a specific day of the week, day of the month, or date?
Is Today Between	Is today between two dates?
Is Suspended	Is the device, program, group, controller or schedule entry suspended?
Is Current Mode	What is the current home mode?
Is Today's category	Is today the selected category as defined by the HCA calendar?

Some conditions are obvious, others are not. Following are more details on each condition.

ON / OFF / DIM

This type of test checks the state of a device, group, or program in your design to see if it's on, off, or dim. One very important point to note is that this test is based upon what state HCA thinks the device is in. For example, if HCA sends an On command to a lamp it records it as On. If you subsequently use a wall switch to turn the light off, and that switch doesn't automatically report its status then HCA may not be informed that the light is off.

When testing for dim you can test to see if the illumination level is less than, greater than, or in a range of levels. If the device supports stored scenes, you can test to see if the device is currently set to some specified scene illumination.

For programs the ON and OFF tests see if the program is currently running or not.

Schedule is Current

This test checks to see if the selected schedule is the current schedule.

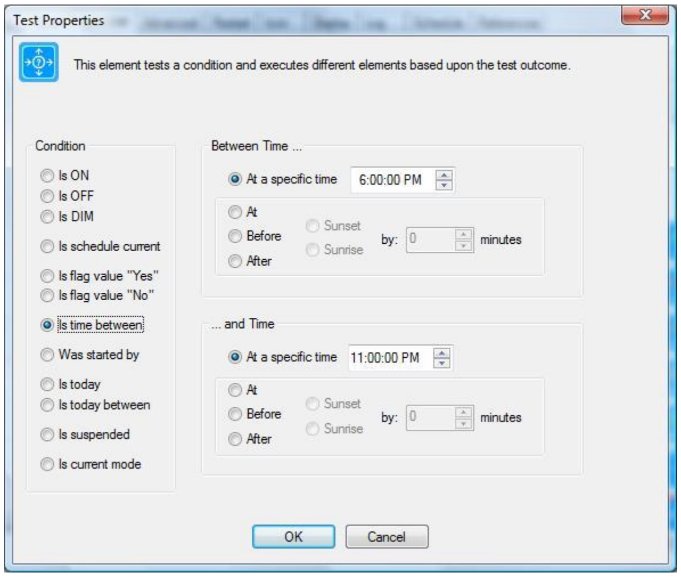
Variable value is "Yes" / Variable value is "No"

This type of test checks the current value of a variable. Since variables can contain Yes and No values, you can test for either of those conditions. Choose the variable begin tested from the dropdown or type in the name of a new variable.

This element only tests variables for simple yes and no values. To test for other values use the *Compute Test* element. If you do use the *Test* element and the variable doesn't contain a simple Yes or No value, the value is converted into a Yes or No value. See the chapter on expressions for more details on this.

Time Between

The *Time Between* test allows your program to perform different actions depending upon the time of day when the *Test* element is executed. This condition is a little more complex and allows you to specify several different settings.


Test Properties

This element tests a condition and executes different elements based upon the test outcome.

Condition: Is time between [✓]

Between Time ...: At a specific time [✓] 6:00:00 PM

... and Time: At a specific time [✓] 11:00:00 PM

You specify the two time points for this test in a manner very similar to the way that you specified time in the Schedule Entry wizard.

In the *Between Time* area, set the **start point** for this test:

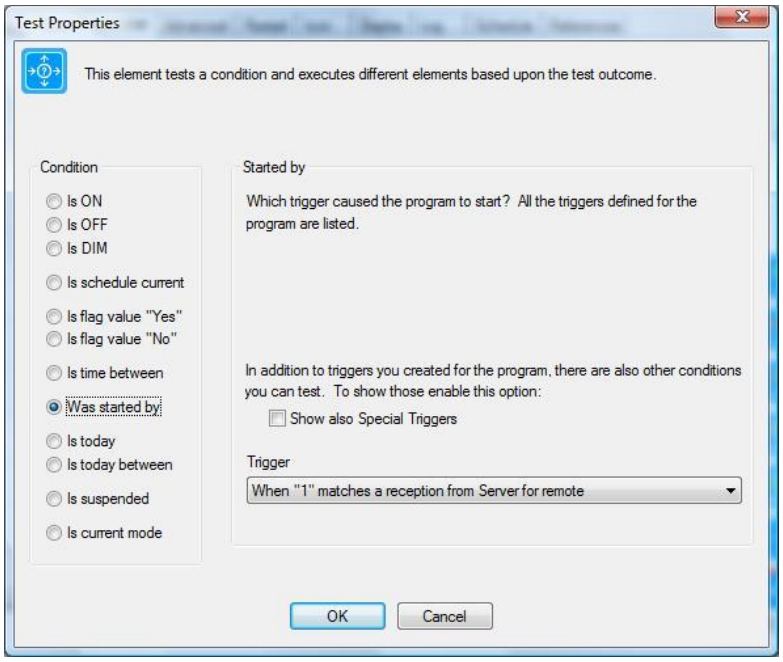
1. Click the button of the time you want to use and specify either the specific start time (like 6:00 a.m. or 5:45 p.m.), or a number of minutes before or after sunrise or sunset.

   In the *…and Time* area, set the **end point** for this test:

2. Click the button of the time you want to use, and specify either the specific end time, or a number of minutes before or after sunrise or sunset.

Started by

This type of test is to check how the program was started. Programs can be started by many different types of triggers. These are specified on the Triggers tab of the program properties.


Test Properties

This element tests a condition and executes different elements based upon the test outcome.

Condition
- Is ON
- Is OFF
- Is DIM
- Is schedule current
- Is flag value "Yes"
- Is flag value "No"
- Is time between
- [✓ Was started by]
- Is today
- Is today between
- Is suspended
- Is current mode

Started by

Which trigger caused the program to start? All the triggers defined for the program are listed.

In addition to triggers you created for the program, there are also other conditions you can test. To show those enable this option:

☐ Show also Special Triggers

Trigger: When "1" matches a reception from Server for remote

The Special Trigger option provides some additional options in the trigger list you can test for. These are:

- Started by any trigger
- Started by a schedule
- Started by user action. That is, by the user using the User Interface to start the program.
- Started by an ON or OFF program element
- Started by a StartProgram element
- Started by group membership

These tests don't compare specific triggers but rather the action that started the program. In this way you could do different things if the program was started, for example, from a schedule and when it starts because you selected the program icon in the user interface and selected ON from the popup menu.

Is Today

This type of test allows you to see if today (the current date when the program is running) is a specific day of the week or month. The same date options used in schedule entries are used here.

Is Today Between

This type of test checks to see that the current date is between two dates without checking the year – only the day and month. In the properties of this test both dates are selected.

Is Suspended

This test type checks to see if the selected device, program, group, controller, or schedule entry is suspended. In order to suspend a schedule entry it must have a name. Suspend is described in the usre guide chapter on schedules and schedule entries.

Is Mode

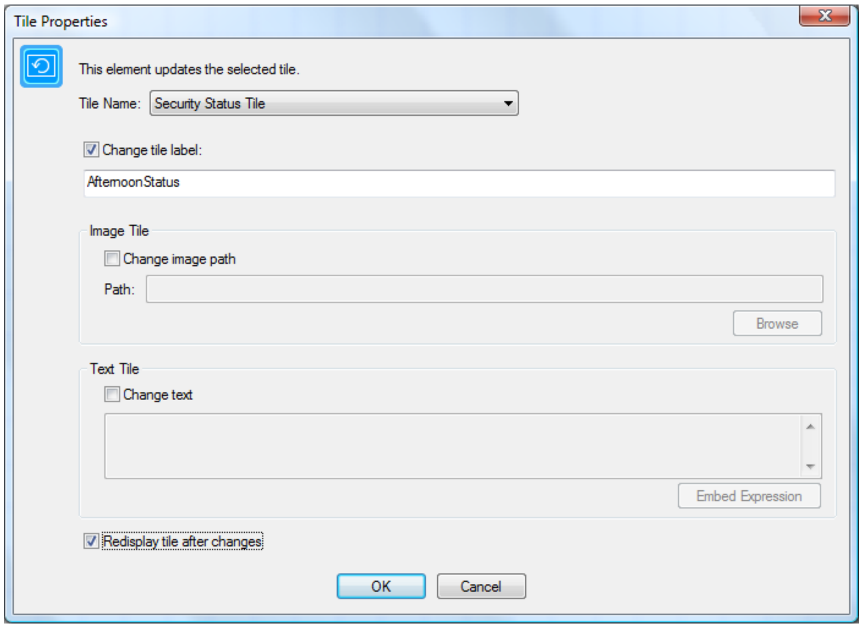
Is the current home mode the selected mode? The home modes are described in the "Home modes" user guide chapter.

Is Today's category

Is today the selected category? Each day of the year can be assigned to a category using the HCA Calendar. The Calendar is described in the "Calendar" user guide chapter.

Update Tile

The *Update Tile* element causes a selected tile on a tiled display to refresh with new information.


Tile Properties

This element updates the selected tile.

Tile Name: Security Status Tile

[✓] Change tile label:

AfternoonStatus

Image Tile

[ ] Change image path

Path:

Text Tile

[ ] Change text

[✓] Redisplay tile after changes

Select the name of the tile to update and what should be updated. For all types of tiles the tile label – displayed at the lower right of the tile – can be updated. For a image tile the image can be changed and for a text tile the text can be changed.

As in other elements – Add to Log, Show Message – you can embed expressions in the text between %%. Before the tile is updated the expressions are evaluated and the final text shows in the tile or the tile label.

Tiles can also be updated based upon a update timer that can be set in each tile. This is an alternative way to cause the update immediately upon this element executing.

In client-server the update is broadcast to all clients.

Variable Set

The Variable-Set element can be used as an alternative to the Compute element. While the Compute element can perform many operations, it is text based. The Variable-Set element is more structured.

The various options let you assign to a variable an:

- Number
- String
- Current Time
- Device Level – the percent the device is at
- Variable – assign from one variable to another
- Expression – An expression like what would go into the Compute element

Variable Test

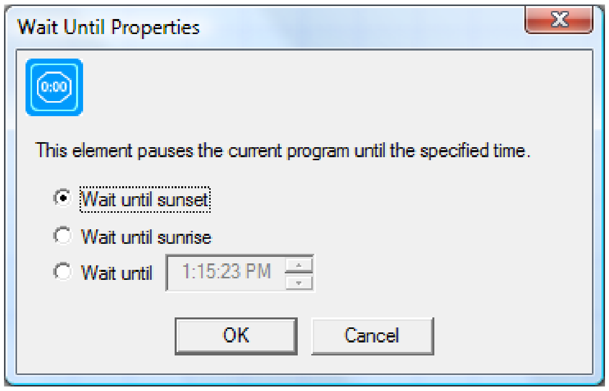
The Variable-Test element can be used as an alternative to the Compute-Test element. While the Compute-Test element can perform many operations, it is text based. The Variable-Test element is more structured.

The tests that can be performed are equal, not equal, less than, greater than, less than or equal to, greater than or equal to.

The is one other option "elapsed time" where you specify a number of hours, minutes, seconds. The variable selected is assumed to contain a date-time and this option checks to see if the current time is >= the time in the variable plus the supplied HH:MM::SS

Wait Until

The Wait Until element is like the delay element in that it causes the program to pause for a while. But unlike the delay element, it does not wait a specified *interval* of time (like 5 or 10 minutes), it waits until a specific *selected* time (like sunrise, or 5:45 p.m.).


Wait Until Properties

This element pauses the current program until the specified time.

[✓ Wait until sunset]
Wait until sunrise
Wait until 1:15:23 PM

There are three different types of *Wait* elements. Click the button corresponding to the type of wait that you want.

1. You can set the program to wait until sunset.
2. You can set the program to wait until sunrise.
3. You can wait until a specific time like 4 p.m., or 6 p.m. If the program starts after the wait time that you enter, the program waits until the selected time on the next day.

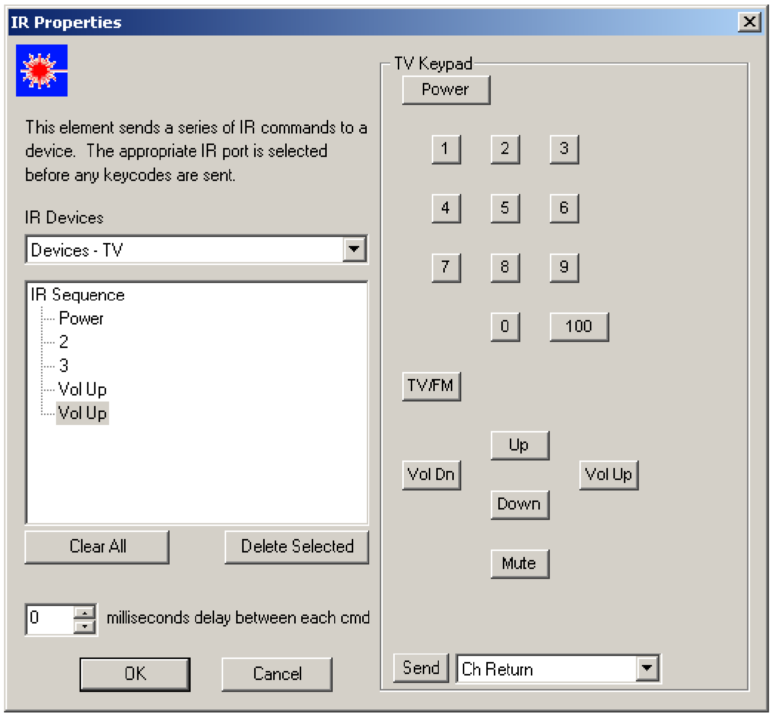
Hint: If you find yourself using the Wait element then you may want to re-think your design. It is almost always better to not use the Wait element but rather break your program into more than one part and schedule them to start at the times you want.

Specific Hardware Elements

These elements are for specific hardware that you may or may not have.

IR

The IR element is used to send sequences of IR commands to a device. This element can only be used if you have an interface that supports sending IR.



This element sends a series of IR commands to a device. The appropriate IR port is selected before any keycodes are sent.

IR Devices: Devices - TV

IR Sequence
- Power
- 2
- 3
- Vol Up
- Vol Up

0 milliseconds delay between each cmd

Ch Return

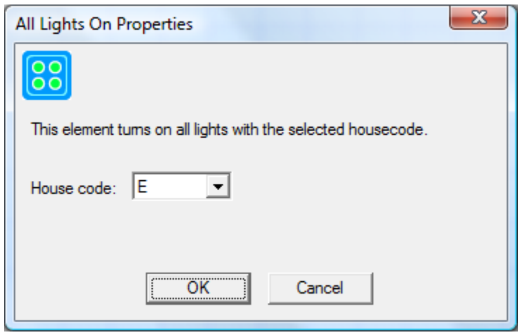
This dialog contains a number of parts. In the *IR Devices* list are all the devices you have setup that use IR. In the right side of the dialog is shown the keypad associated with the device. As explained in the appendix on IR interfaces, each IR device can have its own keypad.

To enter an IR sequence, push the buttons for the signals that you want to send. They are displayed in the IR Sequence tree. New keycodes are entered after the currently selected item. To delete the keycodes previously entered, select then and press the Delete button.

Depending upon the receiving device it may be necessary to space out the IR commands. Enter in a time (in milliseconds) to delay between each key sent.

X10 All lights on / X10 All lights off / X10 All units off

These three operations have the same properties. What each does is obvious. Their properties dialog boxes are the same.



This element turns on all lights with the selected housecode.

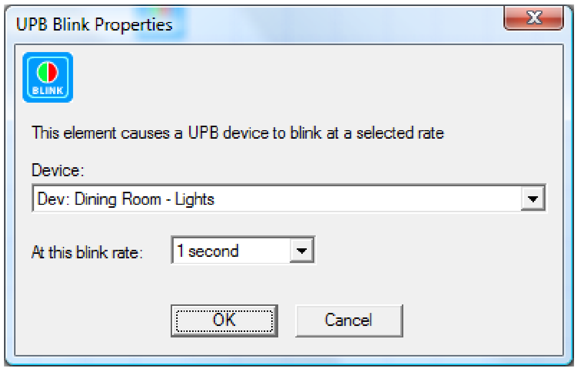
House code: E

Thermostat and Thermostat Test

These elements apply to thermostats and are described in the appendix on Thermostats.

UPB Blink

The *UPB Blink* element sends the UPB Blink command to a device.



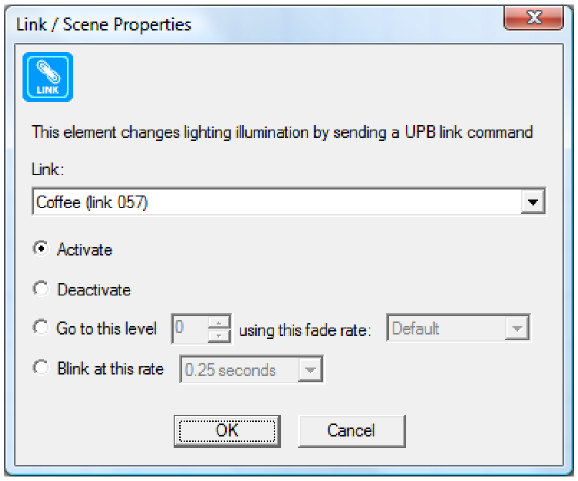
This element causes a UPB device to blink at a selected rate

Device: Dev: Dining Room - Lights

At this blink rate: 1 second

UPB Link

The UPB Link commands send a UPB link command



Link / Scene Properties

This element changes lighting illumination by sending a UPB link command

Link: Coffee (link 057)

[✓ Activate]

Deactivate

Go to this level 0 using this fade rate: Default

Blink at this rate 0.25 seconds

When you Activate a link, any device with that link in its receive components table responds to the preset level and rate stored in the device configuration.

When you Deactivate a link, any device with that link in its receive components table goes off. Doesn't matter what the level in the Receive Components table is, but it does use the rate when going off.

When you dim a link you provide a level and rate. All the device does is to say "Is that link in my receive components table?" and if so it responds to the dim as the command specifies.

Weather Test

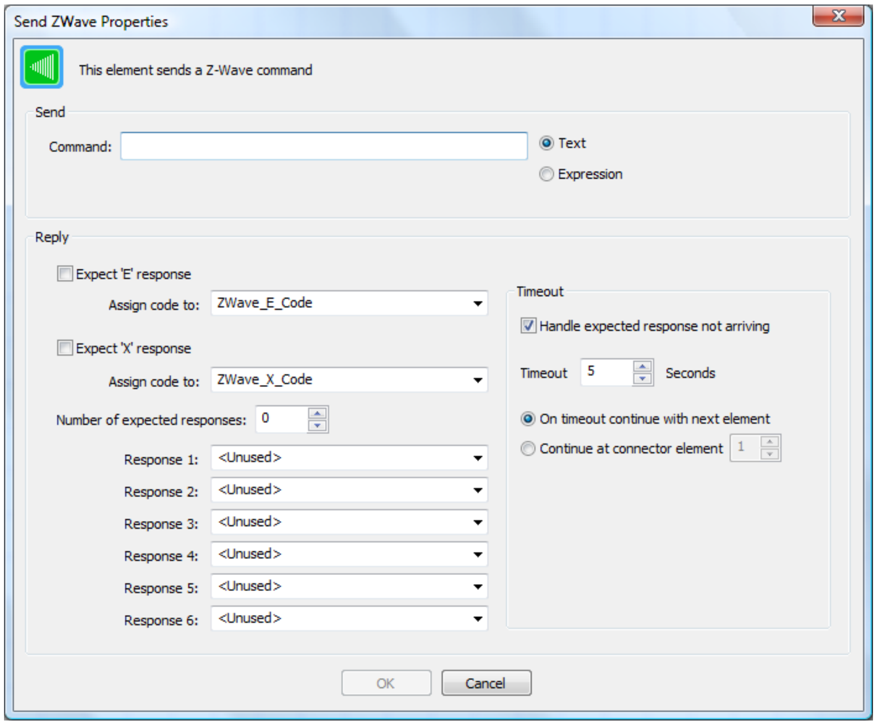
The weather test element is used to read and test data from a weather station. It is described in the appendix on weather stations.

Send X10

The SendX10 element is used to send a stream of X10 commands. Unlike the ON, Off, Dim elements, which send commands to control devices, the SendX10 element can send an arbitrary sequence of X10 commands. This would probably be used in only special circumstances.

Send ZWave

The *Send ZWave* element is very similar to the Port I/O element except that it handles the special reports generated by the Zwave interface as well as the send text and receive text.


Send ZWave Properties

This element sends a Z-Wave command

Send

Command: [✓ Text] Expression

Reply

☐ Expect 'E' response

Assign code to: ZWave_E_Code

☐ Expect 'X' response

Assign code to: ZWave_X_Code

Number of expected responses: 0

Response 1: <Unused>

Response 2: <Unused>

Response 3: <Unused>

Response 4: <Unused>

Response 5: <Unused>

Response 6: <Unused>

Timeout

☑ Handle expected response not arriving

Timeout 5 Seconds

[✓ On timeout continue with next element]

Continue at connector element 1

The only difference in the properties of this element and the Port I/O element is the handling of the 'E' and 'X" responses. To use this element you must understand the syntax and expected responses of the Zwave protocol.

Hint: There is a Zwave technical note available on the support web site that might help get you started with this.

Legacy Elements

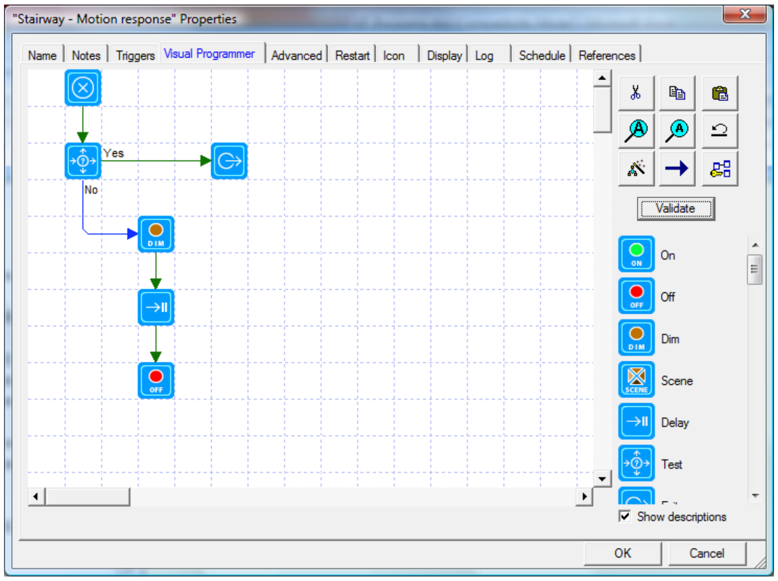
There are additional Visual Programmer elements for Legacy hardware support. These are detailed in the User Guide appendixes for those interface types.

Constructing programs

Now that you know about all the elements used in constructing programs, and the general methods of using the Visual Programmer, you are ready to learn how to draw *Test* and *Repeat* elements.

Test

Here is a sample program that uses a *Test* element.



Notice that from the *Test* element there are two connecting lines drawn: one goes to the Dim element, the other to the Exit element. The connecting line to the Exit element is labeled "Yes," the other "No." When you draw the *Test* element and draw connecting lines from it to other elements, you need to specify which connecting line to follow if the test succeeds or fails.

1. Right click one of the elements directly connected to the test element.
   The popup menu contains two additional options: "Do when Test succeeds," and "Do when Test fails."
2. Pick one of these options, and the connecting line from the *Test* element to the selected element is labeled appropriately.
3. Since you have labeled one path, HCA now labels the other.

   If you later decide that you have the paths incorrectly labeled, you can re-label them by using the same method.

In this example, assume that the test element has its properties set so that when it executed it does: Test variable "Web is on" for Yes.

- If the variable "Web is on" has the value *Yes*, then with the paths labeled as in the above graphic, the next element executed after the *Test* element is the *On* element.
- If the variable "Web is on" has the value *No*, then the next element executed after the *Test* element is the *Off* element.

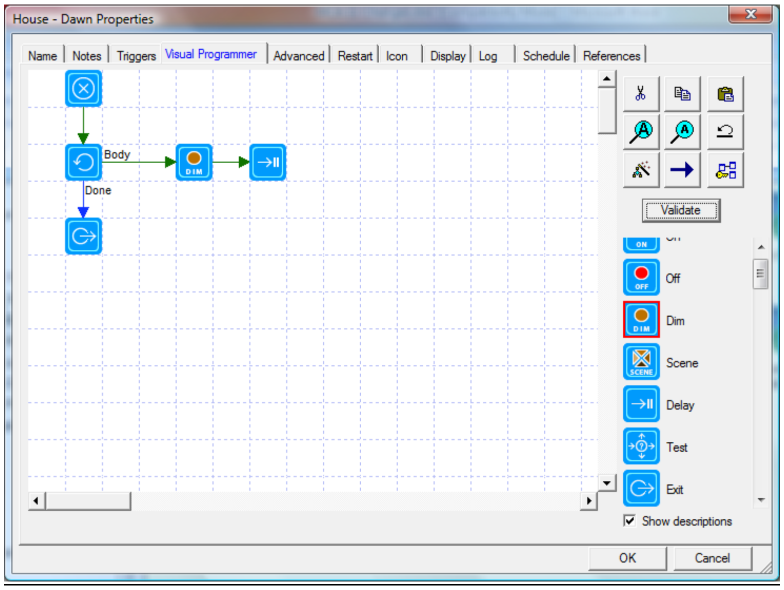
The *Exit* element comes in handy if you have nothing to do in one path from the test. Since there is nothing else for the program to do, it stops running. This is what the *Exit* element is designed for.

A program ends when it has executed an element that has no connecting line leading to another element, or when an *Exit* element is executed.

Hint: All the test elements work in this way, that is, they have two paths that lead out from the element. If the test succeeds, the next element executed is on the Yes path, otherwise on the No path. The Weather Test, Thermostat Test, MM input test, and others, work like this.

Repeat

The *Repeat* element is another complex element. Here is a sample program that uses the *Repeat* element:



Notice that there are two connecting lines drawn from the *Repeat* element. The one labeled “Body” specifies which path to use for the elements to repeat. In this example, the Dim and Delay elements are to be executed repeatedly.

Like the *Test* element, you place the labels on the connecting lines by using the popup menu from either of the elements connected to the *Repeat* element.

In this example we also needed to use an *Exit* element, since there was nothing we wanted to do when the repeat was finished.

Hint: Note that in the above picture that the elements that are repeated are not connected back to the Repeat element. To do so would cause your program not to function as you expect it.

The Exit element has a special use when used in conjunction with the Repeat element. The repeat element can do one of two things. It can:

- Terminate the sequence of elements that make up the body of the repeat and continue the repeat with the next iteration
- Terminate the repeat completely. That is, the next element executed will be the first element in the *Done* path.

Which action the Exit element takes depends upon the setting in the HCA Properties Visual Programs tab.

The Validate button

After you have added all the elements you need in your program and connected them with connecting lines, use the Validate button to perform a simple check. The validate operation can't guarantee that your program does what you want, but it does check for the most obvious errors. It will check to see if:

- All elements are connected to each other
- *Test* and *Repeat* elements have two paths from them and are both labeled
- All elements have their properties set
- Other tests to see if your program is correctly constructed

If you don't use the Validate button, the validate operation is automatically done when you change to another tab in the property dialog or attempt to close the dialog with the OK button.

The first thing that the Validate operation looks for is to make sure that all elements are connected together. That is, there are no elements which can't be reached from the Begin Here element by some path. Normally this is a good test. But you can instruct the program validation to not check for this. There is an option on the HCA Properties Visual Programs tab to set the behavior you want. It can be useful when you are developing programs to cut sections of the program off but not to remove them from the programming canvas.

Troubleshooting: Getting programs to do what you want

Drawing programs is the easy part. HCA makes it easy to draw your program, and check it. Getting the programs to do what you want is a bit trickier. This section discusses some ways that you can check how your programs are working. **The next chapter discusses the Debugger tool that can really help a lot**.

When you create a program, think very carefully about what you want the program to do, and, most importantly, the sequence for the events. You may have a program that turns on a couple of lights. Maybe the order in which you control these lights is not important, but then maybe it is. You need to consider the intent and expected outcome to properly sequence your elements. Once you have a clear idea of what you want to do, you can begin drawing your program.

You may want to start with a simple program and add to it as you refine what it needs to do. You may want to keep a version of your base program that works all right before you improve it. Just make a copy of it, paste the copy into your design with a new name, and work on that one.

When designing your program, you may want to watch for the use of the Delay, Wait, and Repeat elements. These are very useful in the proper circumstances, but they may also be used inappropriately. Remember that a program can be scheduled to start at any time you want—just like scheduling a light to go on at a given time. If you find that you want a program to do a few things, wait, and then do some more things, you may really need two programs. And you may want to schedule them both.

The Delay element is very useful in circumstances where its clear you want to wait, not until some future time, but just for a short pause. For example, turn on some lights, wait a minute then turn them off.

Let's assume you have written your program and it contains some *Test* elements. You schedule the program to start at a certain time and you notice, after that time has passed, that things aren't quite right in your home. How can you tell what the program did, and why it didn't work correctly? You can answer this question by using the Log.

The log is covered in the Tools chapter but the general idea is simple: The log contains an entry for each element in the program as it is executed. Using this you can quickly get a "picture" of what the program did.

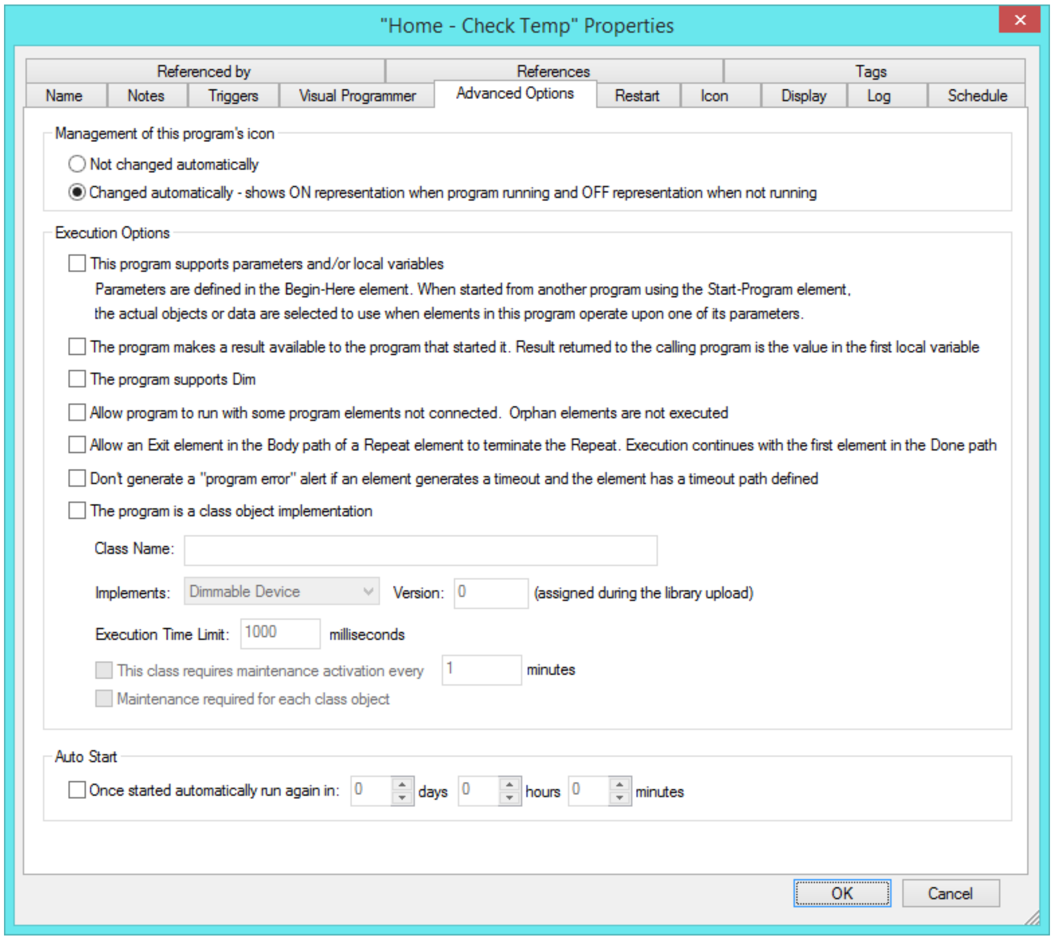
If you want to see only when the program starts and stops, enable the "Log when programs start and stop".

When log entries are made for the *Test* and *Repeat* elements, a note is made of which path is taken from those elements.

You can also use the *Log* element to add your own messages to s log.

Program properties Advanced Options tab

The Advanced tab of the program Properties dialog box has several options


"Home - Check Temp" Properties

Referenced by | References | Tags
Name | Notes | Triggers | Visual Programmer | Advanced Options | Restart | Icon | Display | Log | Schedule

Management of this program's icon
- ○ Not changed automatically
- ● Changed automatically - shows ON representation when program running and OFF representation when not running

Execution Options
- ☐ This program supports parameters and/or local variables
  Parameters are defined in the Begin-Here element. When started from another program using the Start-Program element, the actual objects or data are selected to use when elements in this program operate upon one of its parameters.
- ☐ The program makes a result available to the program that started it. Result returned to the calling program is the value in the first local variable
- ☐ The program supports Dim
- ☐ Allow program to run with some program elements not connected. Orphan elements are not executed
- ☐ Allow an Exit element in the Body path of a Repeat element to terminate the Repeat. Execution continues with the first element in the Done path
- ☐ Don't generate a "program error" alert if an element generates a timeout and the element has a timeout path defined
- ☐ The program is a class object implementation

Class Name:

Implements: Dimmable Device   Version: 0 (assigned during the library upload)

Execution Time Limit: 1000 milliseconds

- ☐ This class requires maintenance activation every 1 minutes
- ☐ Maintenance required for each class object

Auto Start
- ☐ Once started automatically run again in: 0 days 0 hours 0 minutes

OK   Cancel

Icon Option

The first of these options, changing the icon, is discussed earlier in this chapter (see Change icon).

Supports Parameters

As described in the section above on parameters, this is the option that enabled is the program can be started with parameters. Until this option is enabled the Begin-Here element can't be configured to define what the parameters are. When this option is enable, Begin-Here element's properties can be opened and local variables given names as well as parameters.

<u>Program crates a result</u>

If this option is enabled and the program is started from another program using the Start-Program element, then whatever value is assigned to the this programs first local variable is copied into a variable in the program that executes the Start-Program element.

<u>Program supports Dim</u>

If this option is enabled, then the program can operate like a device and can be controlled “on”, “off” and set to a level (0% to 100%). This is done using parameters and is fully described in a technical note on “dimmable programs”.

<u>Orphan elements OK</u>

Normally when completing the editing of program, HCA checks to see that the program is constructed correctly and one of those checks looks for elements that are not connected to other elements (we say those are “orphaned”). If such elements are found then the program can’t be started. If this option is enabled, the program with orphan elements can be started and those orphaned elements just are not executed.

<u>How exit operates</u>

The repeat element, like a decision element, has two paths that come out of it. One, called the “body” path, is the set of elements that are executed the number of times given in the repeat element’s properties. The other path, called the “done” path, is the where execution goes next when the elements in the body path have been executed “count” times. When this option is enabled then an Exit element executed in the body of the repeat causes the Done path to be taken next. When this option is not enabled, the Exit element when executed just starts the Done path again.

<u>Program errors on timeout</u>

When an element whose action can “time out” when communicating to a hardware device and expecting a response, the element has configuration that says what to do on a timeout. One option is just to continue with the next element. The other option handles the timeout by changing execution to another location. In the first case, just going on to the next element, HCA notes in the log that the timeout happened. In the second case, no log note is made because the program “handled” the error.

<u>Class implementation</u>

Class programs are a way for programs to extend the range of devices that HCA supports. This is an advanced topic. For more information about the class options on the program’s Advanced Options tab, see the Class Program technical note.

<u>Auto Start</u>

The Auto Start settings provide a convenient way to start the program again after it completes. This makes it easy to have a program that runs “every hour” and not have to add it to the schedule 24 times.