



HCA 14 Changes (as of 22-Sep-2017)

NOTE: Don't assume any priority or importance of these items just by the placement in the list. The order has no importance.

- [HTTP VP element](#)
- ["Flags" to "Variables"](#)
- [Program local variables](#)
- [Visual Programmer new features](#)
- [Camera support](#)
- [Server Find Button on C/S tab](#)
- [HTML Folder Copy](#)
- [HCA Keypads](#)
- [New "All" Displays](#)
- [HCA Framework](#)
- [Command Line options](#)
- [Phillips Hue](#)
- [JSON Expressions](#)
- [User Implemented Device Classes](#)
- [Periodic Background Poll](#)
- [Device Control Improvements](#)
- [Device Control notification](#)
- [Icon Marking](#)
- [Changes to Alert Manager](#)
- [Changes that impact X10](#)
- [Components Removed / transferred to Legacy](#)
- [HCA Cloud](#)
- [New Icons](#)
- [Change to Suspend](#)
- [Change to Security](#)
- [Miscellaneous](#)

New VP Element

HTTP: Ability to do a HTTP POST, PUT, GET and capture the result

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

☐ Save reply to variable

Timeout

Timeout 2 Seconds

☒ On timeout continue with next element

☐ Continue at connector element

☐ 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.

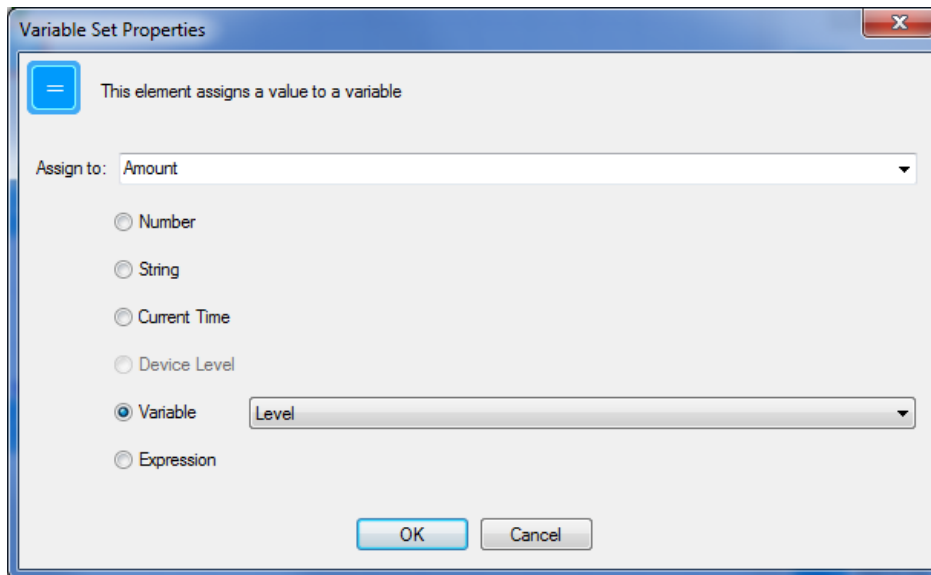
The reply can be ignored or captured to a variable or a file. If a reply is expected then a timeout in seconds can be specified. If the operation doesn’t receive the reply in the specified time the next element to execute is specified. The element doesn’t check for the result from the operation. If there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

“Flags” now “Variable”

Starting in this version the term “Flag” has been replaced by “Variable”. Variables are the same un-typed objects as before that can hold boolean, number, string, date-time, or date-time-span data.

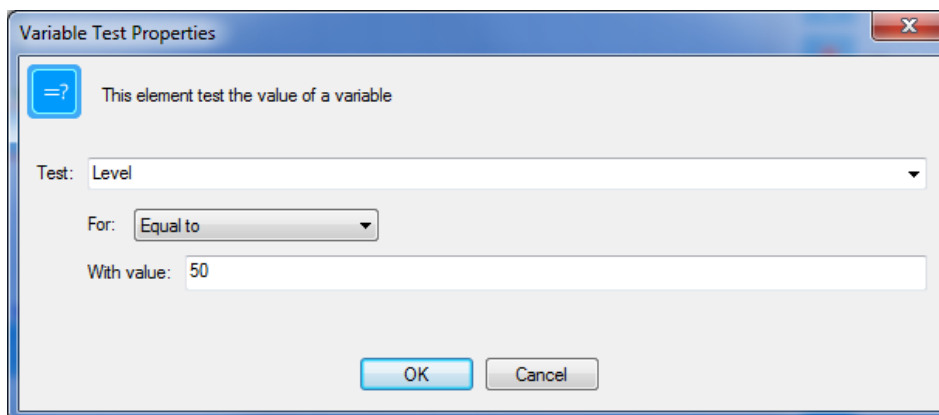
New VP elements were added to set a variable to a value and to test a variable. The same operations can be done in the Compute and Compute-Test elements but these are more structured and less text oriented.

New VP “Variable Set” element



There are options for the source of the value: Number, String, the current time, the level a device is at, another variable, or an expression.

New VP “Variable Test” element



The value can be any expression not just a simple value. The rules for testing are the same as for the Compute Test element.

Local Variables

Starting in V14 a program can now have “local” variables. These are variables that have a value only during the execution of the program. They don’t appear in the Variable Inventory dialog.


Before a program can have local variables, an option must be enabled for that on the program properties “Advanced” tab.

Local Variables

☒ This program supports local variables

Once this is done then the Begin-Here element has properties that can be modified:

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 when the element executes the actual object or value by the starting program is used.

Parameters

Parameter count:

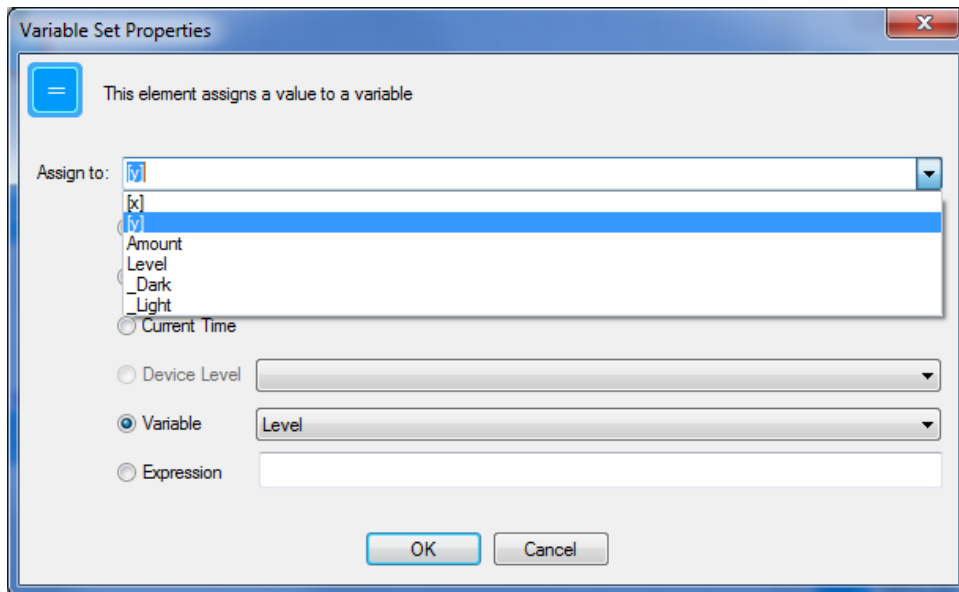
Local variables:

Name: <input type="text" value="duck"/>	Name: <input type="text"/>
Name: <input type="text" value="web"/>	Name: <input type="text"/>
Name: <input type="text" value="sis"/>	Name: <input type="text"/>
Name: <input type="text"/>	Name: <input type="text"/>
Name: <input type="text"/>	Name: <input type="text"/>
Name: <input type="text"/>	Name: <input type="text"/>

OK Cancel

The names of the local variables are entered here. A program can have up to 12 local variables. Their names can be any valid variable name.

In program elements that operate on variables you can choose as that variable either a global variable name or a local variable name. In the variable dropdown the local variables appear first enclosed in square brackets. For example:



If a local variable has the same name as a global variable you could choose either and HCA doesn't get the two confused.

The elements where you can use a local variable are:

- Var-Test
- Var-Set
- Make Flag Yes
- Make Flag No
- Not Flag
- Test
- ZWave
- Serial
- HTTP
- Camera

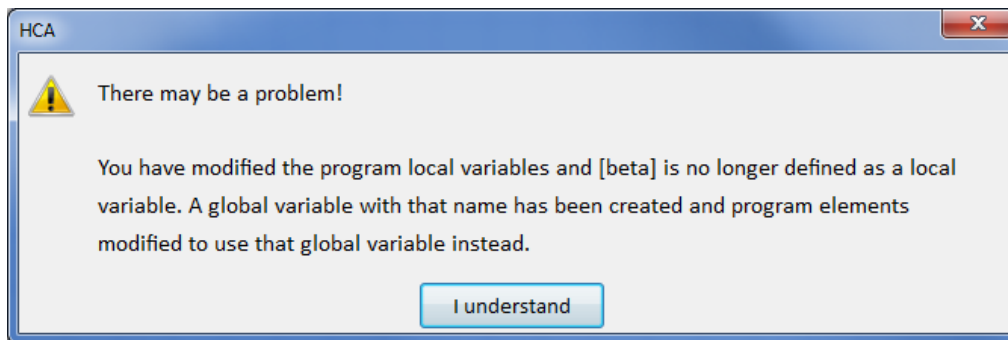
The Request-Input element saves its result in a variable but that variable must be global.

The Compute and Compute-Test elements also can reference local variables. They are named and used like any other variable. In parsing and evaluating the expression if a local and a global variable have the same name, the local variable is used. There is no way to access the global variable in this case.

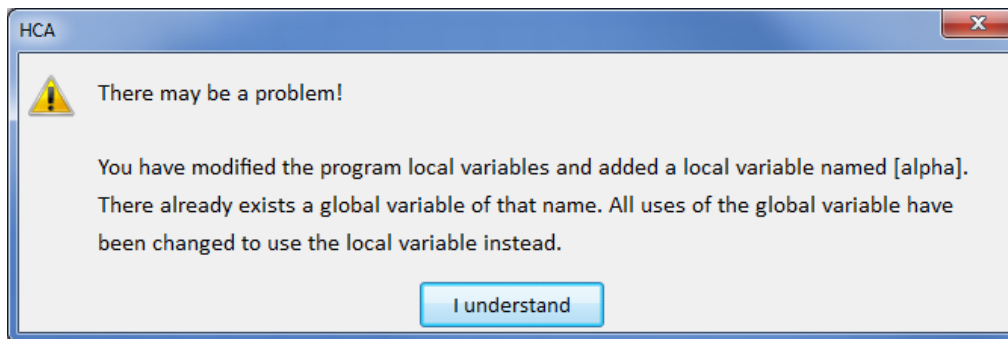
[How the visual programmer handles local and global variables](#)

When the Begin-Here element is edited and local variables modified – a new one added or an existing one removed, the Visual Programmer looks through all the elements in the program and adjusts the program element properties to make them consistent with the changes.

If an element referenced a local that has now been removed, a global variable is created if necessary and the element changed to use that variable.

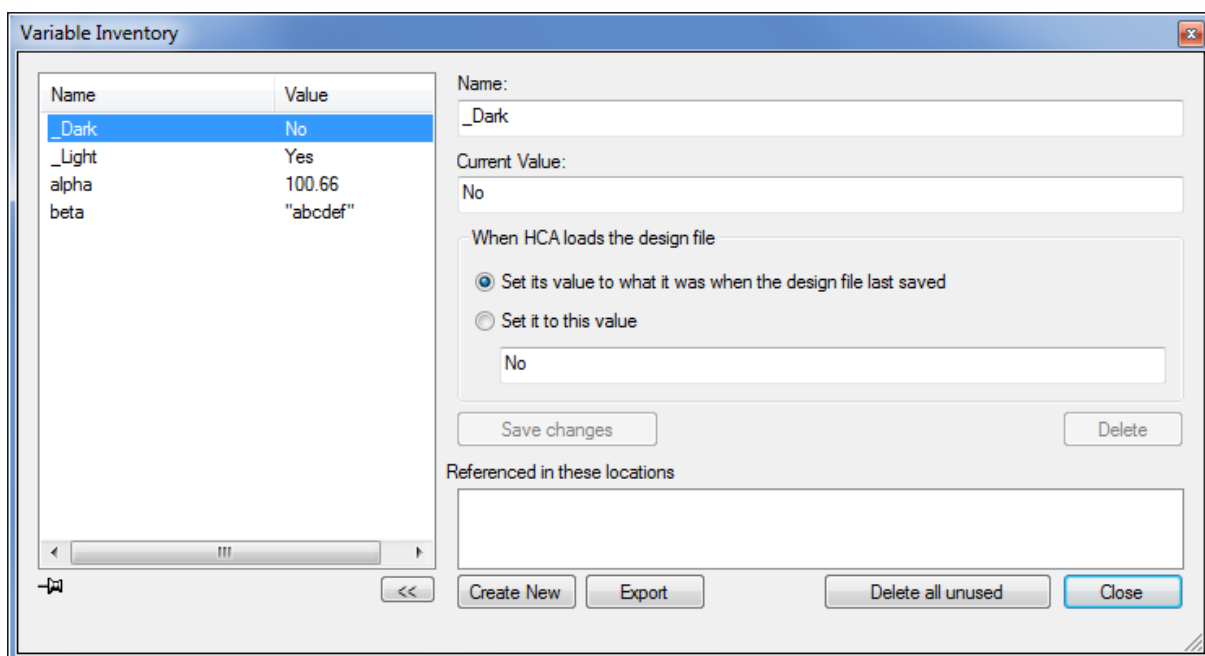


If an element referenced a global variable and a local variable was added with the same name as that global variable then the element is updated to use that local variable instead of the global one. A global variable with that name still exists but you can remove it from the design using the Variable Inventory dialog if it has no other uses.



Variable Inventory

The Variable Inventory, in previous HCA versions, had some strange aspects and changes have been made.

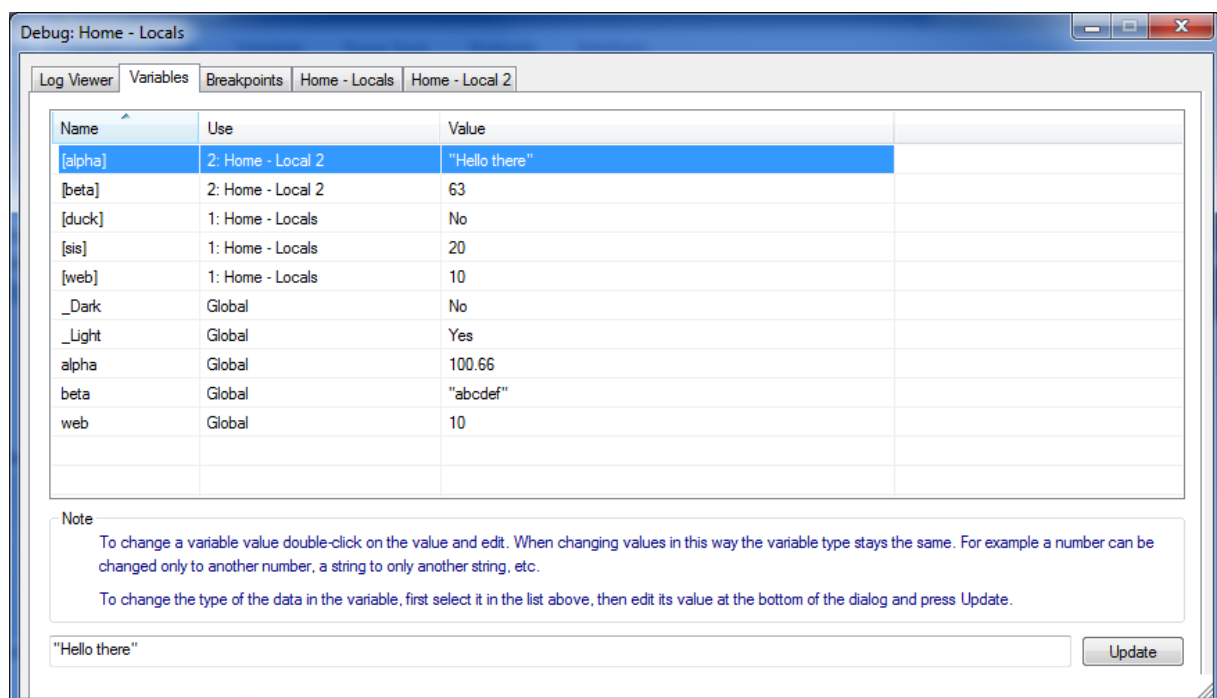


A change was made in how it handles updating variable name or values. When a value is entered into the current value or startup value edit control, HCA converts what you typed into a value and makes the assignment to the current variable. Since variables are untyped, the type of data you entered is determined by this procedure.

1. If it can be converted to a number, then the value assigned is numeric.
2. If it is “yes”, “no”, “true”, or “false”, then the value assigned is Boolean.
3. If it is date-time quoted (the quote character is # in this case) and what is between the quotes can be converted to a date-time, then the value assigned is date-time.
4. If it isn’t date-time quoted but what was entered can be converted to a date-time, then the value assigned is date-time.
5. If it is string quoted (the quote character is “ in this case), then the value assigned is string. The start and end quotes are not saved as part of the string.
6. If none of the above, then the value assigned is a string.

In previous versions, you could change a variable name to something invalid or duplicate an existing name. Now it catches those errors. Also there is now (finally!) a “Create New” button to create a new variable.

The debugger has also been updated to handle local variables. On the “variables” tab are both global and local variables. Depending upon where execution stopped, any local variables on stack frames are displayed.



In the “use” column it shows if the variable is global or local and if local what stack frame and program it is a local to.

New Visual Programmer Features

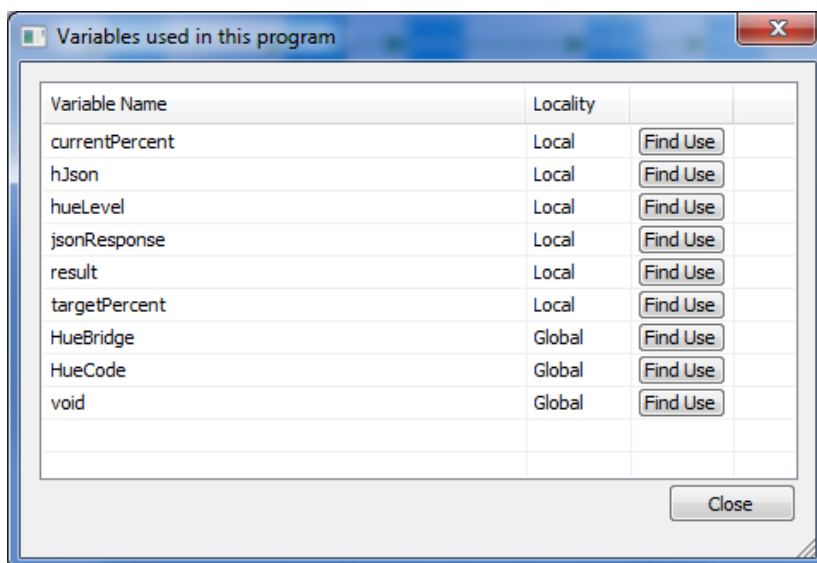
Several new features have been added to the Visual Programmer. The tools panel has changed.



Aside from having better looking icons there are now three new tools.



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

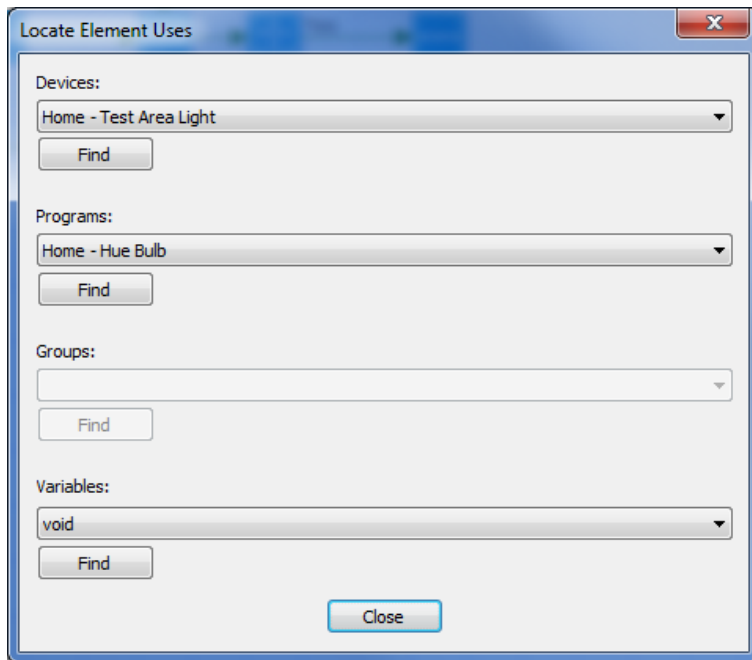


If you press the “Find Use” button associated with a variable, the dialog closes and an element that uses the variable is selected.



These two tools are a “Find” and “Find Next”. When the “Find” tools is pressed a dialog opens that lets you choose the target of the find.

Tip: Keyboard shortcut for find is ctrl-F and for Find-Next is F3.



Select the Device, Program, Group, or Variable to locate and press the *Find* button for that object. An element that uses that object is selected in the programming canvas if one is found. If that object isn't found, then an error popup shows.

The *Find Next* operation moves to the next element that uses the object. It is circular so that Find Next always finds the next even if it must go back to the start of the found objects.

The "Find Use" button in the "Show Variables" dialog ties into this mechanism. That is, the "Find Next" will find the next use of the variable when the Find was started from the Show Variables dialog.

IP Camera Support

Some IP cameras are now supported. Each camera "class" has a definition file that contains the common operations that HCA implements. Initially the Foscam cameras are available – many different companies rebrand these so they are very common.

IP Cameras are added using the new Device Wizard.

New Device Wizard - Step 3 (Home - Camera C)

What type of device are you creating? If the device is not listed select either a similar device or generic device. You can always change the properties of the device later if necessary.

Manufacturer: IP Camera

Kind: All

Model number and description: IP Camera

< Back > Next Finish Cancel

Step 4 of the wizard captures the camera class, address, and access information:

New Device Wizard - Step 4 of 5 (Home - Camera C)

The camera is at what IP address, port, and type?

Camera Type: Foscam

IP Address: 192 . 168 . 15 . 0 Port: 80

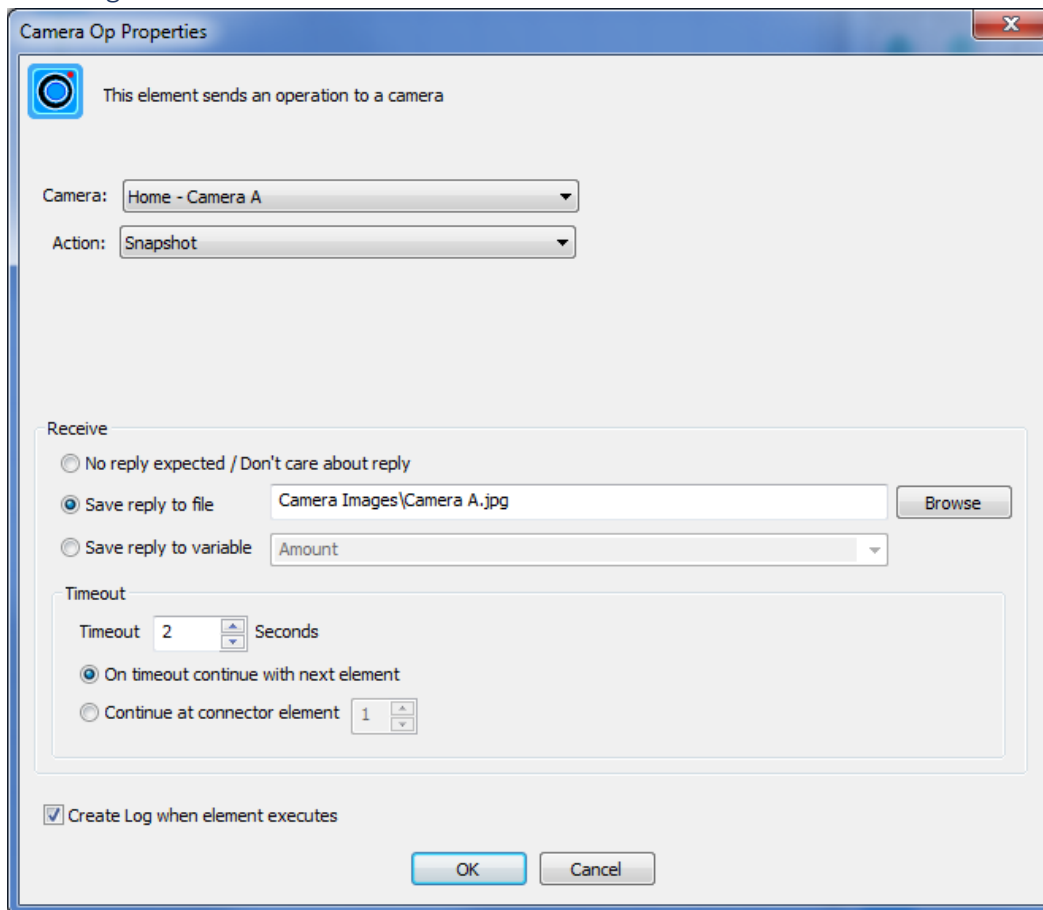
User name: kimberly

Password: kimberly

< Back > Next Finish Cancel

These properties can be modified in the Device Properties dialog on the “Camera Settings” tab. Cameras can only be used in programs. They can’t be scheduled or activated from the user interface.

The Program Camera element



If you are capturing a snapshot image then routing the response to a file is what you want to do. Like the HTTP element you can specify what happens if there is no response from the command.

Some of the actions have additional information that must be supplied. If you select those commands then additional edit controls show where you can enter the needed data.

Server Find Button on C/S tab

On the HCA Options dialog “Client-Server” tab there is now a button that starts an attempt to locate the server if running on a different machine on your internal network.



The operation doesn’t always work on some more secure networks so you may have to fall back to the old method of asking the server computer for its IP address (IPCONFIG at a CMD.exe prompt).

A similar button is now available in the V14 Windows HCA Client.

HTML Folder Copy

HTML displays can be useful for showing complex data in the same way that your favorite browser displays a web page.

A HTML display can be simple in that you supply a URL and when the display is the current display the web page for that URL shows.

A HTML display can be a bit more complex in that the HTML is generated by HCA reading a template file that has placeholders and creating a resultant file that is then displayed.

If you are unfamiliar with this concept please review the Displays chapter of the User Guide.

A HTML file can be complete in itself or it can reference image files and style sheets. If you are working with client-server it will be necessary for you to have those image files and style sheets available on each client. You can manually copy them from the server to the client or HCA can do it for you but for that to work there are some things you need to know.

The first thing you must do is have all the files needed for each HTML display be in its own folder – possibly with sub-folders. These folders are sub-folders of the folder that contains the .HCA design file and relative paths used to locate the HTML files. For example:

Template and result file:

Template file:
Tile Sun Moon\index.htx
View Template Browse

Result file produced from the template:
Tile Sun Moon\index.html
View Result Browse

In this case the folder “Tile Sun Moon” contains the HTML template and result file and a relative path is used to locate it. A “relative path” doesn’t start from the root but the current folder is used from where the path starts. For example, if the path to the loaded design file is:

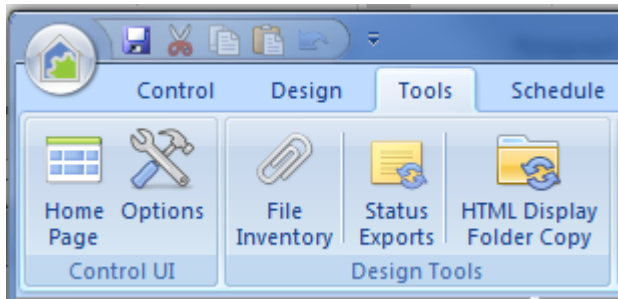
```
C:\users\kimberly\documents\HCA\MyHome.hca
```

Then the path to “index.html” for this display based upon the above example, is located in:

```
C:\users\kimberly\documents\HCA\Tile Sun Moon\index.html
```

HCA when running in client-server mode, locates the template file on the server and after the result file is created on the server it is copied to the client for display. But any other files used by the HTML – style sheet or images – are not also copied at this time.

The Windows client has an operation that for each HTML display it copies the contents of the folder holding the HTML file from the server to the client. A similar operation is now available in HCA. The button for this in the “Tools” ribbon category called “HTML Display Folder Copy”



In addition, a new property has been added to HTML displays.

☒ Template and result file

Template file:

Result file produced from the template:

☐ Display this URL after creating the result file

☒ When loaded by a client automatically copy the result file's folder contents to the client

When the final checkbox in the above picture is ticked then when the Windows client or HCA running as a client connects to the server any HTML display has the contents of its folder copied from the server to the client. This makes images and style sheets available on the client for display.

NOTE: If you use this option it is vital that you follow the recommended practice of creating a sub-folder for each HTML display and using that folder for the files for that single display.

HCA Keypads

If in a HCA Keypad a button is used to start a program and that program supports parameters, you can now specify the arguments to those parameters. For example:

Action: Start program Kitchen - UCQF Control Program args...

Response:

 Button 1: Show On Button 3: Show Off Button 5: No change

 Button 2: Show Off Button 4: Show Off Button 6: No change

Since the selected program has parameters the “Program args” button appears. Pressing it lets you enter the values and objects for that program’s parameters.

UCQF: Dev: Kitchen - Fan Ctrl

 op: 1

OK Cancel

When the keypad button is pressed when in control mode the program is started using those arguments.

New “All” Displays

Like the current “All Programs” and “All Devices” displays HCA can also create and maintain an “All Folders” and “All Rooms” display. These options are set in the HCA Options dialog, “Design Pane” tab.

All Displays

☒ Create “All Devices” display

☒ Create “All Programs” display

☒ Create “All Rooms” display

☒ Create “All Folders” display

In addition, in some places where you selected a display, these displays were not always offered as choices and that has been changed. When one of these displays is deleted, by the normal right-click “delete” or by opening HCA options and un-ticking the box, if the display is referenced in some location you now get info on that before the delete proceeds.

HCA Framework

While mostly benefitting new installations, the HCA framework is now included in the install. This starter design file contains several valuable displays and an overall design that a user can start with and then add these own rooms and devices to. For more information see the updated Getting Started guide.

Command Line

In previous versions of HCA there were command line parameters for setting the HCA Server address but they were inconsistently used, not all the client-server settings provided, and they would overwrite the “normal” ones set in the HCA Options dialog. This is now all corrected. The command line parameters are:

`/HCAServer:###.###.`

Provide the 4 part IP address of the server

`/HCAServerName:<string>`

Provide the DNS name of the HCA Server

`/Port:#`

Provide the port # to use for connections to the server

`/Pswd:<string>`

Provide the remote access password to use during connect

Each of these acts as an override of the same parameter in the HCA Options Client-Server settings. Those settings are not modified using these. If a connection parameter isn't provided on the command line, then that HCA Options parameter is used.

These are very handy when creating desktop shortcuts.

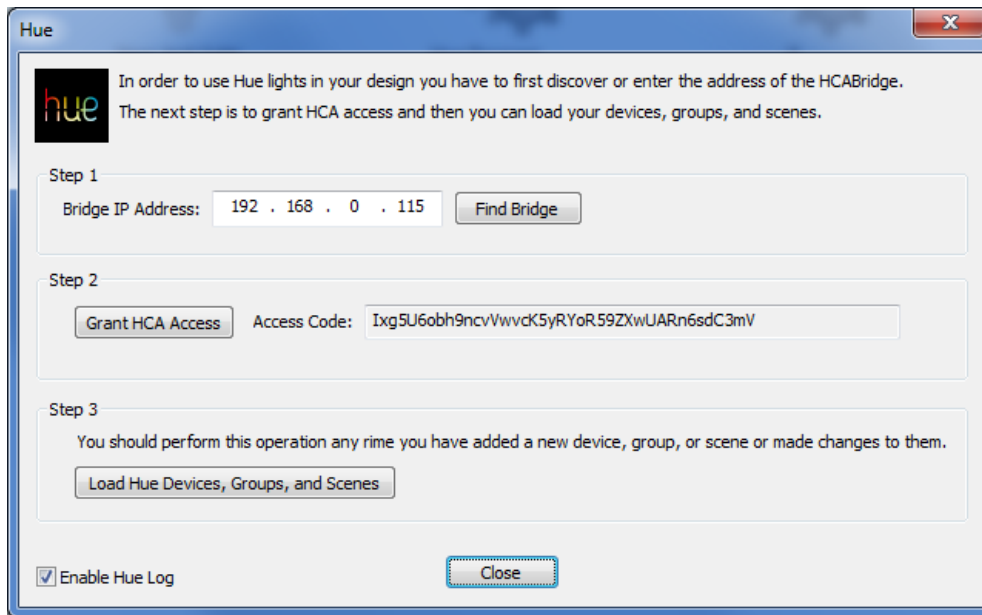
NOTE: These same four settings are also available in the V14 Windows Client application

In addition, you can place on the command line the path to the file to open. This works for HCA and the HCA Server. If any path elements contain spaces, enclose the path in quotes.

Phillips Hue

Support was added for Phillips Hue system. Before HCA can work with Phillips bulbs you must first connect HCA to the Phillips Bridge and then read out the configuration of the Bridge which then adds all your devices to your design. If you are new to Hue get started by using their mobile application [before](#) working with HCA. HCA relies a lot upon the configuration you created there.

To begin the process, use the “Hue” button in the Interfaces ribbon category.



Follow the steps in the dialog. The “Find” operation works only if the Hue Bridge has been powered on for a while. The “Grant HCA Access” asks you to tap the button on the top of the Hue Bridge and then it will authorize HCA and generate an access code which the dialog displays.

Hue has the concept of “groups” which maps closely to the HCA room concept. When devices are read from the Bridge configuration they are placed in rooms of the same name as you put them in using the Hue application. Once the devices are added you can move them or rename them in HCA.

If you make any changes to devices or scenes in the HUE app, you should rerun the “Load Hue Devices” action again and HCA captures any changes or added devices, scenes, or groups. Any existing devices even if renamed or moved in HCA are not duplicated only updated.

Once a device is added it gets a “Hue” tab in the device properties to show the Hue information. All of it is read-only.

"office - Near desk light" Properties

Name Notes Room Type Hue Triggers Restart Icon Display Log Groups Schedule References Power Track Green

Device Id:

Name:

Room:

Manufacturer:

Model:

Type:

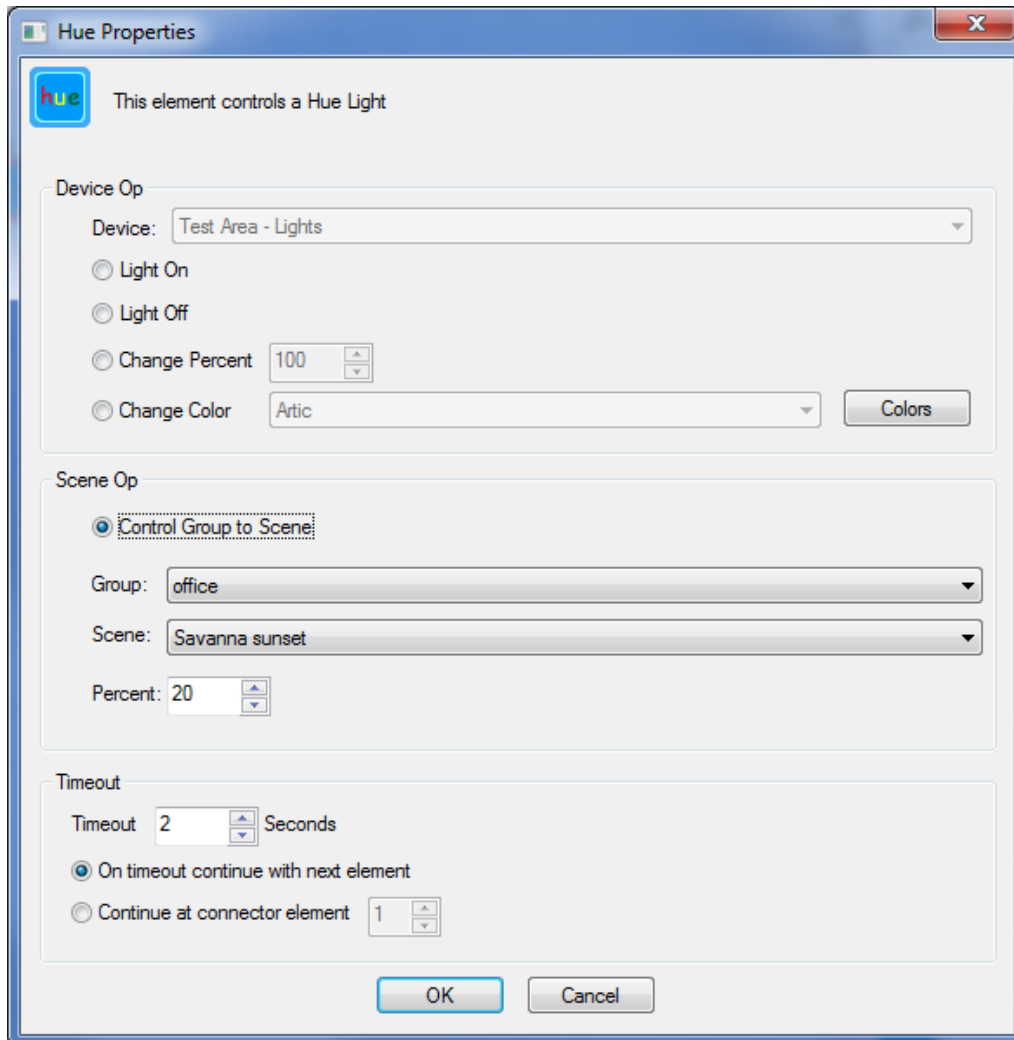
Version:

When Added:

OK Cancel

Hue lights are like lights of any other protocol – you can control them ON, Off, Dim from a schedule or program. Auto Off and the Room operations all work as expected.

New Hue VP element



As you can see in the above image of the Hue VP element, 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:

The first option is to not work with color at all. 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.

The second option, If you want to work with color directly, is that HCA gives you the ability to define, name, and use colors. Press the "Colors" button in the Hue VP element.

JSON Parsing in Expressions

Facilities were added to the expression service for parsing and searching data in JSON format. **This is documented in a separate PDF.**

www.HCATech.com/download/V14/Doc/TechNote_JSONSupport.pdf

User Implemented Device Classes

User Implemented device classes allow support for devices that are not supported in the base HCA product – like X10, Insteon, UPB, and Hue are. A device of a user implemented class is controlled by a user created program. It is assumed that program communicates with the target hardware using the HTTP element or the Generic IP or Generic Serial interfaces using the Port I/O element.

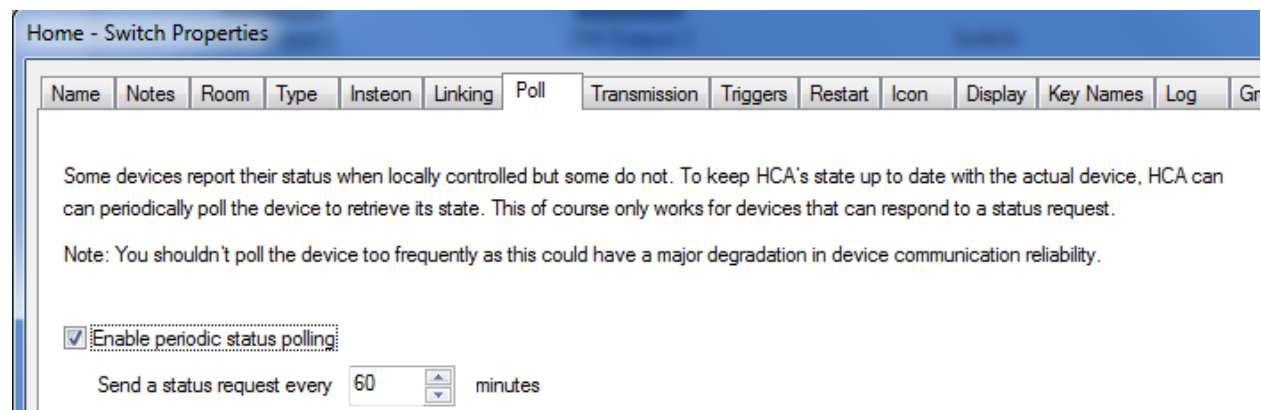
The advantage of this new mechanism is that once the class is added to HCA then all the “regular” facilities in HCA that work on of the built-in types are now available for objects of the added class. For example, a device of a class implemented as a “dimnable device” can be controlled by the UI with the right-click menu, controlled in the control UI and mobile applications with a tap, used in a schedule, controlled by programs using the ON, OFF, DIM, and MULTI elements. Any place a device of a built-in protocol can be used, a device of an added class can be used.

Complete documentation is found in a separate PDF.

www.HCATech.com/download/V14/Doc/TechNote_UserClasses.pdf

Periodic Background Poll

A new device property was added to have HCA in the background periodically poll a device for its status. This is enabled on the “Poll” device properties tab.



You should not rely upon the time to be exact. The polling takes place not less than the time specified but may not occur at exactly the time specified.

When a file is loaded into HCA or HCA Server, the 1st poll happens 'n' minutes after the startup. Where 'n' is the poll interval. If the poll time or enable is changed, the next poll is after the 'n' minutes have elapsed.

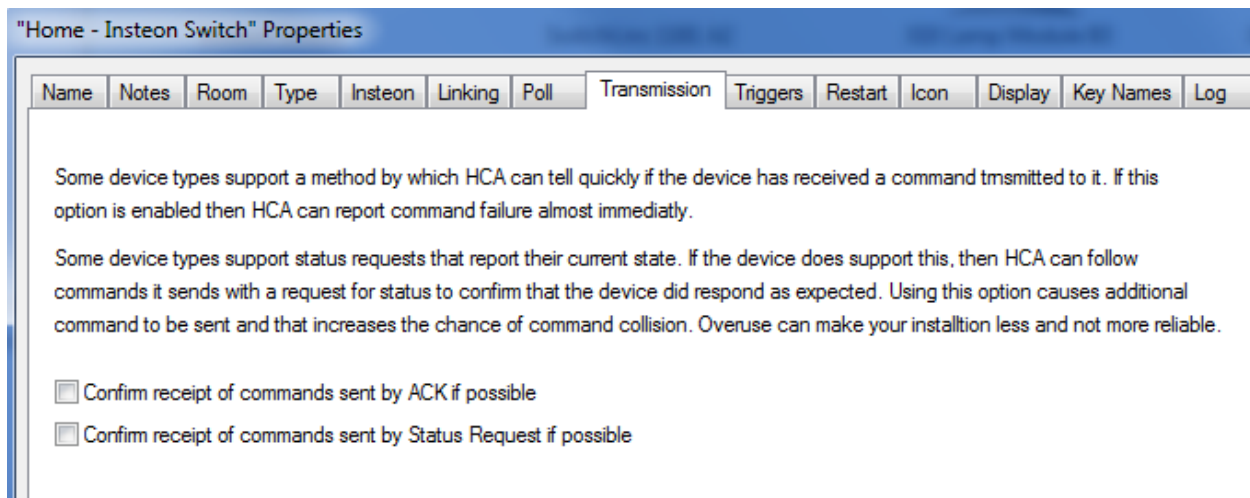
Device Control

In previous HCA versions, the Home Mode concept was introduced. This version builds upon it with a much deeper implementation of the concept than before. In previous versions there were ways that would control a device even though it shouldn't have been due to the home mode. In version 14 this has been much more fully implemented to better implement Home Modes and how they effect device control. You can think of this as a "bug fix" but things that were allowed before – incorrectly - you may find are not allowed now.

Device Control notification

In previous versions of HCA there were methods to confirm that a device received the command sent. These were protocol specific and not available consistently in all protocols.

In V14 this has now changed so that each device contains two options for confirming that a command is received by a device. These are specified on the "Transmission" tab of the device properties. (For older X10 devices the options available for them are on the "Options" tab).



The two forms of confirmation are:

- Request and look for an acknowledgement from a device. Not all protocols support this and those that do support it don't offer it in all contexts. UPB, Insteon, and Hue lights do offer this. However, if you are using UPB devices and your installation has a repeater then the ACK method is unavailable.
- After sending the device a command, send an additional command to request the status of the device to see if it is at the level the original command was trying to achieve. Not all devices have a status request feature.

Both of these confirmation methods attempt to retry the command if it can't be confirmed. If after several attempts the icon for the device will be marked in a way to show that communication with it has failed.

Note: It is important to understand that these confirmation methods can't be used in all cases. For example, if UPB device is commanded to go to 50% over 10 minutes then a status request during those 10 minutes will not show 50% but the current level of the device as it goes – slowly in this example – to 50%.

Also Note: Don't overuse this feature as it can make a system less reliable as the status request commands can collide with other commands and the result is that both commands are lost.

Here is an example. There is a device icon showing ON. You tap on it. It immediately changes it's representation to show OFF. This confirms that you did indeed tap on it. The command is sent out. If one or both command confirmation options are enabled, HCA in the background will confirm receipt of the command. If that happens then nothing changes on the display. If the command confirmation fails then the icon changes to show the error. How that is done is in the next section.

A note about scenes

A scene command – in any protocol – can by its nature update more than one device. HCA makes no attempt when activating or deactivating scenes to confirm receipt of the command by each scene members. When using the UI and right-clicking on a device, those scenes that the device responds to are listed in the popup menu. But when the command is sent, it is sent as a scene command and not a command to that single device. Again, even though you right-click on a specific device if you choose a scene name that scene is activated for all devices that participate in that scene and not just the device right-clicked upon.

Icon Marking

Icons show the state of the device in different ways depending upon the theme used. There are two basic themes: a gray-scale theme where the OFF representation of the device always shows in gray tones, and in the other theme some of the devices have both an ON and OFF representation and others with only a single representation.

Control UI

In the Control UI, and in the display shown by the Windows, Android, and iOS clients, the icon shows the state of the device and the text below the icon shows the state and possibly other information.

- Text in default color: Device is off, no additional information
- Text in yellow: Device is ON, no additional information
- Text in Green: Device is suspended by the current home mode
- Text in Red: Communication problem with the device. This is the result of the command confirmation method outlines in the previous section.

In the case of the green or red text, the state of the device can be seen in the icon itself as the gray-scale themes are always used in the clients.

Development UI

In the development UI the color of the text below the icon isn't used to show information as in the Control UI. Instead a box is drawn around the device icon. The color of the box shows the additional information.

- Yellow: The device is on. This isn't shown if the device icon has different representations for ON and OFF
- Shades of brown. The device is dim. This isn't shown if the device icon has different representations for ON and OFF
- Red: Communication problem with the device. This is the result of the command confirmation method outlines in the previous section
- Green: Device is suspended by the current home mode
- Black: The device is suspended or disabled.

Note: This is approximately the same model used in previous versions of HCA. The major difference is the use of the red text or red box. In previous versions, this was used to identify devices that were suspended. This has been repurposed to show devices with communication problems.

Implications with device control or query within a program Compute and Compute-Test elements

In the Compute and Compute-Test elements there are functions that control devices and query their status. In past versions of HCA there was little consistency on how these operations reported problems. With this version, all functions that communicate with devices have a negative return value if the operation failed and a zero or positive value returned when they did work. Here are the functions.

IsOn

IsOff

IsDim

DimPercent

If the device is queried and it doesn't respond, result = -1

On

Off

DimToPercent

If device control fails then result is -1. Note if "confirm by status" is enabled and the command is transmitted to the device successfully, then the result will be non-negative as the status request and confirm is done in the background.

SetToScene

If device control fails then result is -1.

If able to send scene command then result is a string containing device scene before it was changed

Temperature

SetPoint

Mode

IsFanOn

IsEconomy

IsAuxHeat

Humidity

HeatSetpoint

CoolSetpoint

If access to the thermostat fails, the result is -255

ChangeHeat

ChangeCool

ChangeMode

ChangeFan

If access to the thermostat fails, the result is -1. If success, value is 1. NOTE: This is a change from V13 and may require changing programs.

GetThermostat

If access to the thermostat fails, the result is -255.

SetThermostat

If access to the thermostat fails, the result is -1. If success, value is 1. NOTE: This is a change from V13 and may require changing programs.

A few words about communication problems

As said above in the section on device control notification, a device could be marked as having communication problems if:

- When sent a command the device didn't respond with an ACK – if the protocol has that.
- It supports status requests and that option is enabled in the device properties and the status reply either didn't arrive or reported an incorrect level.

In either case the device will be marked showing a communication problem. But if another command is sent and that command *is* received then the communication error marking is removed. You can still see an Alert for it – depending of course on how the alert is configured - so the info is not lost.

One more thing to be aware of with this new error marking scheme: Suppose a device shows that there was a communication problem with the device. Then if a scene command is then sent and the device does respond, since HCA doesn't confirm scene commands the device would have responded correctly by the error marking wouldn't be removed in this case. This is because HCA doesn't know with scene commands that the device did respond.

There are an upgraded iOS and Android apps to make all this happen.

Changes to Alert Manager

Note: Change to V13 designs may be required.

Several alerts have been removed and others reoriented to be more general. The alerts generated are:

alert	code	text	name
	0		
Unknown Reception	1	ID Info appropriate to the protocol	
	2		
	3		
	4		
UPB Missing Sequence	5	ID info if reception not from a device	device name
	6		
Status Poll Fail	7		device name
Confirm Receipt Fail	8		device name
Confirm Receipt All Fail	9		device name
Interface Error	10	error text + interface name	
Weather Observation Fail	11		
	12		
Power Out	13	date-time	
Power Restored	14	date-time	
Program Error	15	Error text	program name
Confirm Receipt Ack Fail	16	ID info if transmission not to a device	device name
Client Disconnect	17	disconnect/reconnect info	
No Reception Group 1	18	Last reception time	device name
No Reception Group 2	19	Last reception time	device name
No Reception Group 3	20	Last reception time	device name
No Reception Group 4	21	Last reception time	device name
User 1	22	text supplied in _AlertAdd	program name
User 2	23	text supplied in _AlertAdd	program name
User 3	24	text supplied in _AlertAdd	program name
User 4	25	text supplied in _AlertAdd	program name
Interface Disconnect	26	Interface name	
DDNS Update failed	27	DDNS System message	

Note that the “Unknown Reception” is now used for all protocols as is the “Confirm Receipt Ack Failed” alert.

If an alert is configured to start a program on an alert, the program now is passed parameters rather than using global variables for the alert info. The parameters for the program should be configured as shown below:

Parameters

Parameter count:

Parameter 1 name: <input type="text" value="Code"/>	<input type="button" value="Value"/>	Parameter 3 name: <input type="text" value="Count"/>	<input type="button" value="Value"/>
Parameter 2 name: <input type="text" value="Text"/>	<input type="button" value="Value"/>	Parameter 4 name: <input type="text" value="Name"/>	<input type="button" value="Value"/>

The code, text, and name are as defined in the above table. The “count” is the current count for that alert.

Changes that impact X10

There will be several changes in X10 support. Don’t worry, X10 is still being supported! But there are some areas that may impact X10 users. If you are using Insteon or UPB then none of this need concern you.

Note: Change to V13 designs may be required.

Group operations

Suppose in X10 you had two devices, one with address B2 and the other B6. If those two devices were in a HCA group and you controlled the group ON, previous versions of HCA sent X10 commands so that both devices would come on at the same time. This no longer happens. Now one goes on and then the other. No big deal but it is different.

Dim

Dimming with the most basic devices has always been difficult. Unlike UPB or Insteon there was no good way to dim an X10 device to a percentage. The device got controlled ON and then some number of DIM commands are transmitted to control the device to go down some number of levels to a final level approximating the desired percentage. A particular device, with a particular wattage bulb, controlled by a particular X10 interface, would get you to about the percentage wanted. The new version of HCA may not get to the exact same percentage as the current version. Close but no guarantee of the exact same.

Preset-Dim

There are some X10 devices made that used an alternative form of dimming called “Preset Dim”. Basically, this gave the device 32 levels of dim and the ability to go directly to any of those 32 levels directly. HCA tried to expose this – a bad idea - and that changed. If a device does support preset-dim then a dim operation will still use it but you will no longer be able to select the exact one of the 32 levels desired. This effects the DIM visual programmer element as in previous versions the DIM element let you specify a preset-dim level but that was removed.

Percentages and Levels

In HCA, most everywhere where you can control a device to a level other than OFF or ON accepts a percentage. For example, the DIM Visual Programmer element, the popup slider in the UI, the “hover” tip over a device that shows its percentage. Unfortunately, there remained in all versions since version 1.0 (!) some places where device “levels” were exposed. As time went on this really didn’t matter as

most modern devices all have 100 levels so a percentage and level are the same. But the holdover persisted and it caused difficulties that needed to be resolved for support of new features.

All the places in HCA where “levels” were still exposed are changing This includes these Compute expression functions:

- `_dimLevel`. Instead use `_dimPercent`
- `_dimToLevel`. Instead use `_dimToPercent`

Also effected are these HCA Object methods.

- `Device.DimToLevel`. Instead use `Device.DimToPercent`
- `Device.DimUp`. Instead use `Device.DimUpPercent`
- `Device.DimDown`. Instead use `Device.DimDownPercent`
- `Controller.DimToLevel`. Instead use `Controller.DimToPercent`
- `Controller.DimUp`. Instead use `Controller.DimUpPercent`
- `Controller.DimDown`. Instead use `Controller.DimDownPercent`

The expression functions and object methods remain but they now do the exact same thing as the substitute function/method. For devices that have 100 levels – Insteon, UPB, etc – the old functions are retained so as not to break existing designs as they already had the same effects as the older functions/methods.

Test element – Is Dim

The visual programmer Test element when testing for dim used levels and not percentages. If your design contains X10 devices and you use the Test element testing for their level, those elements must be updated to use percentages rather than levels. No attempt has been made to automatically upgrade the elements in older files.

Components Removed / transferred to Legacy

- METAR weather provider has been removed
- Clock correction – configured in HCA Options – has been removed
- Web Component moved to legacy status

HCA Cloud

Many changes in V14 for all those parts of HCA that reside in the cloud. This is a big topic and is described in these three documents:

General introduction to this topic and upgrading from V13 to V14.

www.HCATech.com/download/V14/Doc/TechNotes/TechNote_500_HCACloud.pdf

Working with Partner services

www.HCATech.com/download/V14/Doc/TechNotes/TechNote_501_CloudPartnerServices.pdf

Accessing the HCA Server from your own hardware and software:

www.HCATech.com/download/V14/Doc/TechNotes/TechNote_502_CloudDeveloperAccess.pdf

New Icons

Added to the icon themes are now icons named:

- Audio Video
- Bookshelf lights
- Camera
- Closet lights
- Relay A (use for ON = closed, OFF = open)
- Relay B (use for ON = open, OFF = closed)
- Wall sconce

In addition, the image used for the power meter tile has changed to better match the others in size.

Changes to Suspend

HCA since version 1 has a concept of a “suspend” device. The idea is that some device would be controlled by a schedule, a program, or by some reception and for some reason you wanted to stop that action for some length of time. A useful idea I think. The problem was because of subsequent changes on how HCA works – client-server, protocol bridges, etc. – the utility of having three kinds of suspend became less useful and confusing and almost never needed.

While “suspend” remains around, it is getting reduced to a single state. A device is either suspended or not. If it is suspended then any action that isn't from you explicitly controlling it from the HCA UI or a mobile app has no effect.

Suspend is now much more like “Home Mode” suspend except that a device becomes suspended by an action from the UI or a program and not by the change in home mode based upon how the device is configured to respond to that change.

Changes to Security

Another overly complex area in the current HCA that has been in existence since version 1 – also little used and no longer correctly implemented – are two passwords that could be set that would block actions from the UI in modifying the design and/or controlling devices. These are being removed. Only two remain – the remote access password and the code (password) to exit from the Control User Interface back to the Development UI.

Miscellaneous

- Added support for PulseWorx Gateway as a supported UPB interface
- The method used to start a program after a weather observation is made is changed. Previously this was done from the Weather provider dialog. Now this is done by creating a trigger on the program. Select from the “Special Trigger” types.
- New expression functions:
 - `_ClientTable ()`
This generates a HTML table that contains information on client connections. The table shows each client connection organized by client name. It provides the last connection and disconnection time for that client.
 - `_Num (<any>)`
Converts the argument to a number. Like the `_Int` function but doesn’t truncate the result to a whole number.
 - `_SetCurrentState(<object name>,<percent>)`
Sets the internal state of the named object to either ON, OFF, or DIM and if dim to the supplied percentage. A side effect of this function is to update the icon for the device to show the changed state.
 - `_SetSunriseDelta` and `_SetSunsetDelta`
These change the local sunrise and sunset minutes different from the computed values and are seen in Home Properties on the location tab. The value returned is the existing value before it is changed to the supplied value.
- Modified expression functions:
 - The expression build-in `_ON` now takes an optional second parameter which is used when controlling indicators on keypads.
- New option on Home Properties security tab to allow use of the HCA Objects even if a remote access password is set to connect without that password. Default set to “allow” scripts to run without needing to supply a remote access password.
- Bug fixes:
 - Clearing Alerts in the summary and details tabs is now much more efficient.
 - Display of references for X10 devices wasn’t always complete. Now better.
 - Redisplay of display pane and the ribbon as the alert count changes is now reduced.
 - In the inspector, some of the messages related to devices didn’t always have links to the device with the problem. Now they do.
 - The “Show Message” element when used in client-server mode would always cause connected clients to show the message even if the option to not do so was enabled.
 - A few long-standing issues with multi-select copy and delete from the design pane were resolved.

- In the “Update Tile” visual programmer element the list of tile names could show duplicates. This has been corrected.
 - The “Update Tile” visual programmer element updates all tiles with a given name. The check for the tile names was done using a case-sensitive lookup. Now the lookup for matching tile names is case-insensitive like other HCA text comparisons.
 - A fix was made to prevent errors that would happen when attempting to debug a second program while an existing debugging session is active.
- In HCA expressions, all variables have no fixed type. A variable can contain a string, a Boolean (yes/no), a date-time, or a number. When comparing two variables with different types the following procedure is now used:
- **Date-Time compared with Boolean**
The Date-Time is converted to a Boolean. If the date-time is null (has not been assigned to) it is converted to “No”, otherwise “yes”. Then the comparison compares two Booleans.
 - **Date-Time compared with Number**
Internally a date-time is a number so its type is changed to “number” but the value left alone. Then the comparison compares two numbers.
 - **Date-Time compared with String**
If the string contains something that can be converted to a date-time, then the string is so converted and the comparison compares the two date-times.

If the string doesn’t contain something that can be converted to a date-time, then the image of the date-time is made and the comparison compares the two strings.
 - **Boolean compared with Number**
The Boolean is converted to a number: “yes” = 1, “No” = 0. Then the comparison compares the two numbers.
 - **Boolean compared with String**
If the string can be converted to a Boolean – contains “yes”, “true”, “1” makes a “yes” - “no”, “false”, “0” makes a “No” Boolean. Then the comparison compares two Booleans.

If the string can’t be converted to a Boolean, then the Boolean is converted to a string as “No” or “Yes” and the comparison compares two strings.
 - **Number compared with String**
If the string contains something that can be converted to a number, then the string is so converted and the comparison compares the two numbers.

If the string doesn’t contain something that can be converted to a number, then the image of the number is made and the comparison compares the two strings.

##end##