



# Home Control Assistant Version 13 Appendixes

[WWW.HCATech.com](http://WWW.HCATech.com)

The information contained in this document is subject to change without notice. Advanced Quonset Technology, Inc. provides this information “as is” without warranty of any kind, either expressed or implied, but not limited to the implied warranty of merchantability and fitness for a particular purpose. Advanced Quonset Technology, Inc. may improve or change the product at any time without further notice; this document does not represent a commitment on the part of Advanced Quonset Technology, Inc. The software described in this document is furnished under a license agreement or nondisclosure agreement. The software may be used or copied only in accordance with the terms of the licensing agreement.

Windows is a registered trademark, and Windows NT is a trademark of Microsoft Corporation.

All other product names and services identified in this document are trademarks or registered trademarks of their respective companies and are used throughout this document in editorial fashion only and for the benefit of such companies. No such uses, or the use of any trade name, is intended to convey an endorsement or other affiliation with Advanced Quonset Technology, Inc.

© 2001-2016 Advanced Quonset Technology, Inc.

All rights reserved. Printed in the U.S.A.

March 1, 2016

## Contents

Appendix 1 HCA versions and Interface Support.....	1
Appendix 2 CM15 / CM11 .....	3
CM11 features .....	3
CM11 menu selections.....	3
CM15 features .....	4
Appendix 3 Thermostats.....	5
Supported Thermostats .....	5
Creating a thermostat device.....	5
NEST Thermostat.....	6
Thermostat Properties.....	7
Working with thermostats.....	8
Compute and ComputeTest element support.....	10
Additional information for X10 RCS thermostats .....	11
Practical Design Group THUM .....	12
Appendix 4 Weather .....	13
Weather Providers .....	13
Weather Provider Setup.....	14
Weather Underground .....	14
Reading data from a file.....	17
WeatherLink Program.....	18
Weather Display Program.....	20
Virtual Weather Station Program.....	21
METAR weather data .....	22
Weather database.....	23
Visual Programmer .....	24
Weather Triggers .....	25
Weather Data and the Compute and Compute Test elements .....	25
Weather Underground .....	26
Appendix 5 IR Interfaces.....	29
GC100- Setup .....	30
IR Keypad builder.....	31
Creating a HCA device for an IR Output Port .....	34
IR sequences in programs .....	36
IR sequences in schedules.....	37
GC-100 Digital sensor ports .....	38
Creating a HCA device for a sensor port .....	38
Using a G-100 sensor port for program triggers .....	39
GC-100 Relays.....	40
Creating a HCA device for a relay.....	40
Using a GC-100 Relay.....	41
Design Inspector .....	41
BC4 Setup.....	42
Bitwise BC4 device type selection.....	44
IR Keypad builder.....	45
Creating a HCA device for an IR Output Port .....	48
IR sequences in programs .....	49
IR sequences in schedules.....	51
Appendix 6 Wireless Interfaces.....	53
What are the W800RF32 and the MR26?.....	53

Use and Configuration .....	54
MR26 .....	55
W800RF32 .....	55
Wireless Devices .....	56
Using Wireless Triggers .....	57
Appendix 7 HCA Objects .....	59
Introduction .....	59
Example .....	59
HCA Object Model .....	60
Error Handling .....	60
Controller and Device Objects .....	61
Obsolete Methods .....	61
HCA Object .....	62
Device, Group, Controller common methods .....	67
HCA.Device Object .....	75
HCA.Program Object .....	78
HCA.Group Object .....	80
HCA.Controller Object .....	80
HCA.Schedule Object .....	80
HCA.Flag Object .....	81
HCA.Log Object .....	85
HCA.Display Object .....	87
Passwords .....	91
Appendix 8 Universal Powerline Bus (UPB) .....	93
What is UPB? .....	93
UPB device setup and configuration .....	93
Powerline Interface Module (PIM) .....	94
Network Import .....	94
Getting ready for export in UPStart .....	95
Importing a UPB network .....	96
Generic UPB Devices .....	97
Device Properties .....	98
UPB Commands .....	99
Direct commands .....	100
Link commands .....	100
HCA support for scenes and device command features .....	101
Direct commands .....	101
Scene or Link commands .....	102
Program triggers for UPB events .....	105
UPB Action Triggers .....	106
UPB Powerline Message .....	107
Hints and Tips .....	108
Appendix 9 Insteon .....	111
What is Insteon? .....	111
Insteon devices .....	112
Adding an Insteon Device .....	112
Modifying an Insteon Device .....	114
Insteon PowerLinc Interface .....	115
PowerLinc Address Database .....	116
PowerLinc Interface models and Insteon firmware .....	117
Insteon Tools .....	118
Multi-Add .....	118
The Insteon linking model .....	119
PowerLinc Swap .....	120
Network Capture .....	122

Network Map .....	124
Network Clean .....	124
Device Replace .....	126
Changing Local ramp rate and Local Level .....	127
Device and Keypad Linking .....	128
Device Linking Tabs .....	128
Multi-Way Wizard .....	130
Scene Control without scenes .....	132
Program triggers for Insteon messages .....	134
What HCA knows of the Insteon network .....	135
Hints and Tips .....	135
Appendix 10 Protocol Bridges .....	137
What is a Protocol Bridge? .....	137
Configuring a bridge .....	138
X10 Bridge .....	139
UPB Bridge .....	140
Insteon Bridge .....	141
Limitation of Bridges .....	141
Appendix 11 Z-Wave .....	143
What is Z-Wave? .....	143
Building a Z-Wave network .....	143
Configuring the Leviton Z-Wave interface .....	144
Z-Wave devices .....	145
Z-Wave Visual Program Element .....	147
Interface Viewer .....	147
Triggers .....	149
Pattern .....	149
Flag assignment .....	150
Trigger of last resort .....	151
Appendix 12 Generic Serial, Generic IP, and Generic Server .....	153
Configuring a serial interface .....	154
Configuring an IP interface .....	157
Client or Server .....	158
Configuring an IP Interface .....	159
Configuring a Generic Server .....	160
Send and Receive .....	161
Triggers .....	161
Pattern .....	162
Flag assignment .....	163
Trigger of last resort .....	164
Working with Serial and IP Interfaces .....	164



## Appendix 1

### HCA versions and Interface Support

The Home Control Assistant is available in three versions. Each version has many similarities but a different level of support for various automation hardware.

Files written by one version can be read by any other version.

The three versions are:

- HCA Limited
- HCA Standard
- HCA Plus

Which version should you be using? It depends upon what hardware you currently have, or expect to have and what features wanted.

Interface	Limited	Standard	Plus
CM11/CM15	√	√	√
Generic Serial Interfaces			√
Generic Network Interfaces			√
Insteon		√	√
IR Interfaces			√
UPB			√
Wireless Interfaces		√	√
X10 PowerLinc family	√	√	√
Z-Wave			√
Lightolier products			√
Magic Module			√
Marrick Interfaces	√	√	√

Note: The interfaces in **red** have been designated as *Legacy Support* only and must be enabled before they can be selected in the interface configuration. Legacy support is enabled on the *HCA Options* dialog *Legacy* tab.

In addition to support for interfaces and protocols there are a number of feature differences between the three versions.

<b>Feature</b>	<b>Limited</b>	<b>Standard</b>	<b>Plus</b>
Client-Server			√
Dynamic DNS			√
Alexa Support			√
Control User Interface			√
Object Interface			√
Thermostats			√
Tiled Displays			√
Visual Program Script element			√
Visual Program Debugger			√
Web Component			√
Weather			√

Finally, in addition to the above limitations, *HCA Limited* is an entry level product designed to introduce you to HCA. It is limited in the number of programs and schedules that can be created.

## Appendix 2

### CM15 / CM11

This appendix describes the features of the X10 CM11 and CM15 interfaces and covers these topics:

- CM11 and CM15 features and capabilities
- CM11 and CM15 menu selections
- CM15 Wireless setup

**Note:** A special device driver is needed for the CM15 to work with HCA. There are instructions for installing this device driver for Windows 7 and 8 and a separate note for Windows 10. Please review it before starting with the CM15.

---

#### CM11 features

The CM11 is an X10 interface made by the X10 corporation. It is sometime seen with a model number of CM11A, HD11A, and CK11A. For the purposes of HCA, they are all the same.

The CM11 has a place for a battery. This battery is used to keep its real time clock correct in the event of power failures. If you are not using the CM11 for schedule execution, that is, you have not and do not intend to download a schedule into to it; we recommend you do not install the battery.

This interface is connected to a serial port on your computer and plugged into the powerline. Because it connects directly to the powerline it is capable of sending X10 extended codes. These codes are used by the LM14 style modules and some Leviton switches.

This interface does not report X10 collisions or reception problems so the Log Setup features for reporting these conditions have no effect.

There is HCA support for downloading schedules into the CM11. While the CM11 does have the capability to execute programs in response to X10 events, HCA doesn't support this due to the extremely limited memory in the CM11 for this.

More information on downloading schedules to the CM11 is contained in the Appendix on Downloading.

---

#### CM11 menu selections

There are menu selections associated with the CM11/CM15 and they are reached by selecting the menu below the CM11/CM15 icons in the ribbon *Interfaces* category. These menu selections are:

- CM11 Download
- CM11 Clear memory
- CM11 set clock

The Clear Memory selection is used to remove any downloaded programs and / or schedule from the memory contained in the CM11.

The Download Schedule selection is used to select, translate, and download one of the schedules in your design to the CM11 memory. It is described in the appendix on downloading.

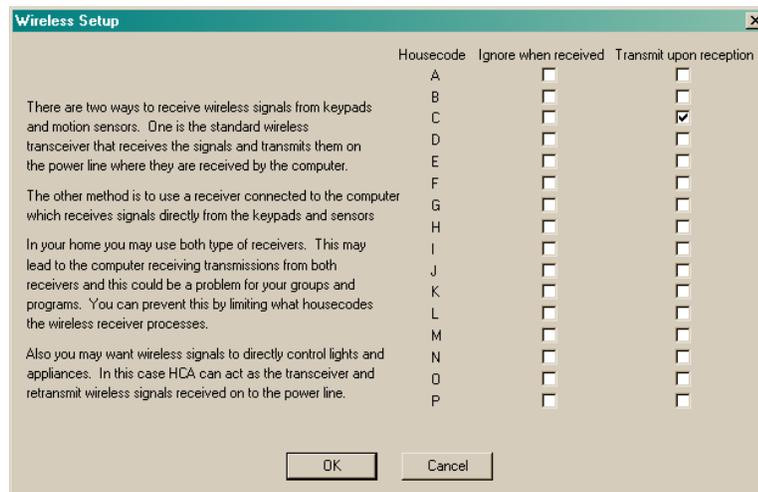
The Set date and item selection sets the CM11 clock to the same date and time as your computer. You need only do this if you are using downloaded schedules.

---

## CM15 features

The CM15 is a more modern version of the CM11 that also includes a wireless receiver. An inconvenience of this interface is it requires a special device driver to use it with HCA. Installation of this driver is described in a technical note on the HCA web site.

Once installed it supports the same features as the CM11 except that download is not supported.



As this text in this dialog explains, you can have HCA both ignore transmissions from some house codes and also act as a transceiver for others.

While you may think that the CM15 Wireless Interface can replace all the X10 transceivers in your home – and it can – there may be a few problems yet to solve.

What problems? For example you have a wireless remote for housecode B and a transceiver set for housecode B and a lamp on B3. You send a B3 and the lamp comes on. This happens because the transceiver picks up the wireless broadcast and sends the B3 out on the power line. No computer involvement.

Now without the transceiver and using a wireless interface the B3 goes right into the computer bypassing the powerline so the light doesn't come on. To recover the function you want you could write a program that is triggered on the wireless reception, but there is a better way. Another option in this Setup dialog is to broadcast onto the powerline signals it receives, thus replacing the transceiver function. You can make this happen for selected housecodes. If you checked this option for housecode B, what would happen is this:

Wireless keypad → Wireless Interface → HCA → X10 interface → lamp

Thus HCA becomes the transceiver.

## Appendix 3

### Thermostats

This appendix describes how to work with thermostats in HCA and covers these topics:

- Supported thermostats
- Creating a thermostat device
- Thermostat properties
- Working with thermostats
- Additional information for RCS thermostats
- Practical Design Group THUM device

---

#### Supported Thermostats

HCA fully supports these thermostats:

- NEST
- SmartHome Insteon Thermostat adapter
- SmartHome 2441TH and 2441ZTH Insteon thermostats
- Residential Control Systems (RCS) TX15 Thermostat
- Residential Control Systems (RCS) TX15B Bi-Directional Thermostat
- Residential Control Systems (RCS) TXB16 Bi-Directional Thermostat
- Residential Control Systems (RCS) TU16 UPB Thermostat

In addition to these thermostats, some other devices, while not really thermostats, are supported in similar ways in HCA. These are:

- SmartHome X10 TempLine
- Practical Design Group THUM

---

#### Creating a thermostat device

Adding a thermostat to your home design is like adding any device, start the New Device Wizard and at step three when the device type is selected, select one of the above listed types.

The UPB thermostat is imported into your design in the same way as other UPB devices.

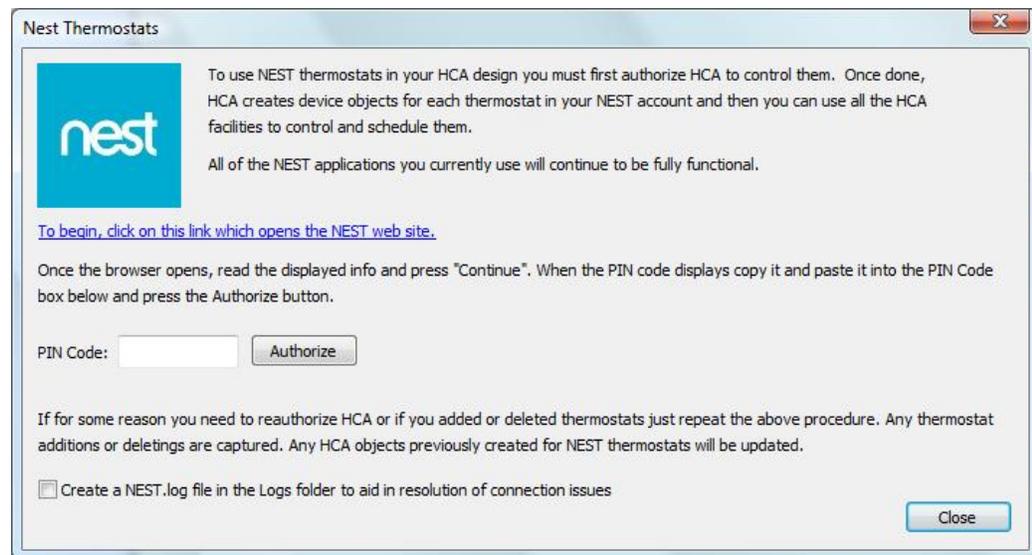
Insteon thermostats, like all Insteon devices, are assigned an address at the factory and it is discovered during add.

The NEST thermostat is handled much directly and the next section explains the add process.

## NEST Thermostat

The NEST thermostat has a much more complex protocol for communication than any of the other supported thermostat types. Communication is not made with the thermostat directly but with the NEST servers which then control the thermostat. Before HCA can work with it you must first authorize HCA to access all the thermostats in your NEST account.

Select the “Nest” button in the Protocols category.



As the dialog says, you first click on the link to login to your nest account and then to authorize HCA. The web site displays a PIN Code that is entered in HCA. Once completed then HCA creates devices for each thermostat in your account. This need only be done once. If you add or remove thermostats just repeat this process and new thermostats are added and removed ones deleted.

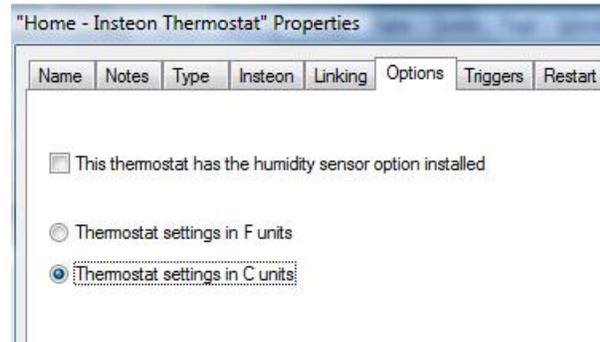
**CAUTION:** When you click on the link your default browser is opened. Some users have reported problems with Internet Explorer in that the page comes up with a big gray rectangle and no controls or other display. Chrome seems to work fine.

**NOTE:** Providing HCA access to the thermostat in no way limits you use of all the other options for controlling your NEST thermostats – mobile clients, browser support, etc. In fact, you should view the HCA support of the NEST thermostats as an adjunct to all the facilities that NEST provides in their mobile applications.

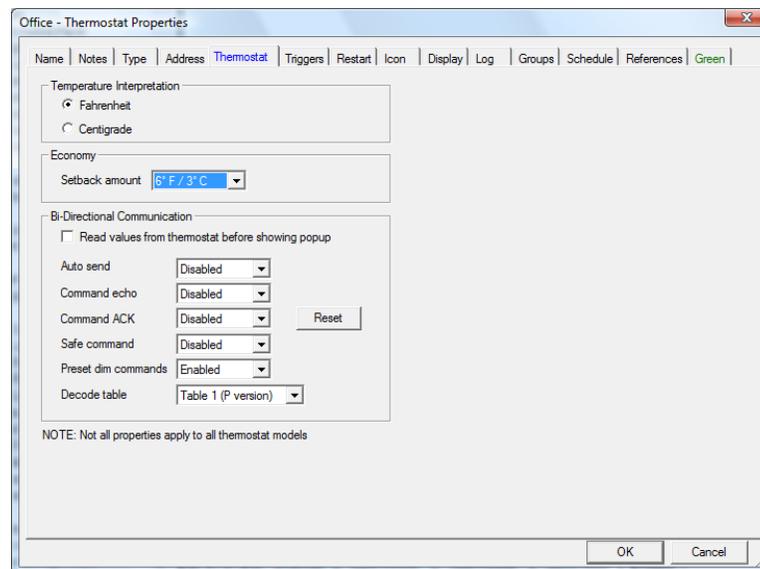
## Thermostat Properties

When the properties dialog of a thermostat device is opened, there is an additional tab labeled *Options*. What appears on this tab depends up[on the kind of thermostat in use.

For the NEST and Insteon thermostats the tab shows only a few options:



For the older RCS X10 thermostats the tab appears as:



Most of these settings should be obvious or are described in the RCS documentation and should only be changed after checking that documentation. A few notes may be helpful:

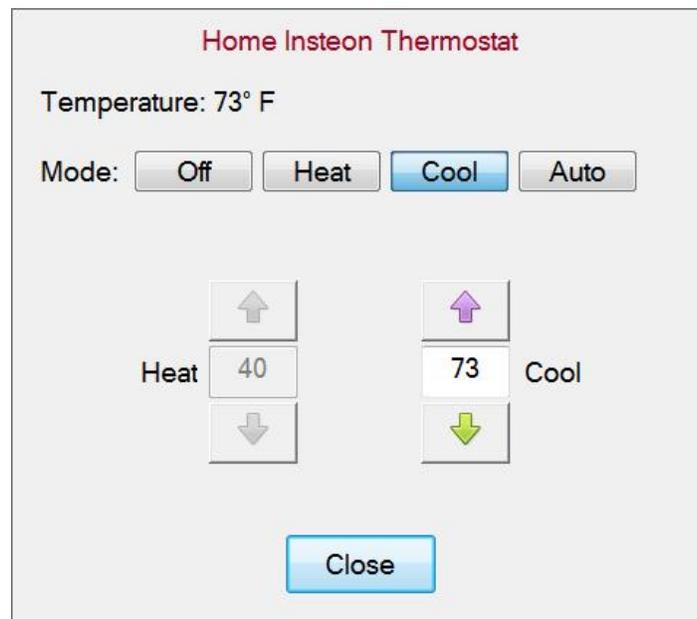
- The Command echo, Command ACK, and Safe Command options are communication modes of the thermostat that can increase the transmission reliability of commands sent to and received from the thermostat.
- The Preset dim commands, and decode table options are for the method of sending commands to the thermostats.
- The Auto Send option enabled the thermostat to send information out when the current temperature changes or changes to the setpoint, mode, etc, are made at the thermostat.
- The *Read values* checkbox tells HCA to poll the thermostat each time the thermostat popup is shown with the current temperature and setpoint. Since communication is by X10 powerline commands this will introduce a small time lag in opening the dialog.

- The Reset button does not reset the thermostat but rather resets HCA's communication with the thermostat. If any of the special communication modes are in use, and somehow the communication mode is different for HCA and the thermostat, communication will not be possible. To get HCA and the thermostat back in sync, press the Reset button and follow the directions given.
- Not all of these options apply to all thermostats. Thermostats that are not bi-directional don't use many of these options.

---

## Working with thermostats

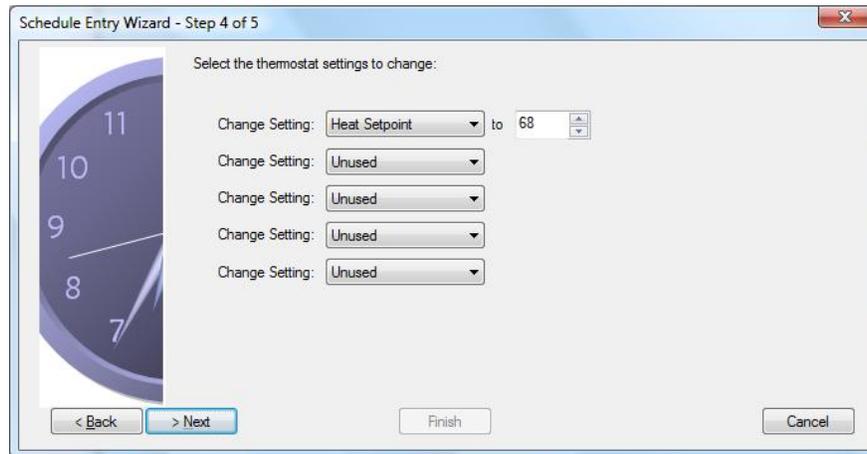
Once a thermostat is added to your home you can do many things with it. Right click on the icon and the popup menu contains a thermostat selection. Selecting it displays this control panel for thermostat.



The up and down buttons move the set points by one degree. After a 1 second pause the set point change is sent to the thermostat. This lets you change the setpoint with multiple button presses and then when the desired setpoint is entered the command is sent to the thermostat.

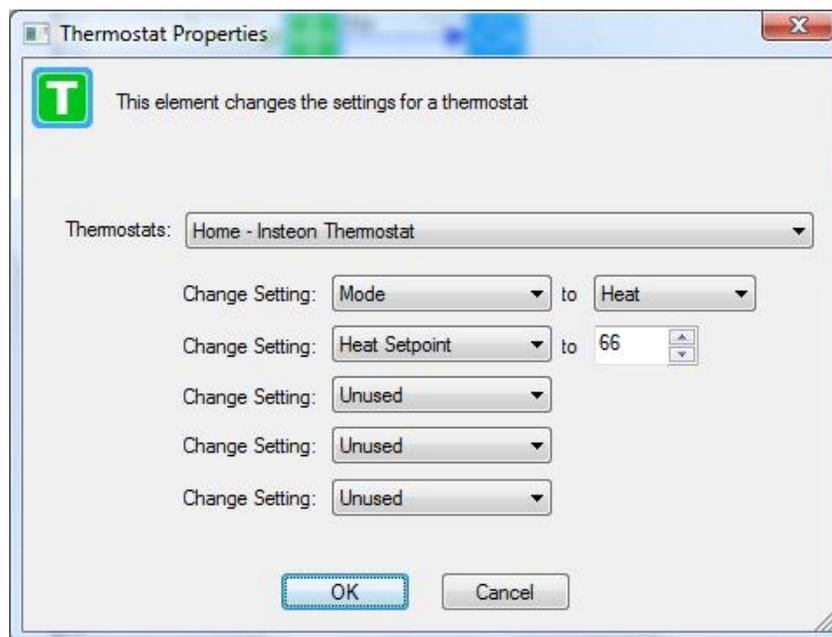
**NOTE:** Each supported thermostat has different remote access capabilities and restrictions. Some types make the heat and cool setpoints available for modification at any time. Other thermostat types only allow for remotely changing the setpoints based upon the mode. For example, the Insteon thermostat only let the heat setpoint to be set when in heat mode, the cool setpoint only when in cool mode, and only the heat setpoint when in Auto mode. The UI implements this restriction by enabling and disabling the setpoint controls as needed.

You can also schedule and program thermostats. When scheduling a thermostat rather than specifying on and off conditions, the setpoint can be changed:

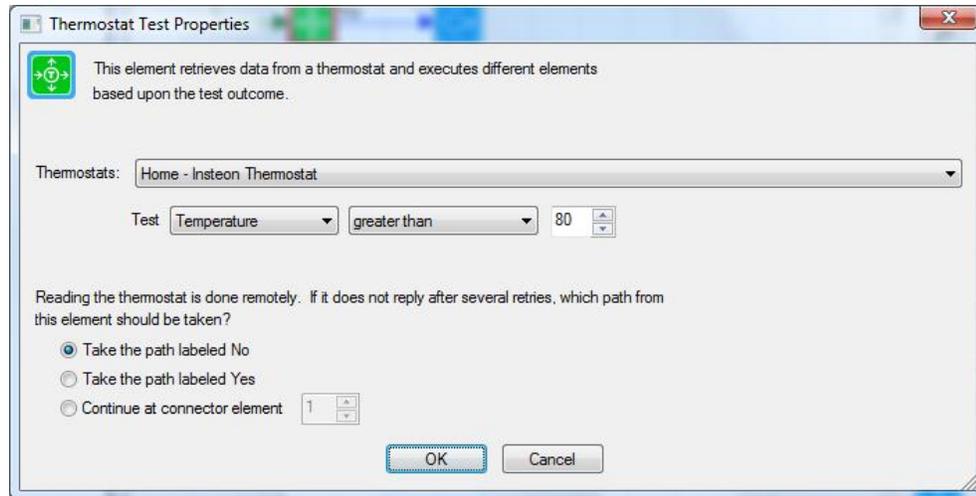


When using the Visual Scheduler to create schedule entries for a thermostat drag and drop the time markers with the T. A popup dialog allows you to select a setpoint and other options.

In programs the Thermostat and Thermostat Test elements are used to work with thermostats. The Thermostat element allows selection of the setpoint and other options. Its properties are:



The Thermostat Test element allows for reading data from a thermostat and executing different elements based upon a test on the data retrieved. As such it is only useful with bi-directional thermostats. Its properties are:



Specify the data to retrieve and then the test to be performed on the data. Like all testing elements that operate upon data retrieved from some device, if for whatever reason, the data does not come back from the request, you can specify which branch to take from the testing element.

Finally, you can start programs when various conditions reported by the thermostat happen. In the X10 RCS case this can be when the setpoint, current temperature, mode, etc is changed. If the Auto Send option is enabled (see the thermostat properties dialog above) these conditions send preset dim commands. Preset dim commands can be used as alternative triggers for HCA programs. Exactly what preset dim commands are sent for each condition is described in the RCS documentation.

---

### Compute and ComputeTest element support.

Two functions are available for thermostats: `_GetThermostat` and `_SetThermostat`

#### **Value = `_GetThermostat` (“thermostat name”, code)**

The “Thermostat name” is the two part name for the thermostat.

The code is the setting to be retrieved. These are:

Code	Setting	Returned value
0	Temperature	Integer value
1	Heat Setpoint	Integer value
2	Mode	Off = 0, Heat = 1, Cool = 2, Auto = 3
3	Fan	0 = On, 1 = Off
4	Economy	0 = On, 1 = Off
5	Aux Heat	0 = On, 1 = Off
6	Humidity	Integer value
7	Cool Setpoint	Integer value
8	Has Leaf (NEST only)	0 = On, 1 = Off

13	Nest Mode (NEST only)	0 = Away, 1 = Home
----	-----------------------	--------------------

It is up to program that use this function to request only settings supported by the thermostat and for the setpoints only when in the correct mode.

The return value is the setting retrieved or an error. You should use the `_IsBool` on the result to determine if you have received the requested data or an error.

**Bool = `_SetThermostat` (“thermostat name”, code, value, code , value, ...)**

You can supply up to 11 arguments. The 1<sup>st</sup> is the two part name for the thermostat device. The next two arguments are the code and value of the setting to change. The next arguments are up to 4 other code-value pairs.

The valid codes are:

Code	Setting	Data
0	Temperature	Integer value
1	Heat Setpoint	Integer value
2	Mode	Off = 0, Heat = 1, Cool = 2, Auto = 3
3	Fan	0 = On, 1 = Off
4	Economy	0 = On, 1 = Off
7	Cool Setpoint	Integer value
13	NEST mode (NEST only)	0 = Away, 1 = Home

Note: When changing the NEST mode it changes all thermostats in the structure associated with the thermostat being controlled

---

### Additional information for X10 RCS thermostats

The RCS thermostat does **not** store the communication protocol settings for Command Echo, Command ACK, and Safe command in a memory chip that retains its state when power is removed from the control unit.

If HCA and the thermostat are using one of the special communication modes and power is lost, when power is restored, HCA and the thermostat will be out of sync.

You may have the thermostat control unit on a circuit that doesn't lose power, or you may not. So HCA has a hard time knowing what to do upon power restore.

Because of this, power failure recovery is left up to the user in this case. If you use one of these special communication protocols, the suggestion is that you create a program that is run during power failure restoration.

This program would transmit to the thermostat to reestablish the communication mode. Specifying that a program be run at power failure restoration is done by creating a trigger for this condition.

In this program you can use the "X10 Send" visual programmer element to send out the appropriate command sequence to set the thermostat control unit into the communication mode you want.

For example, if your thermostat is on house code J and you want to enable ACK mode, create a program, mark it to run during power failure restoration (see the Advanced tab), then open the visual programmer tab. Add a "Send X10" element and set its properties as appropriate.

Again, see the RCS documentation for all the appropriate command sequences.

In this way you can tailor power failure recovery for RCS thermostats to your specific needs.

---

## Practical Design Group THUM

The THUM is a USB connected temperature and humidity monitor. While not a thermostat in the true sense of the word it does share many common properties with how HCA implements them.

To add a THUM device to your computer connect it via a USB cable then add it to your design using the New Device Wizard. Once you do that then you can:

1. Right click on the icon and a popup appears showing the current temperature and humidity.
2. Use the THUM in the Thermostat Test visual programmer element to test for the current temperature and humidity levels.

The only property you can set with the THUM is the reporting units – Celsius or Fahrenheit. To do this select the device and open it's properties dialog and choose the Units tab.

## Appendix 4

### Weather

Using weather data in your automation solution can have many benefits. Without weather data, your home automation happens regardless of environmental conditions. Some things you can use weather data for are:

- Don't run sprinkler systems when too hot or too windy
- Retract awnings when too windy
- When very hot or very cold adjust the thermostat setpoint to be a bit higher or a bit lower. In doing this you trade off a bit of comfort for what can be substantial energy savings
- Close remote control skylights when it's raining
- Adjust indoor lighting to make the house appear warmer when it's cold and cooler when it's hot. Your perception of the temperature varies with many environmental factors

This appendix describes how to work with weather data and covers these topics:

- What weather providers HCA supports
- Configuring HCA to use Weather Underground as a weather provider
- Configuring HCA to use the output of another program for current weather data.\
- Configuring HCA to use METAR observations
- Configuring HCA to add weather observation data to a database file. This database is used for tests of historical weather data – like the max temperature in the last 4 hours
- The Visual Programmer weather test element
- Weather program triggers
- Additional support for Weather Underground

---

### Weather Providers

HCA accepts weather data from three main types of sources. These are:

- HCA can retrieve weather data from the *Weather Underground* web site. You must first establish an account with them but once created, then you have access to not only current weather data – temperature, humidity - but also to forecasts and astronomical data.
- HCA reads a file that contains the weather data. These files are typically produced by another program. HCA can read these files in several formats. These are the formats produced by three programs that work with weather data: Virtual Weather Station, Weather Display, and the Davis WeatherLink program.
- HCA can retrieve weather data from the Internet using what are known as METAR reports. These are weather observations taken at airports across the country.

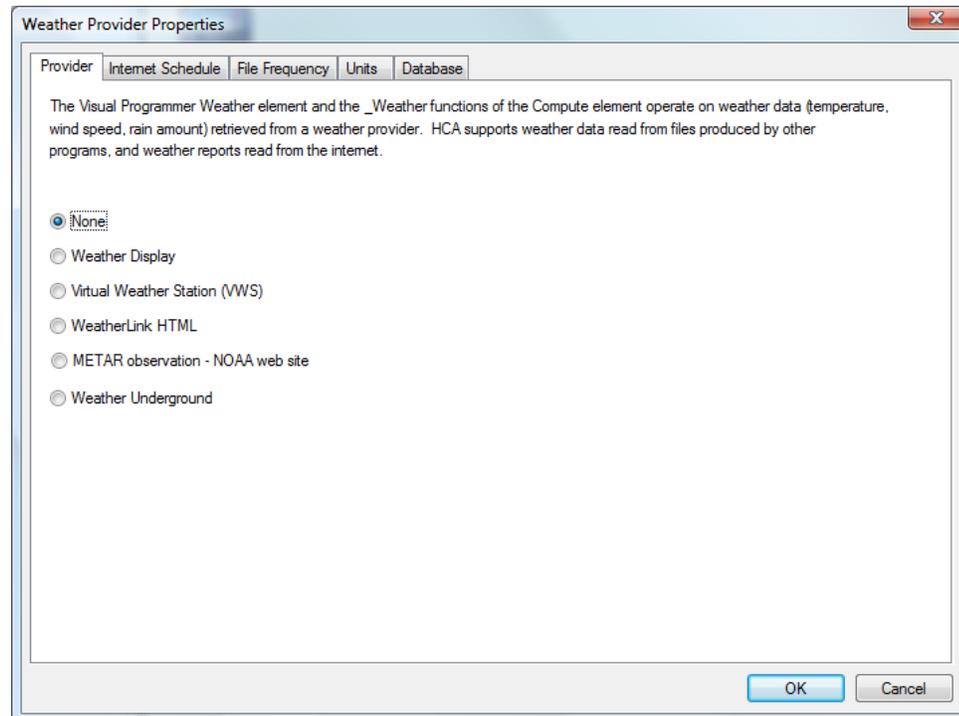
Once you have configured a weather provider, at periodic intervals weather data is retrieved. Each time this happens it is called an *Observation*. Contained in a weather observation are all the sensor data - temperature, barometer, wind speed, etc.

If you are new to working with weather data we recommend that you use the *Weather Underground* provider even if you own your own weather station. Your weather station can publish its data to Weather Underground – a feature of many weather programs - and then you can access that data from them. Working this way you also gain access to all that Weather Underground offers in addition to your own station data.

---

## Weather Provider Setup

To setup a weather provider, press the *Weather Provider Setup* button in the ribbon *Interfaces* category. This weather provider dialog opens.



The setup options vary with the chosen weather provider. Refer to the next sections for setup of each weather provider.

In addition to choosing the weather provider and its setup options, there are three other items to configure:

- The units to use for the sensor data – F or C, inches or mm, etc.
- How often a weather observation is taken
- The location of the database files that contain the historical weather data

These are covered in sections below.

---

## Weather Underground

The Weather Underground web site (<http://www.wunderground.com/>) provides weather data to applications and that data can be accessed by HCA for your use.

Before you can use Weather Underground as a weather provider you need to sign up as a developer. What Weather Underground is doing is making weather data available to people creating applications, but since each of you are a “developer” of your own home design you should have your own key.

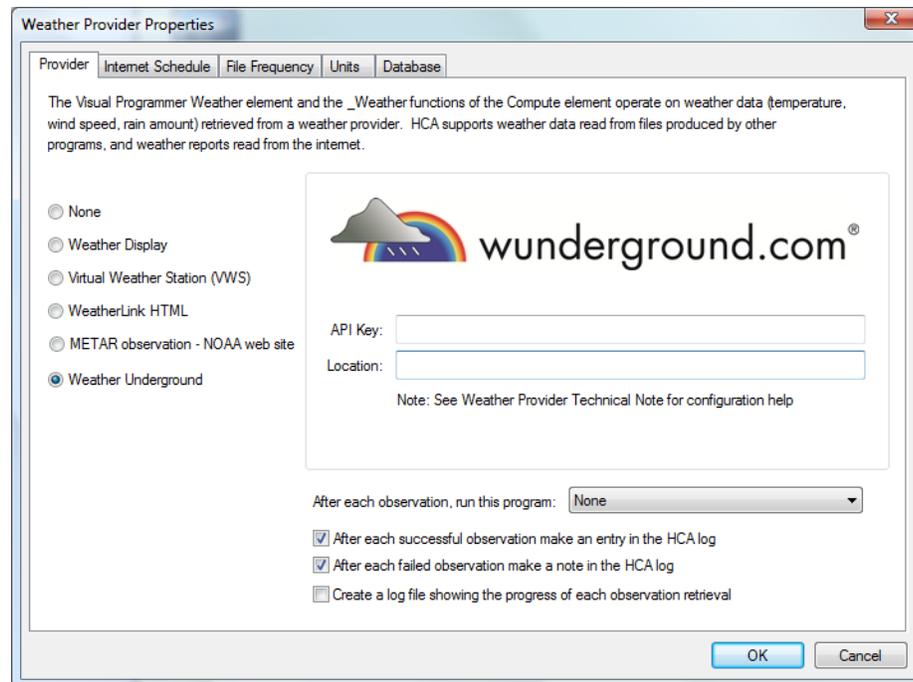
Signup for the key is here: <http://www.wunderground.com/weather/api/>

Complete the signup for the API Key and save the API Key they send in a safe place. During the signup process you may also want to do a quick review of the documentation that Weather Underground provides. This is important if you want to use Weather Underground beyond the standard weather data retrieval that HCA automatically performs.

Once you have your API Key, the next thing to do is to determine a location. Go here:

<http://www.wunderground.com/weather/api/d/docs?d=data/index>

Look at the section on “Query”. There are lots of different ways to specify your location. The easiest is to use your zip-code. Once you have a key and have determined a location you are ready to configure HCA.



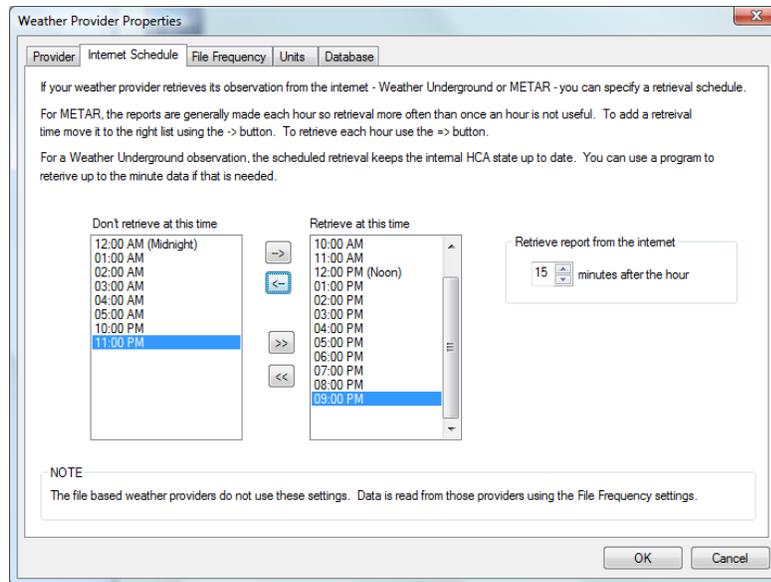
Enter your API key and location. No verification is done on either so enter them carefully.

Also in this tab are additional options. After each weather observation is taken a program can be started. You can also make entries in the HCA log (always the first of the three possible logs) after each observation.

If you choose to create a log file – last checkbox on this tab – it is located in the *Logs* folder and is called “WeatherLog.txt”

The *Internet Schedule* tab sets the schedule to be used to retrieve observations from Weather Underground. The *File Frequency* tab is used for file based providers and has no effect on Weather Underground.

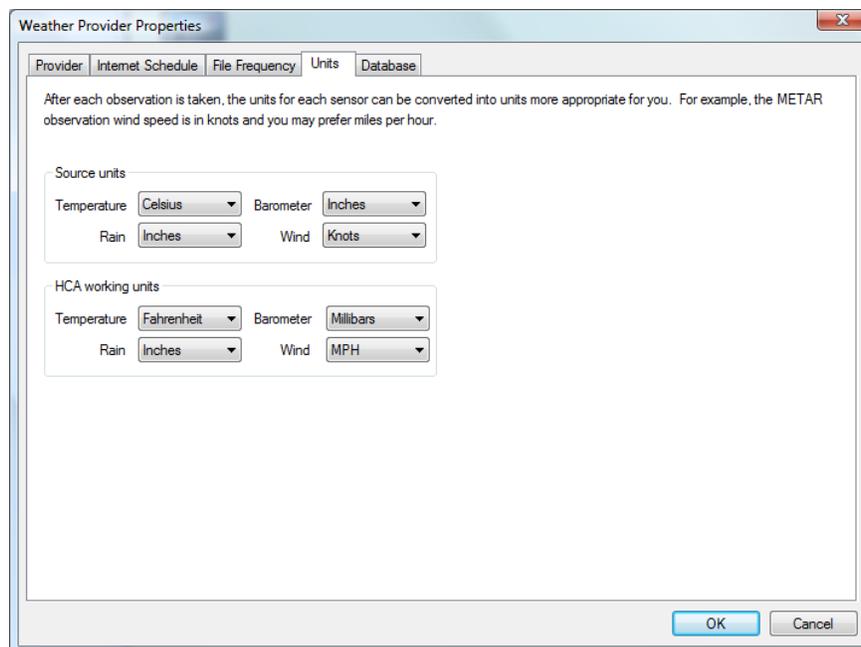
Using the schedule you can set the times during the day that weather observations are taken.



As the text in this tab explains, HCA goes out to the internet at most once an hour to retrieve the current weather observation. There are items to configure in this dialog:

- When during the hour to read the data. This is given as the number of minutes past the hour. In this example, the weather data is retrieved at, for example, 9:15 or 11:15, etc.
- What hours of the day to retrieve the data. In the above example, the data is retrieved every hour except between 10pm and 6am.

The Units tab is where you specify the units of measurement the data from the provider are in, as well as the units that you want to work in your programs. For example, while the provider may report temperatures in Fahrenheit you may want to work in Centigrade. HCA translates the source units to the working units as needed.



In the dialog above, the weather provider supplies temperature in degrees Fahrenheit, rain in inches, barometric pressure in inches and wind speed in knots. In this example, when working with weather data in the *Weather* element and weather triggers, temperature values are in degrees Fahrenheit, rain in inches, barometric pressure in millibars and wind speed in miles per hour.

When using Weather Underground, on the units tab make sure that the “source” units are configured as:

- Temperature: F
- Rain: Inches
- Barometer: Inches
- Wind: MPH

For other providers, you will have to determine the source unit choices based upon documentation or on examination of the data..

When the schedule tells HCA to take an observation – in this case retrieval of data from Weather Underground - an XML file containing the data is saved in the *Temp* folder in the HCA documents area.

After each observation is taken HCA extracts the data for these variables:

- Outside temperature
- Relative humidity
- Barometer (inches)
- Barometer Trend
- Wind speed (mph)
- Wind Dir (degrees)
- Wind Direction (string)
- Wind Chill
- Daily rain (precipitation today inches)
- Rain rate (1 hour Precipitation)
- Dewpoint
- Wind Gust (mph)
- WeatherCondition (string)

All these various values are extracted and are available to the *Weather* program element – described later in this appendix..You can also use the `_Weather` function –described in the *Expressions* chapter - to operate on these values.

---

## Reading data from a file

HCA can read data from three different file formats produced by three different programs that run on the same computer as HCA or on a different computer but create the file in a network location that HCA can access. You must configure these programs to produce the file that is read by HCA.

## WeatherLink Program

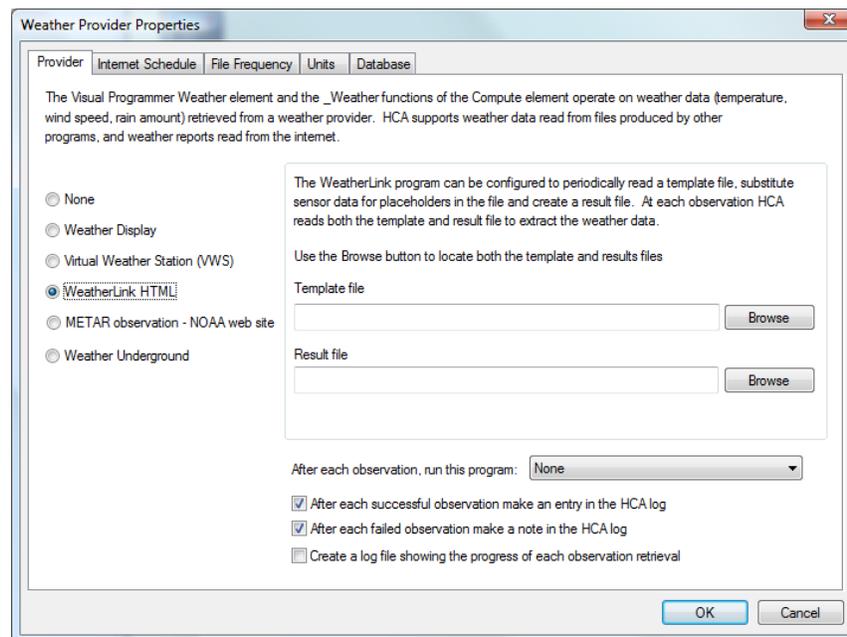
The WeatherLink program from Davis Instruments can be configured to produce a file on a schedule you set. That file is read by HCA and the weather data values extracted.

In a very similar manner as the HCA *Status Export* feature, WeatherLink reads a template file and produces a result file while replacing placeholders with data read from the weather station data.

The list of placeholders available in WeatherLink template files is very large. There are over 250 possible placeholders.

HCA uses both the template and results file produced to extract the weather data.

To setup the WeatherLink HTML as the weather data provider, select WeatherLink HTML and the setup dialog shows those settings:



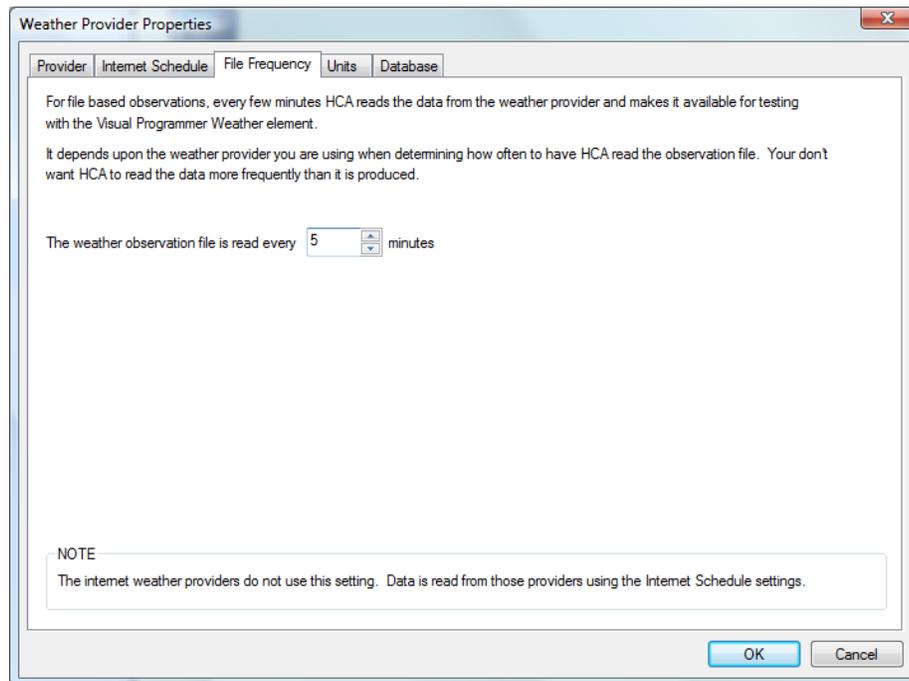
The two setup parameters are the paths to the template file and the result files.

To configure WeatherLink to produce the HTML files and what the placeholder keys are, see the WeatherLink documentation. Look in the *Internet Settings* sections.

Optionally, after the file is read a program can be started. You can also make entries in the HCA log (always the first of the three possible logs) after each observation.

**Tip:** Extracting the data from the file produced by WeatherLink can be difficult. In the HCA installation is a Windows program called *WTest* that lets you enter the path to the two files and then performs and displays the decoded information. This can be a good way to check the extraction of the data.

On the *File Schedule* tab is specified how often the data file is read.

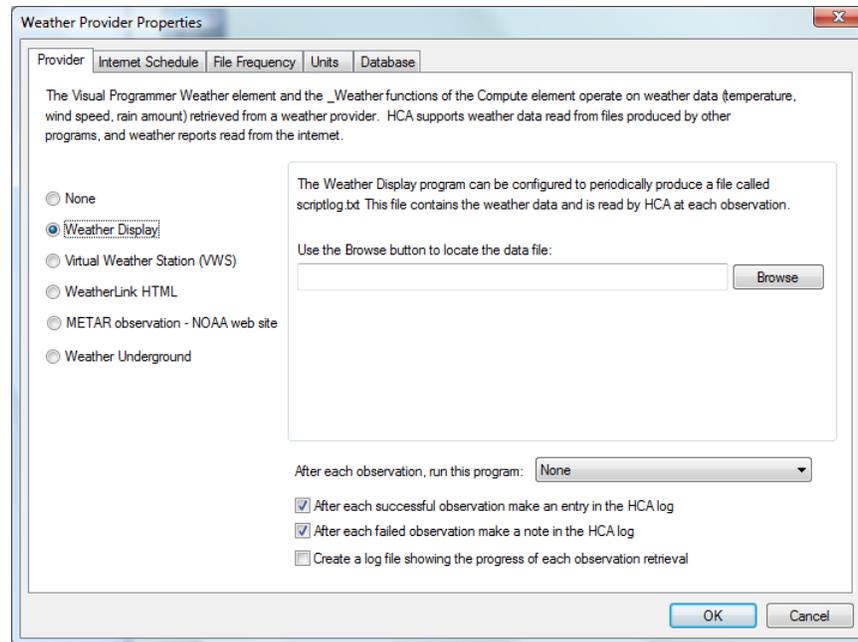


How often the file is read should approximate how often the file is produced by the program creating it.

When working with the *WeatherLink* program you should also set the observation units and working units on the *Units* tab as described above in the Weather Underground section.

## Weather Display Program

The Weather Display program periodically writes a file containing the current weather sensor data. When you select this option the setup dialog settings are:



All that is needed is the path to the file produced by *Weather Display*.

To configure *Weather Display* to produce the export file, it is all done from Weather Display menu selections. Select from the *Weather Display* menu:

```
Setup
Log file recording
Produce a 1 line log output for scripting (scriptlog.txt)
Yes
```

The exported file, scriptlog.txt, is placed in the *Weather Display* installation log files folder.

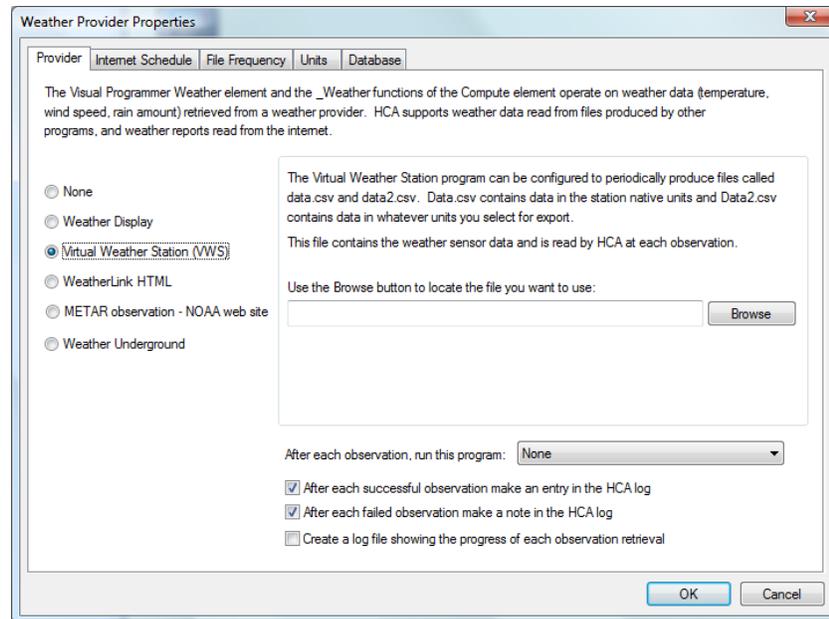
**Hint:** The method for producing scriptlog.txt may have changed in the current version of Weather Display. Refer to the *Weather Display* documentation for current instructions.

When working with the *Weather Display* program you should also set the observation units and working units on the *Units* tab as described above in the Weather Underground section.

When working with the *Weather Display* program you should also set the observation frequency as described above in the WeatherLink section.

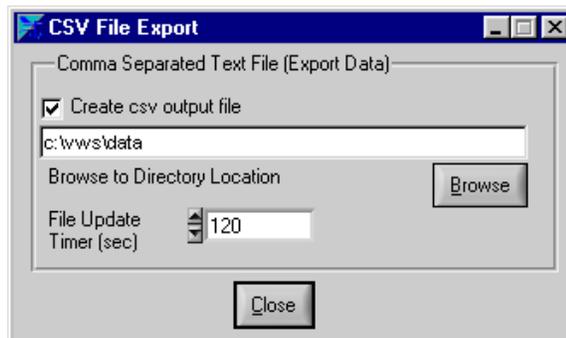
## Virtual Weather Station Program

The *Virtual Weather Station* program periodically writes a file containing the current weather sensor data. When you select this option the setup options are:



All that is needed is the path to the file produced by VWS.

To configure VWS to periodically create the data.csv and data2.csv files, start VWS and select from the VWS menu: Settings then CSV Export. This dialog appears:



Make sure that you check the *Create csv output file* option and choose a folder for where the file is produced and how often. Don't use too short an update time or HCA will never get access to the file – VWS will always have it open. Then press *Close*.

**Hint:** The method for producing scriptlog.txt may have changed in later versions of *VWS*. Refer to the *VWS* documentation for current instructions.

When working with the *VWS* program you should also set the observation units and working units on the *Units* tab as described above in the Weather Underground section.

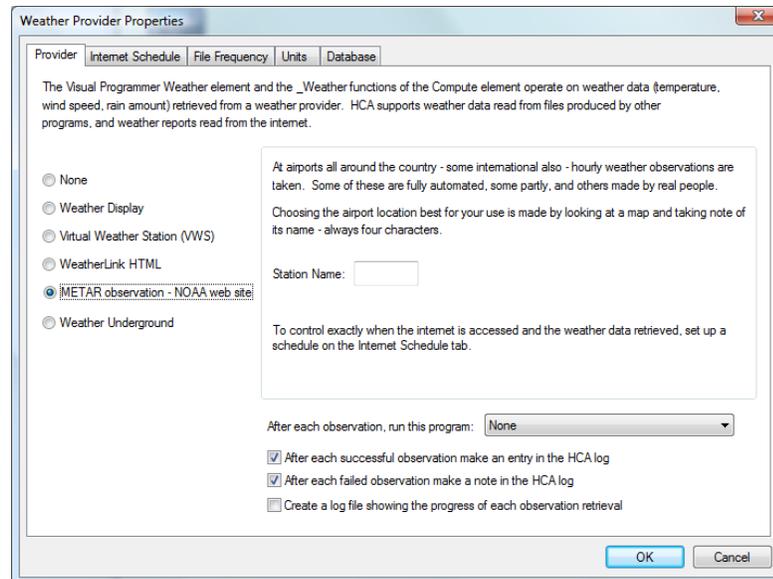
When working with the *VWS* program you should also set the observation frequency as described above in the WeatherLink section.

## METAR weather data

METAR is a format for reporting weather information. A METAR weather report is predominantly used by pilots in fulfillment of a part of a pre-flight weather briefing, and by meteorologists, who use aggregated METAR information to assist in weather forecasting. HCA can also retrieve and decode METAR reports.

Before using METAR data, the first thing you need to do is to locate the nearest airport that has a weather reporting station. The best way to do this is to use an internet search to find such an airport. What you are looking for is the 4 character airport code.

When you choose the METAR option the setup dialog contains these configuration options:



When working with METAR data you should also set the observation units and working units on the *Units* tab as described above in the Weather Underground section.

When working with METAR data you should also set the observation frequency as described above in the Weather Underground section.

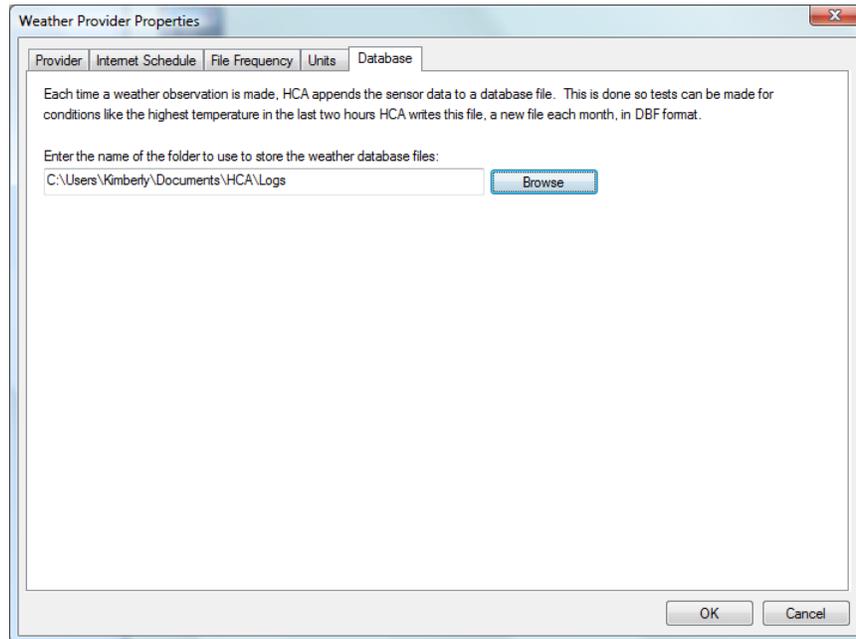
**Hint:** METAR observations seem to make it to the internet sometime between 50 minutes after and the top of the hour. A good time to retrieve data is 10 or 15 minutes after the hour.

---

## Weather database

Based upon the configured schedule HCA take a weather observation and writes data to a file. This is necessary to support testing on weather data over time. For example, the database is used when testing to see if the average temperature in the last 24 hours has been above or below freezing.

To setup a weather database, choose the *Database* tab in the weather provider setup.

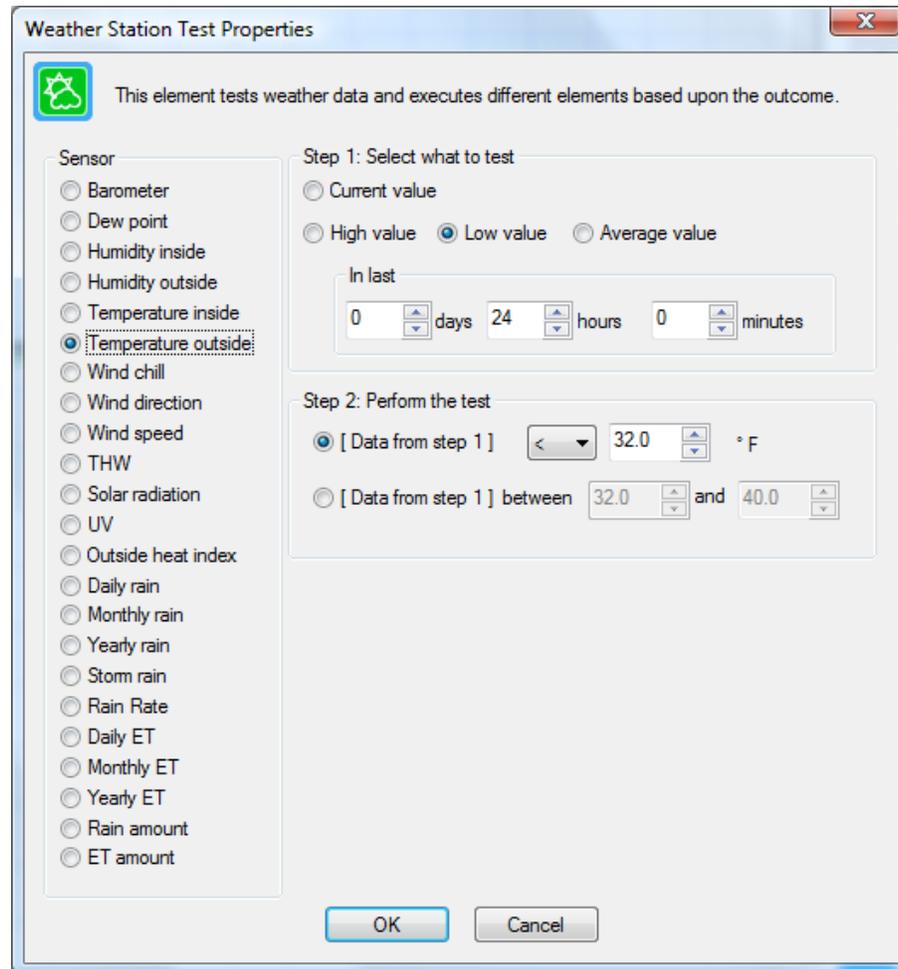


The only information to enter is the path to a folder that contains the monthly log files. What you enter here is the path to a folder and *not* the path to a file. Each monthly log file is given a name by HCA.

**Tip:** Not all weather data is saved in the weather database. On the HCA support web site is a technical note on Weather providers. The technical note describes exactly what parts of an observation are saved in the database and which can be tested with the historical weather tests.

## Visual Programmer

To use weather data to make decisions in your HCA programs, the *Weather Test* element is used. This visual programmer element makes it possible to test both current weather data and historical weather data. When added to a program, the element properties are:



The parts of this dialog are:

- What data item to test. Not all weather providers support all data items.
- What to test. Either the current value taken from the last weather observation or a value determined from the weather database.
- The test. You can test the data to be  $<$ ,  $<=$ ,  $>$ ,  $>=$ ,  $=$ ,  $<>$  to a value or within a range of values.

In the above example the test is to see if the low outside temperature over the last 24 hours is less than 32 degrees F.

Like other Test elements, execution continues following the path marked *Yes* if the test is true, otherwise execution follows the *No* path.

## Weather Triggers

Weather data can be used to trigger a program when an observation is taken and the values of weather data meet certain criterion.

To create a weather trigger for a program, select the *Triggers* tab of the program's properties, select *Weather Condition Change* and press *Add*.

The trigger creation dialog shows these options.

This dialog looks a lot like the properties for the visual programmer *weather* element described above. As described in the User Guide *Programs* chapter, HCA evaluates all weather triggers every minute and when the condition evaluates to Yes, the program starts. Before the program can start again, the weather condition must evaluate to No and then to Yes again. For more information on how these kinds of triggers are evaluated and programs started see the User Guide *Programs* chapter.

## Weather Data and the Compute and Compute Test elements

In addition to examining weather data with the Visual Programmer *Weather* element and with weather triggers, there are also functions that can be used in expressions in the Compute, Compute Test, and in expression placeholders in elements that work with text strings. Refer to the User Guide expressions chapter for information on the `_Weather` functions.

## Weather Underground

In addition to the *Weather Test* Visual Programmer element and *\_Weather* function in expressions, an additional function has been added in support of Weather Underground. The function allows you to get deeper into all the data provided by Weather Underground.

The function is called *\_WUnderground* and it takes a minimum of 4 arguments and a max of 10. The format is:

```
_Wunderground ("feature", "location", "path1", "path2", ... "pathn", "tag")
```

The “feature” – 1<sup>st</sup> argument - is one of these on this list:

<http://www.wunderground.com/weather/api/d/docs?d=data/index>

The location – 2<sup>nd</sup> argument - is any acceptable location as described on that page in the “query” section. You can use “” for the feature and it defaults to “*conditions*”. You can use “” for location and it defaults to whatever you entered as the location in the weather provider dialog setup for Weather Underground.

Using the *\_Wunderground* function you can get to **anything** that weather underground provides. For an example, enter this into a browser:

<http://api.wunderground.com/api/<your api key>/forecast/q/<your zipcode>.xml>

Substitute your API key and location into this URL. This is similar to what you will see:

```

▼<response>
  <version>0.1</version>
  ▼<termsofService>
    http://www.wunderground.com/weather/api/d/terms.html
  </termsofService>
  ▼<features>
    <feature>forecast</feature>
  </features>
  ▼<forecast>
    ▼<txt_forecast>
      <date>7:00 AM PST</date>
      ▼<forecastdays>
        ▼<forecastday>
          <period>0</period>
          <icon>rain</icon>
          <icon_url>http://icons-ak.wxug.com/i/c/k/rain.gif</icon_url>
          <title>Wednesday</title>
          ▼<fcttext>
            ▼<![CDATA[
              Overcast with rain. High of 45F. Breezy. Winds from the South at 15 to 20 mph. Chance of rain 90%.
            ]]>
          </fcttext>
          ▼<fcttext_metric>
            ▼<![CDATA[
              Overcast with rain. High of 7C. Windy. Winds from the South at 25 to 30 km/h. Chance of rain 90%.
            ]]>
          </fcttext_metric>
          <pop>90</pop>
        </forecastday>
      ></forecastday>...</forecastday>

```

As you can see this is forecast data for your location. To extract the forecast string for today the path though this XML is:

```
Response - forecast - txt_forecast - forecastdays - forecastday - fcttext
```

To make it easier to see, this image has that path highlighted.

```

▼<response>
  <version>0.1</version>
  ▼<termsOfService>
    http://www.wunderground.com/weather/api/d/terms.html
  </termsOfService>
  ▼<features>
    <feature>forecast</feature>
  </features>
  ▼<forecast>
    ▼<txt_forecast>
      <date>7:00 AM PST</date>
      ▼<forecastdays>
        ▼<forecastday>
          <period>0</period>
          <icon>rain</icon>
          <icon_url>http://icons-ak.wxug.com/i/c/k/rain.gif</icon_url>
          <title>Wednesday</title>
          ▼<fcttext>
            ▼<![CDATA[
              Overcast with rain. High of 45F. Breezy. Winds from the South at
            ]]>
          </fcttext>
          ▼<fcttext_metric>
            ▼<![CDATA[
              Overcast with rain. High of 7C. Windy. Winds from the South at 2:
            ]]>
          </fcttext_metric>
          <pop>90</pop>
        </forecastday>
      ></forecastday>...</forecastday>

```

Extracting the forecast string using the `_Wunderground` function uses these arguments:

```
Forecast = _wunderground ("forecast", "", "forecast", "txt_forecast",
                          "forecastdays", "forecastday", "fcttext")
```

The first argument to `_wunderground` is the *feature*. In this case “forecast”. The second argument is the location. Since it is set to “” the location as set in the weather provider setup is used. The remaining arguments are the path to the data using the highlighted names shown above.

Given the above XML, and the function with those arguments, it evaluates to this string:

```
“Overcast with rain. High of 45F. Breezy. Wind from the South at 15 to 20 mph. Chance of rain 90%.”
```

Instead of the forecast, if you wanted to get the “icon” the path would be:

```
Response - forecast - txt_forecast - forecastdays - forecastday - icon
```

In addition to weather data, Weather Underground offers lots of other data as well. For example, to get the current moon phase percent use:

```
Moon = _wunderground ("astronomy", "", "moon_phase", "percentIlluminated")
```

There is one complication with some keys: In this example XML below there are multiple “forecastDay” branches:

```

▼<forecast>
  ▼<txt_forecast>
    <date>7:00 AM PST</date>
    ▼<forecastdays>
      ▶<forecastday>...</forecastday>
      ▶<forecastday>...</forecastday>
      ▶<forecastday>...</forecastday>
      ▶<forecastday>...</forecastday>
      ▶<forecastday>...</forecastday>
      ▶<forecastday>...</forecastday>
      ▶<forecastday>...</forecastday>
      ▶<forecastday>...</forecastday>
    </forecastdays>
  </txt_forecast>

```

As an example, to get data for a specific day, include a qualifier in the name of the tag – note the “:3” which requests the 3<sup>rd</sup> *forecastday*.

```
Forecast = _wunderground ("forecast", "", "forecast", "txt_forecast",
                          "forecastdays", "forecastday:3", "fctext")
```

This retrieves the 3rd forecast day. To get the first forecast day use “forecastday:1”. To find the second use “forecastday:2”, etc. If you leave off the qualifier it defaults to 1.

A good suggestion is to try out the various retrievals in your browser first then look at the data and figure out the path you need to get at that data in HCA. Pay close attention to any paths that will need qualifiers.

### Two more important points when using Weather Underground data

To prevent multiple queries to Weather Underground each time a *\_wunderground* function is evaluated, the file is retrieved, parsed, and saved internally along with the date/time that it was retrieved. Subsequent *\_wunderground* functions don’t retrieve the data again from the internet if the XML for that feature was retrieved less than 15 minutes ago. This way you can extract lots of data and do it quickly since only one internet action is needed.

And finally, the interactions with the Weather Underground site are done using HTTP so there should be no firewall issues.

## Appendix 5

### IR Interfaces

This appendix describes HCA support for two supported IR Interfaces: the Global Caché GC-100 and the Bitwise BC4. These network devices – they are located by an IP address – contain IR output ports. Some units also contain digital sensor ports, relays, and serial ports.

HCA supports the IR output ports and using them can send IR sequences to control TVs, Audio, These sequences come from the stored database inside the Bitwise BC4, and when using the GC100, from sequences captured by the Global Caché IR Learner.

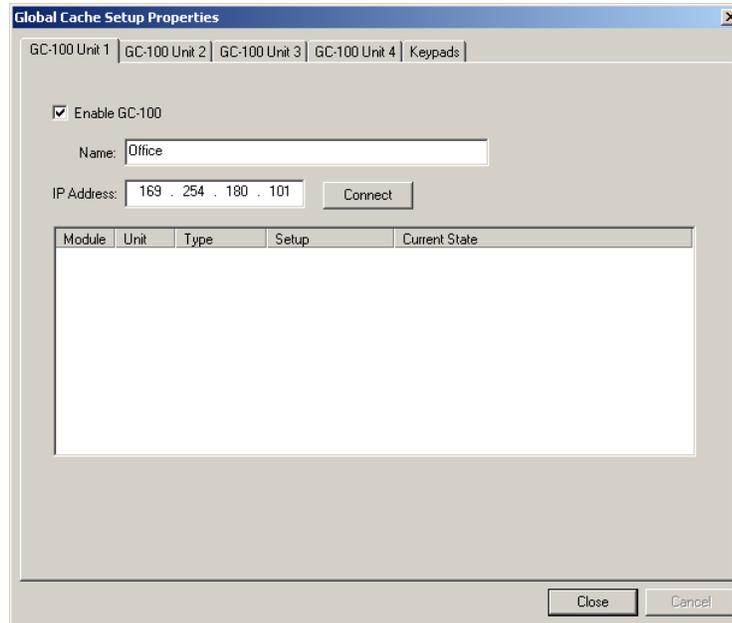
When using the GC-100, digital inputs are supported and can be polled for status or, if they are set up as notify ports, used to start HCA programs. The GC-100 relays are also supported.

This appendix describes

- GC-100
  - GC-100 Setup
  - GC-100 IR outputs
    - IR Learner
    - IR Keypad builder
    - Creating an HCA device for an IR Output Port
    - IR sequences in programs
    - IR sequences in schedules
  - GC-100 Digital sensor ports
    - Creating a HCA device for a sensor port
    - Using a GC-100 sensor port for program triggers
  - GC-100 Relays
    - Creating a HCA device for a relay
    - Using a GC-100 relay
  - Design Inspector for the GC-100
- Bitwise BC4
  - Bitwise BC4 Setup
  - Bitwise BC4 device type selection
  - IR Keypad builder
  - Creating an HCA device for an IR Output Port
  - IR sequences in programs
  - IR sequences in schedules

## GC100- Setup

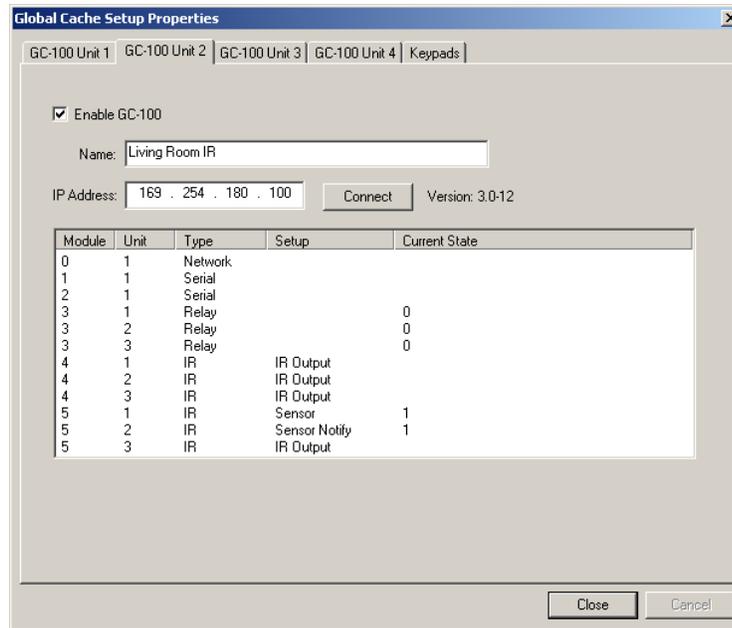
To set up the GC-100, press the *GC-Setup* button in the ribbon *Interfaces* menu and select *Configure* from the menu.



HCA can utilize up to four GC-100s. The first 4 tabs of this dialog are for configuration of them. Each tab contains:

- Checkbox to enable or disable use of this GC\_100
- Name of the GC-100. In other areas of HCA, you refer to the different GC-100s that you have configured by this name
- The IP address of the unit. The GC-100 has many different mechanisms for determining an IP address. HCA supports only a static IP address. This IP address is set using the software available from Global Caché or the internal web server configuration pages in the GC-100.
- A display of how the various ports are configured.

After entering the name and IP address, press the connect button. The configuration of the GC-100 displays:



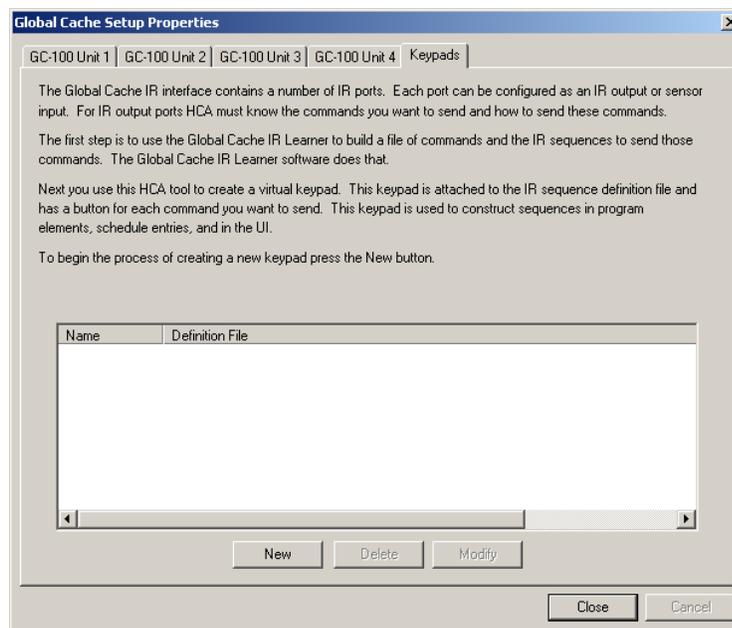
Configuration of the ports is done using the software from Global Caché or the internal web pages of the GC-100. While HCA can read out the setup it does not provide facilities to configure the ports.

---

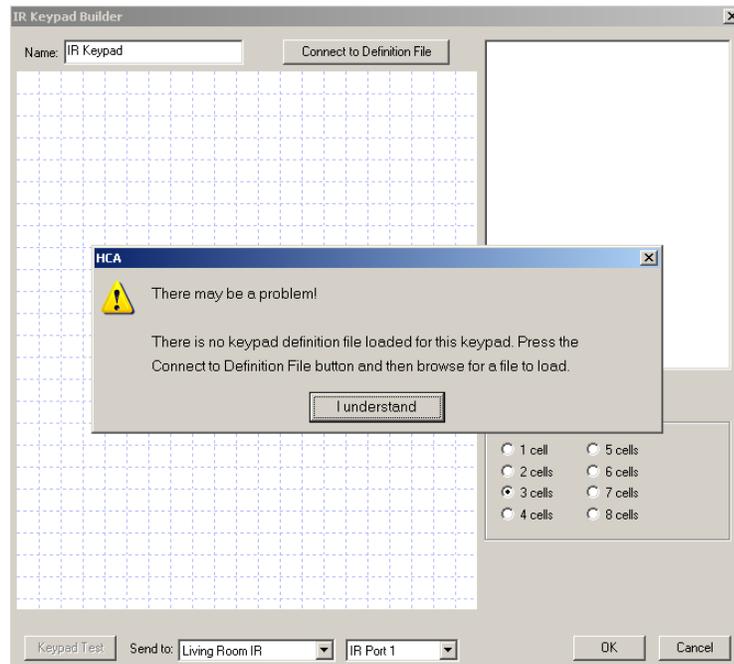
## IR Keypad builder

The next step in getting ready to use an IR Output Port is to create an IR keypad. The keypad is used to enter the sequences you want to send from programs and schedules.

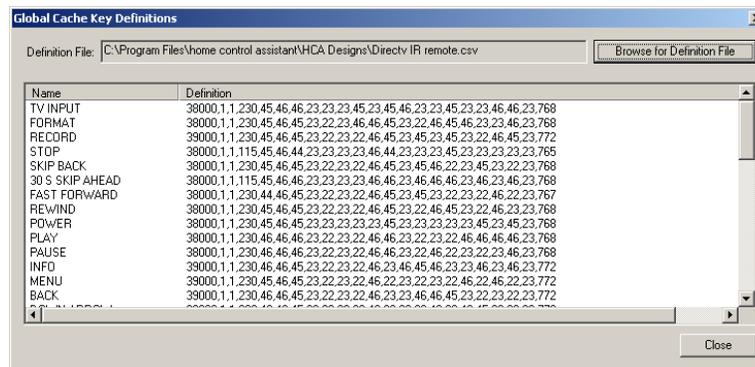
Press the *GC-100 setup* button in the ribbon *Interfaces* category and select *Configure* from the menu. Then choose the *Global Cache keypads* tab.



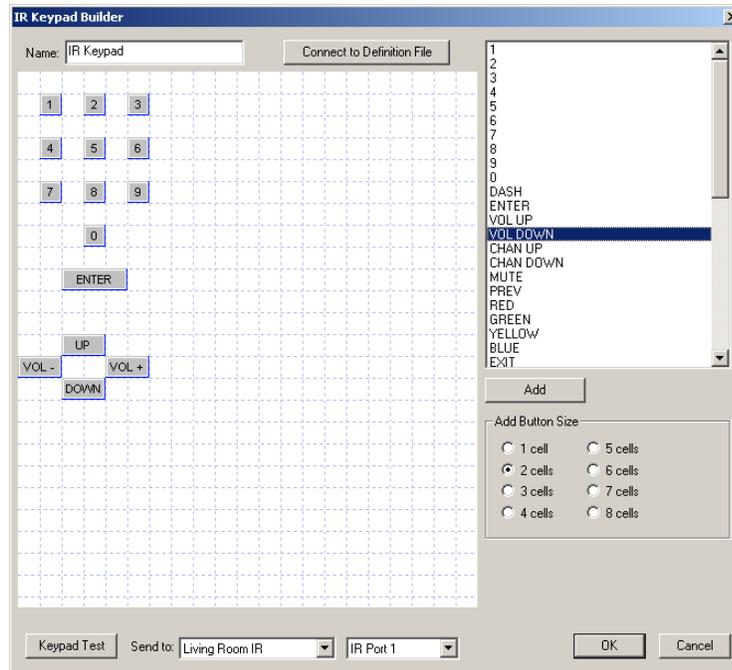
From the dialog you can create, delete, or edit IR keypads. To create a new Keypad press the New button.



As the popup message points out, the first action you need perform is to connect this keypad to a definition file. Pressing the *Connect to Definition File* button displays this dialog. Press the *Browse* button to locate the IR definition file you created with the IR Learner.



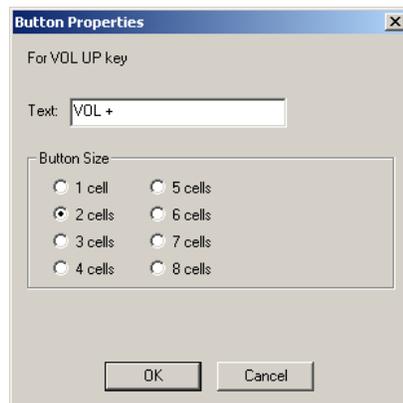
After the file is loaded, the key sequences display. Closing this dialog places you back in the Keypad Builder.



This dialog looks a bit like the Visual Programmer. On the canvas are placed the keypad buttons. In the upper right of the dialog is a list of all the IR keys from the definition file.

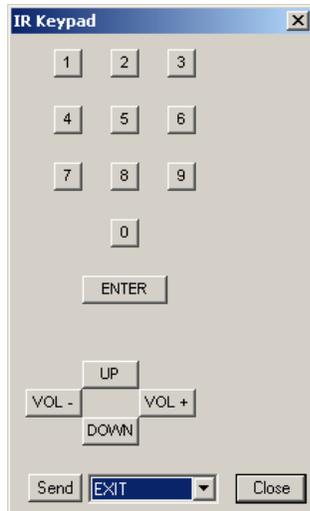
First, select a key that performs some action that you want. Then select the button size and press Add, or like the Visual Programmer, select the code in the list and drag it on to the canvas.

Once on the canvas you can then drag the buttons to whatever location you want. To change the label on the button or the button size, double click the button to open its properties.



In the above example, several buttons have different names than listed in the keypad button list (Vol+ instead of Vol Up). This is because the name has been modified by opening the button properties and entering a new name.

When you have placed all the buttons you want on the keypad, you can test it using the *Keypad Test* button at the bottom of the keypad builder dialog. Before starting the test, select the GC-100 and port you want to use.



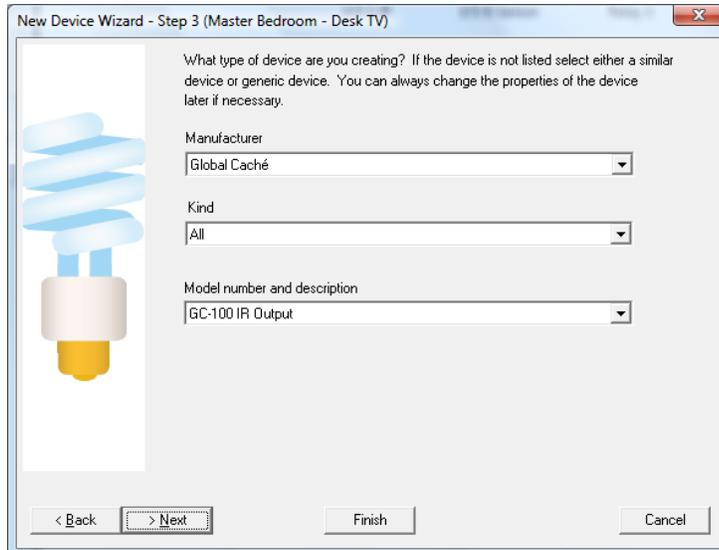
From the test keypad, pressing a button causes the IR sequence to be transmitted by the selected GC-100 out the selected port.

On HCA IR Keypads, any key sequence from the definition file that you didn't create a button for can still be sent using the dropdown and *Send* button at the bottom.

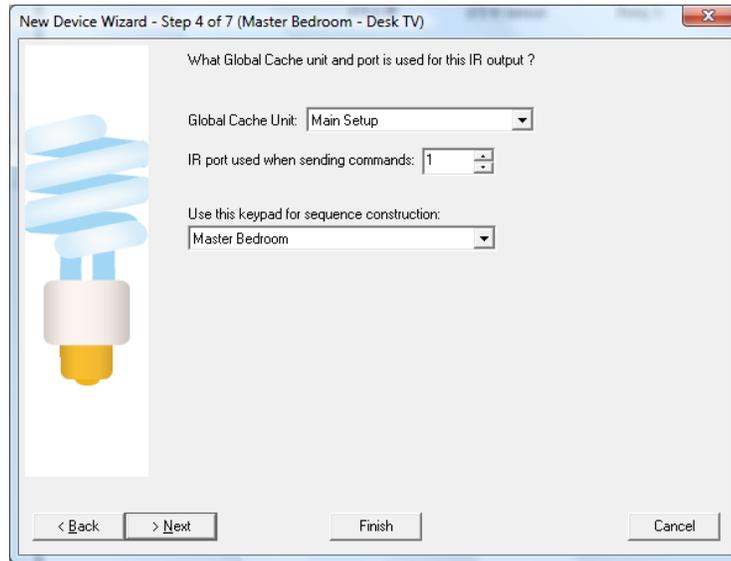
---

### Creating a HCA device for an IR Output Port

Now that the IR definition file has been created and associated with a keypad, you can create the HCA device to represent the IR output port. In step 3 of the New Device Wizard, select as the manufacturer *Global Caché* and type *GC-100 IR Output*.



On step 4 the specifics of this port are configured.



You select:

- Which GC-100 unit and port this device represents. The GC-100 is selected by the name you entered when it was identified to HCA in the *IR – Global Caché GC-100 Setup* dialog.
- Which IR keypad to associate with this device. Keypads are selected by the name you used when you created it in the keypad builder.

The remainder of the New Device Wizard is the same as for all other devices.

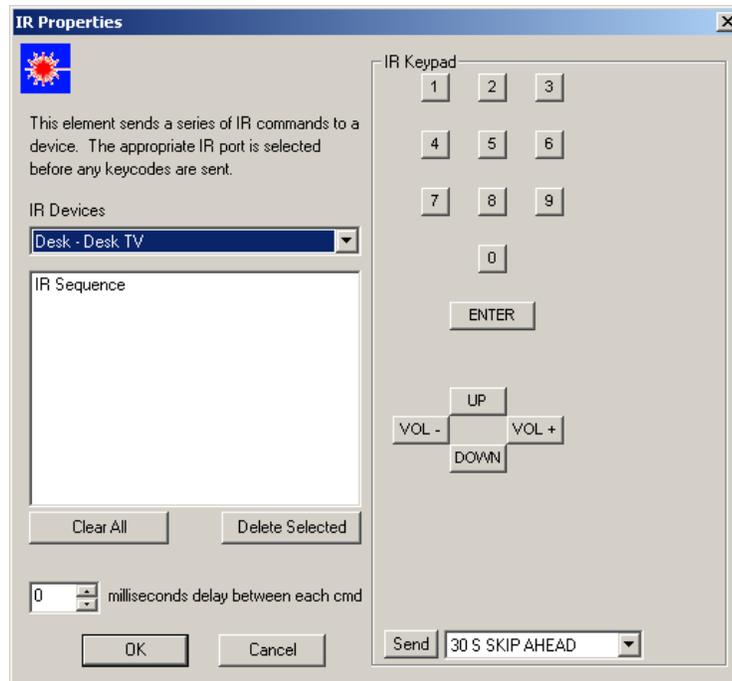
**Hint:** Remember that the ports on the back of the GC-100 can be configured as IR Outputs or sensor inputs. HCA doesn't have the ability to change their use – you must use the GC-100 software for this.

**Hint:** The GC-100 ports are labeled on the back of the GC-100 starting at 1. HCA uses the same numbering.

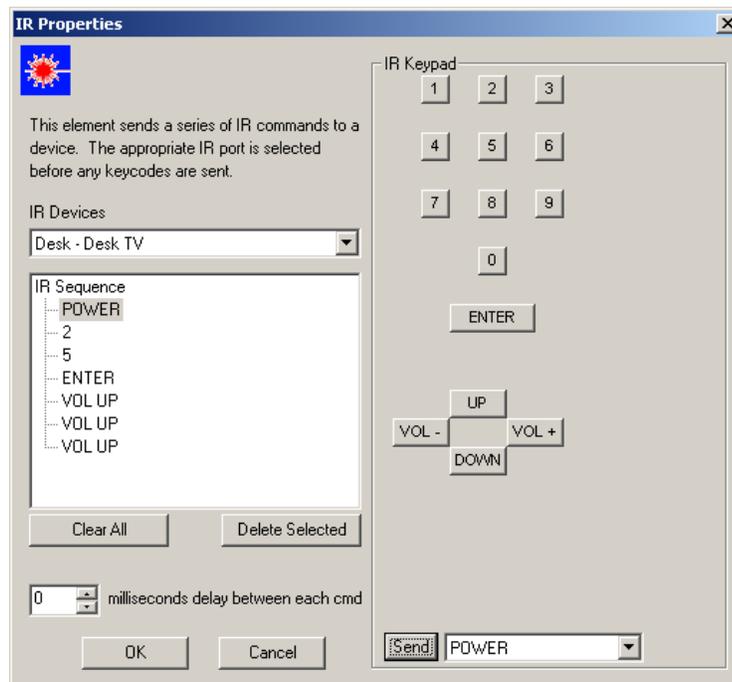
**Hint:** Rather than name the device with the GC-100 name and port number you may want to name the device with what it controls. For example, you may want to use Desk TV, Home DirecTV tuner, etc.

## IR sequences in programs

Now that all the setup steps are complete you can finally use the IR port in a program. The IR element is used for this:



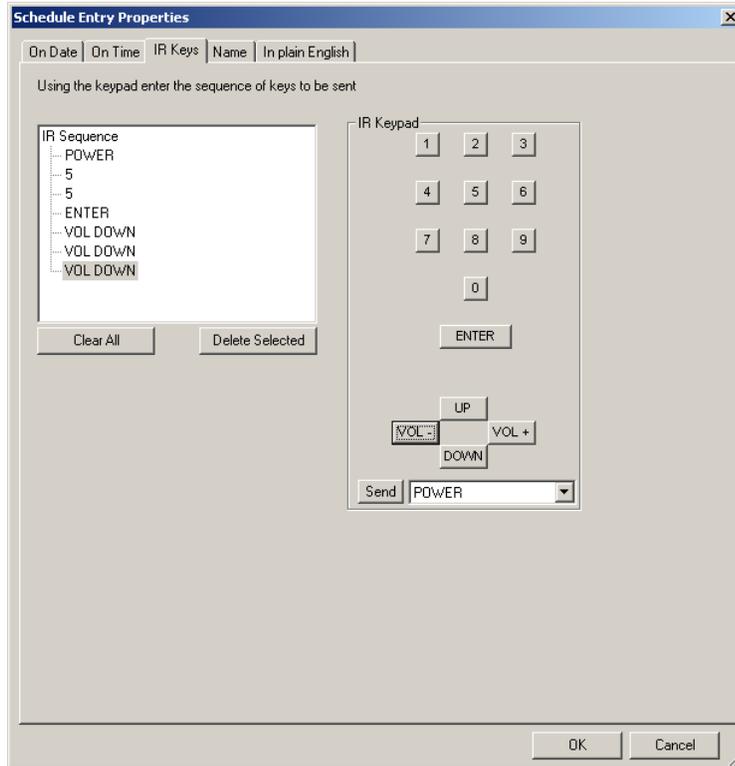
Use the keypad to enter the sequence you want to transmit.



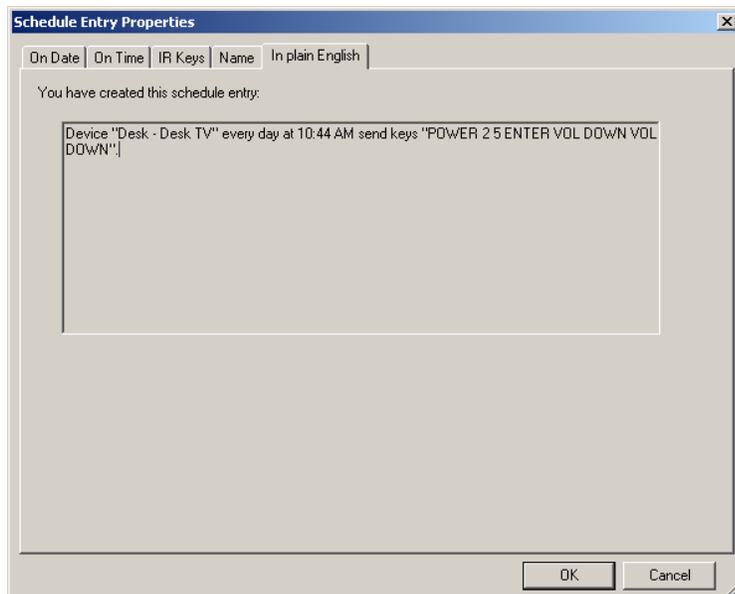
If you want to delete some keys in the sequence, select and use the *Delete Selected* Button. New entries are always inserted after the selected item so it is easy to enter missed keys in the middle of a sequence.

## IR sequences in schedules

Creating a schedule entry that sends an IR sequence is very similar to the IR Visual Programmer element. The schedule entry wizard has a step for IR sequences if you select an IR device. Also the Visual Schedules has properties for IR time bar markers.



Enter the sequence you want to transmit in the same manner as the Visual Programmer. The final tab (or wizard step) of the schedule entry shows in text the sequence to be transmitted.



---

## GC-100 Digital sensor ports

The GC\_100 has a number of ports that can be used as IR Output ports and as digital inputs. Available from Global Caché video, voltage, and contact closure sensors that can be plugged in to these ports for sensing when things are on and off. Regardless of the sensor used, all the ports report in the same manner – a simple ON (or one) of OFF (or zero).

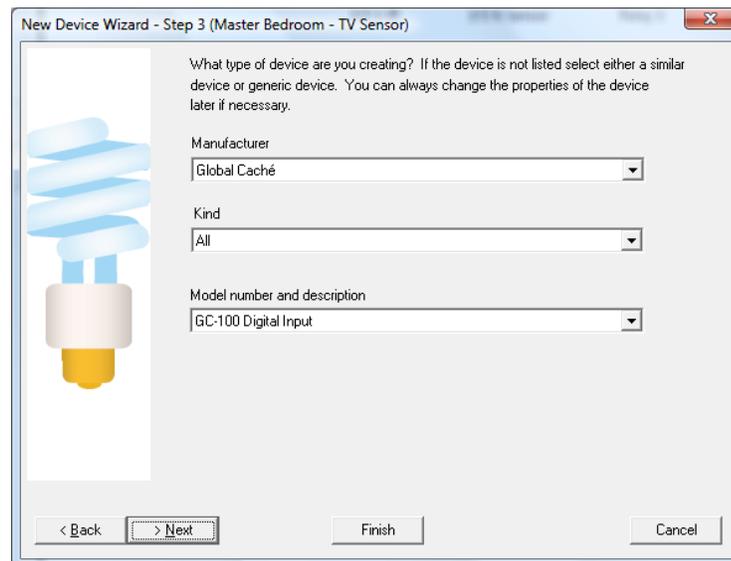
These can be set up using the Global Caché software in one of two ways:

- Sensor – the port can be polled for its current state and returns one or zero
- Sensor Notify – The GC-100 sends a message to HCA each time the port changes from zero to one or one to zero. In addition, the port can be polled like the Sensor port to get its current value.

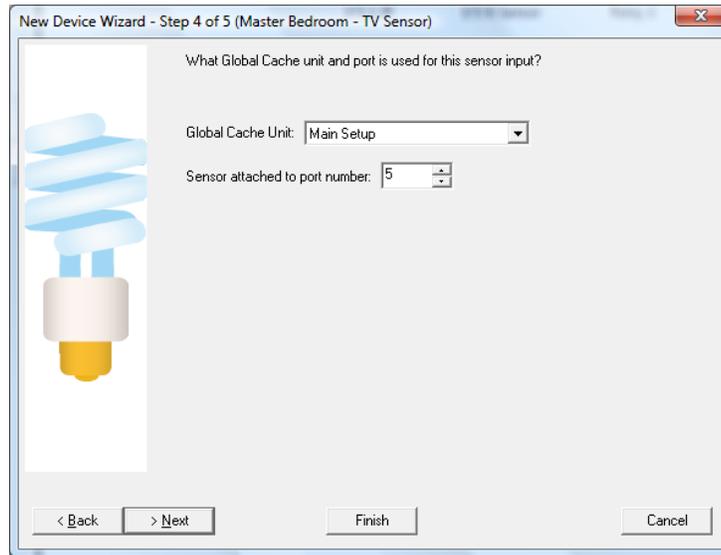
---

## Creating a HCA device for a sensor port

GC-100 sensor ports are added using the New Device Wizard. The first steps of the wizard are as usual. On step 3, choose the type as *GC-100 Digital Input*



On step 4 select the GC-100 and the port:



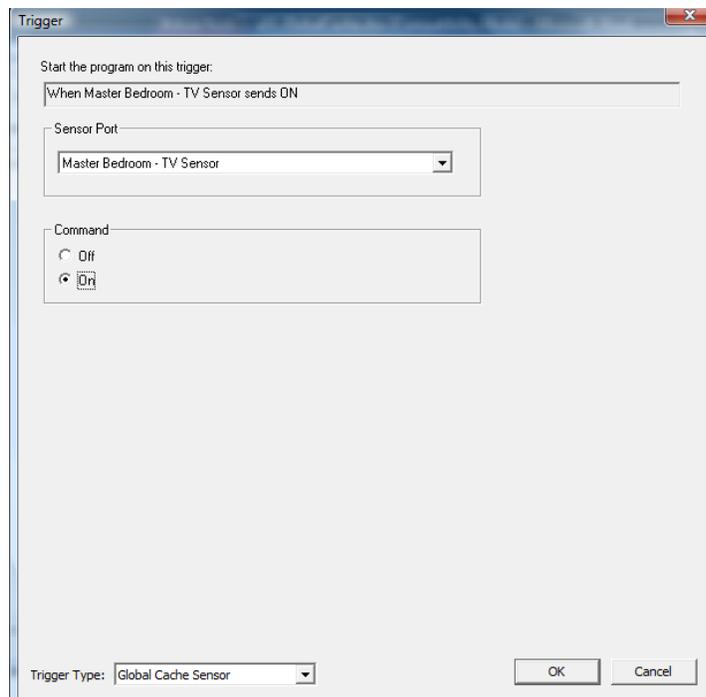

---

### Using a G-100 sensor port for program triggers

Once you have added a digital input it can be used in one of two ways. You can use the Test element with the *Is Device On* and *Is Device Off* test conditions. When that element is executed, the GC-100 port is polled to determine the outcome of the test.

If the port was configured as Sensor Notify then you can create program triggers. When the messages about port state changes come from the GC-100, programs can be started.

On the Program properties dialog choose the *Triggers* tab. As the type of trigger choose *Global Caché Sensor*.



Select the GC-100 sensor and if the program should be triggered when the sensor sends an ON or an OFF.

---

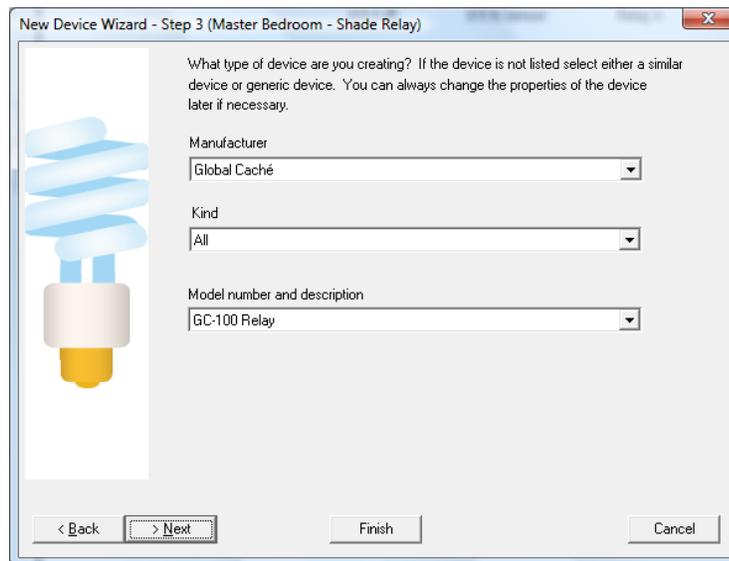
## GC-100 Relays

In addition to the GC-100 IR output ports and Digital Input ports, some GC-100 models have relays. These can be used with HCA after a device is created for them. When the relay is *closed* HCA says it is *On*. When the relay is *open*, HCA says it is *Off*.

---

### Creating a HCA device for a relay

To create a GC-100 relay, start the New Device Wizard. On step 3 select as the type *GC-100 relay*:



New Device Wizard - Step 3 (Master Bedroom - Shade Relay)

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.

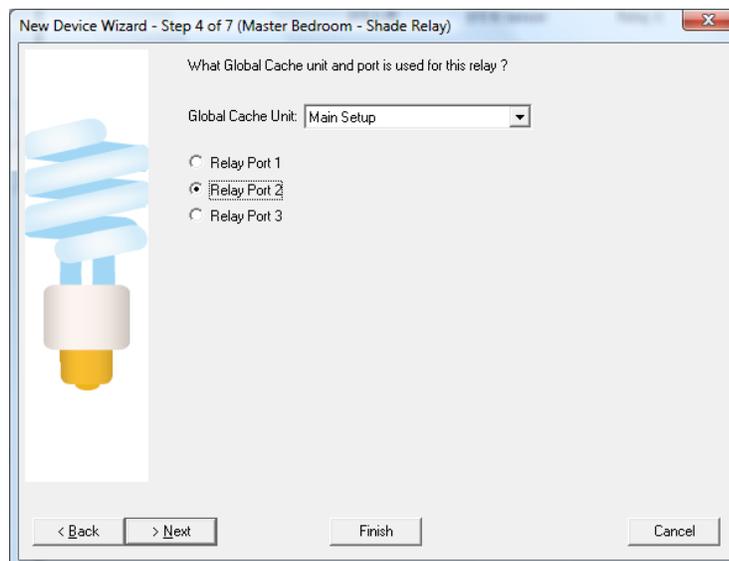
Manufacturer: Global Caché

Kind: All

Model number and description: GC-100 Relay

< Back   > Next   Finish   Cancel

On the next step of the wizard enter the GC-100 unit and relay number.



New Device Wizard - Step 4 of 7 (Master Bedroom - Shade Relay)

What Global Cache unit and port is used for this relay ?

Global Cache Unit: Main Setup

Relay Port 1  
 Relay Port 2  
 Relay Port 3

< Back   > Next   Finish   Cancel

**Hint:** The GC-100 relays are numbered on the back of the GC-100. HCA uses the same numbering.

---

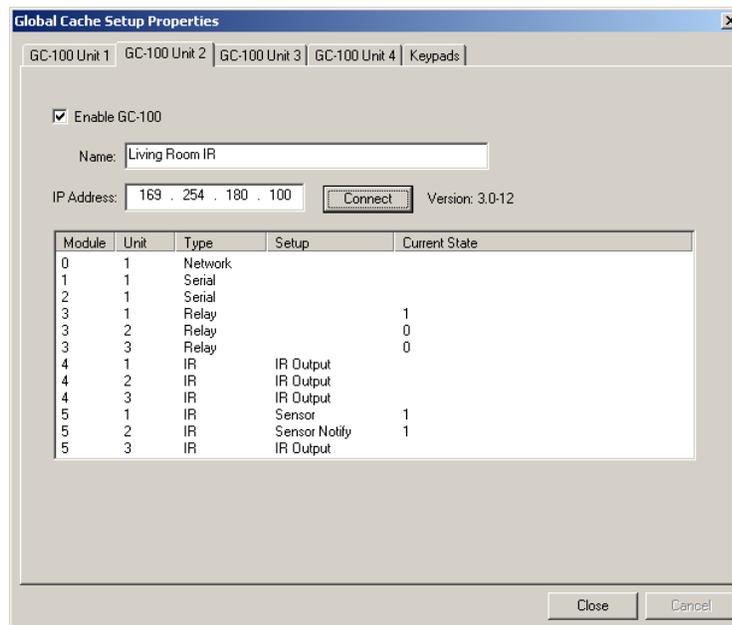
## Using a GC-100 Relay

The GC-100 relay can be used as any other non-dimmable device in HCA. You can control it from the UI, schedules, and programs. The relay also responds to starts requests so it can be used in the Visual Programmer Test element and HCA asks the GC-100 for it's current state – On (closed) or Off (open).

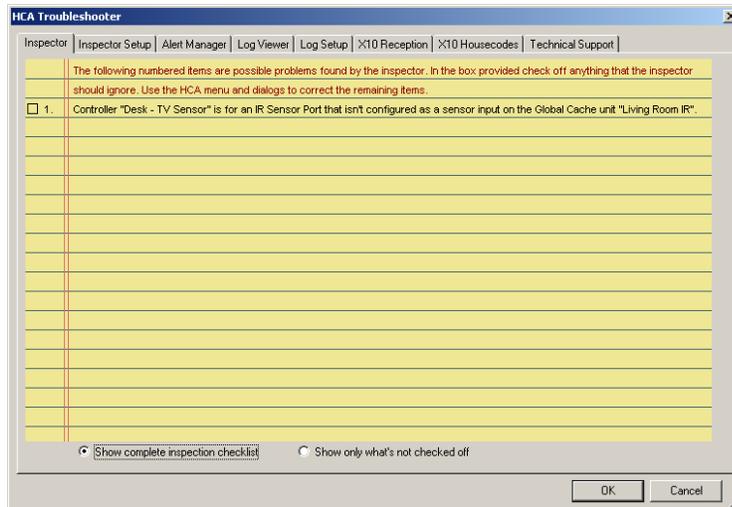
---

## Design Inspector

With all the relays, IR output ports, digital input ports – both Sensor and Sensor Notify – it can be easy to become confused. HCA helps in two areas. First you can always open the GC-100 setup dialog and press the connect button. HCA shows the current setup of the GC-100, how the ports are set up, and their current values.



Also, the design inspector (part of the troubleshooter) looks at all your usages of the GC-100 and tries to detect problems. For example, creating a program trigger using a port not programmed as Sensor Notify; or using the IR Visual Programmer element with a port that isn't set up as an IR Output.

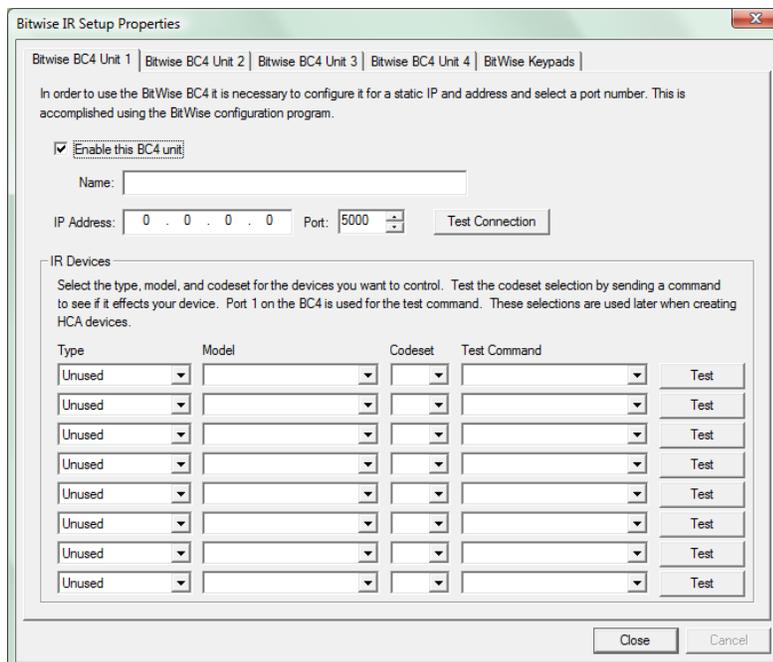


## BC4 Setup

To utilize IR devices in HCA programs and schedule sit first is necessary to perform several actions first. These are:

1. Program the BC4 for the kinds of devices you have. A device kind is specified by what it is – TV, DVD, etc – the model and codeset.
2. Create a keypad for a device kind to enable specification of IR sequences.
3. Create a device in HCA using the New Device Wizard that can then be used for scheduling and programming.

To set up the BC4, press the *Bitwise setup* from the ribbon *Interfaces* category then select the *Configure* from the menu.



HCA can utilize up to four BC4s. The first 4 tabs of this dialog are for configuration of them. Each tab contains:

- Checkbox to enable or disable use of this BC4
- Name of the BC4. In other areas of HCA, you refer to the different BC4s that you have configured by this name
- The IP address of the unit. The BC4 has many different mechanisms for determining an IP address. HCA supports only a static IP address. This IP address is set using the software available from BC4 or the internal web server configuration pages in the BC4.
- A display of the kinds of IR devices you will control with this BC4.

After entering the name and IP address, press the connect button. If the connection is successful “Connected ok” displays.

Bitwise IR Setup Properties

Bitwise BC4 Unit 1 | Bitwise BC4 Unit 2 | Bitwise BC4 Unit 3 | Bitwise BC4 Unit 4 | BitWise Keypads

In order to use the BitWise BC4 it is necessary to configure it for a static IP and address and select a port number. This is accomplished using the BitWise configuration program.

Enable this BC4 unit

Name:

IP Address:  Port:   Connected ok

IR Devices

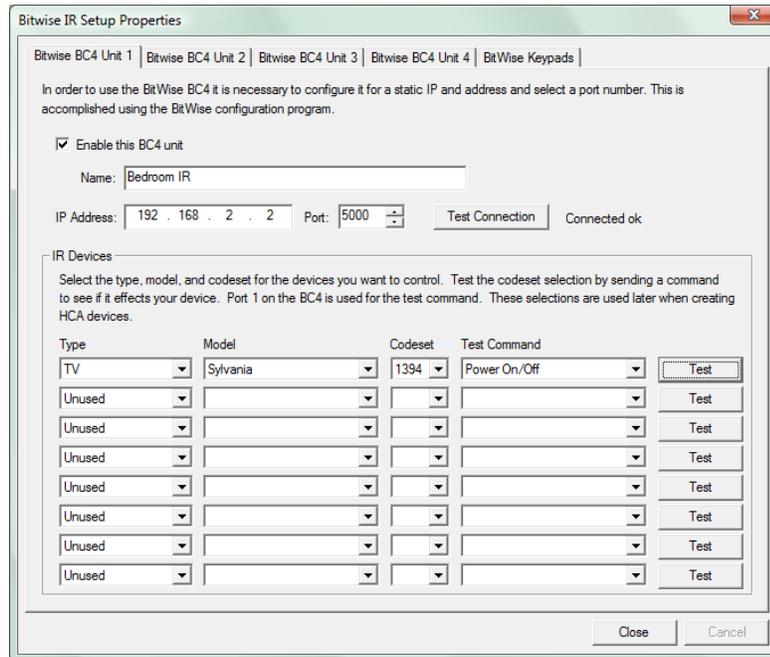
Select the type, model, and codeset for the devices you want to control. Test the codeset selection by sending a command to see if it effects your device. Port 1 on the BC4 is used for the test command. These selections are used later when creating HCA devices.

Type	Model	Codeset	Test Command	
Unused				<input type="button" value="Test"/>
Unused				<input type="button" value="Test"/>
Unused				<input type="button" value="Test"/>
Unused				<input type="button" value="Test"/>
Unused				<input type="button" value="Test"/>
Unused				<input type="button" value="Test"/>
Unused				<input type="button" value="Test"/>
Unused				<input type="button" value="Test"/>

Once the connection is made the next step is to program the BC4 to enable it to control the kinds of devices you have.

## Bitwise BC4 device type selection

You can think of the BC4 like a universal remote control. You first have to tell it what kind of device you have: Is it a TV, DVD, Audio, etc? Then you select the model of that device type you have: Is it a Sony TV, Toshiba TV, Sylvania TV, etc? Lastly you have to select the codeset the device uses. This one is harder and the only way you can know what codeset to use is by testing.



In this example, the kind of device is a *TV*, its model is *Sylvania*, and the chosen codeset is *1394*.

How do you know the codeset is correct? Select one of the codeset choices – there may be one or there may be quite a few – then chose a command and press the *Test* button.

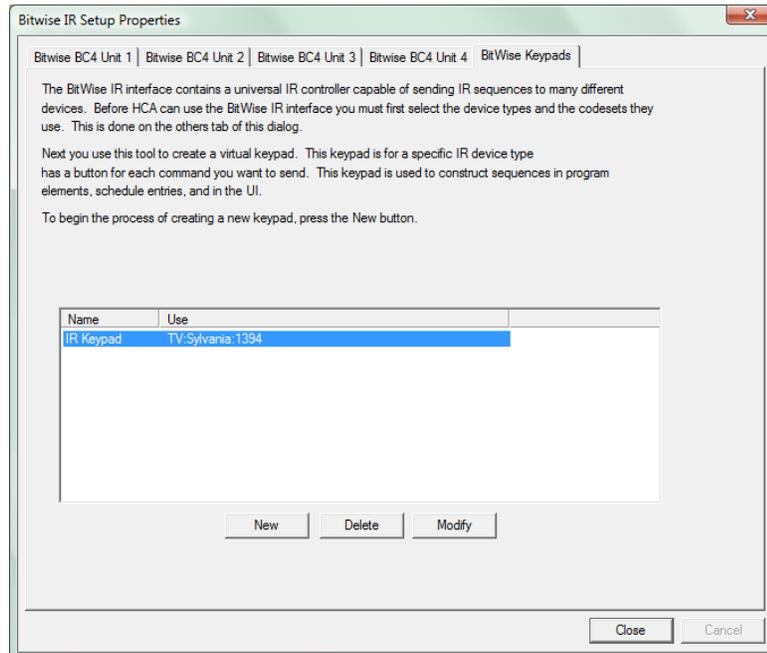
That command is then sent out of IR Port number 1 – they are labeled on the back of the BC4. Watch the device and see if it responds. If it does you have selected the correct codeset. If it doesn't responds as expected then select a different codeset and try again.

HCA limits you to programming the BC4 for up to 8 device kinds.

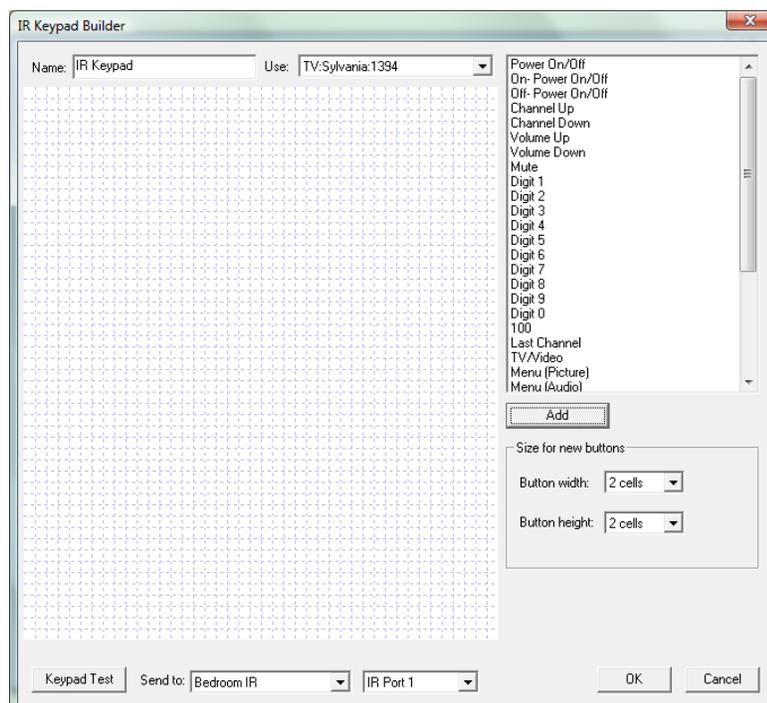
## IR Keypad builder

As outlines above, the next step in getting ready to use an IR device is to create an IR keypad. The keypad is used to enter the sequences you want to send from programs and schedules.

Press the *Bitwise setup* button in the ribbon *Interfaces* category and select *Configure* from the menu. Then choose the *Bitwise Keypads* tab.



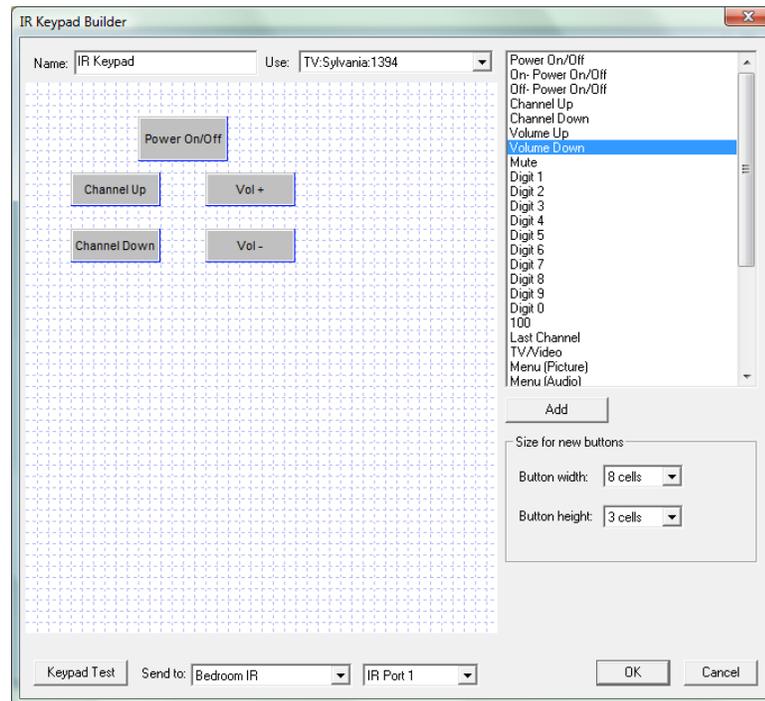
From the dialog you can create, delete, or edit IR keypads. To create a new Keypad press the New button.



This dialog looks a bit like the Visual Programmer. On the canvas are placed the keypad buttons. In the upper right of the dialog is a list of all the IR keys from the definition file.

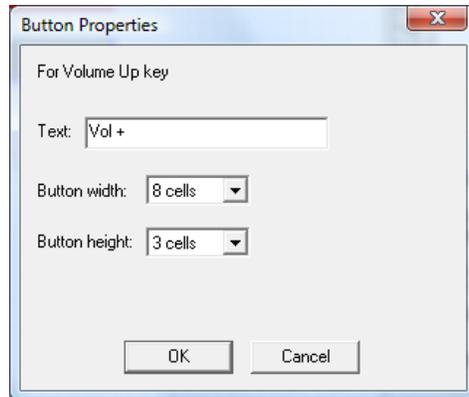
The major areas of this dialog are:

- The name of the keypad. You should choose a unique name for each keypad you create as you will use these names when you \*create HCA devices.
- One of the different device kinds you programmed the BC4 for. Select the one to use with this keypad.
- All the possible IR keys that you can use on this keypad. This list comes from the selection you made for the device kind, model, and codeset.
- The size of the buttons that you will create.
- A test area where you can select the BC4 unit and IR port. Pressing the *Keypad Test* button opens a test of the keypad.



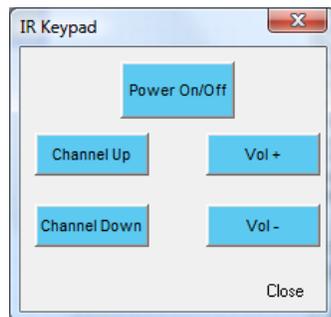
To add a new button to a keypad, select a key in the list of keys that performs the action that you want. Then select the button size and press Add, or like the Visual Programmer, select the key in the list and drag it on to the canvas.

Once on the canvas you can then drag buttons to whatever location you want. To change the label on the button or the button size, double click the button to open its properties.



In the above example, several buttons have different names than listed in the list of possible keypad buttons (Vol+ instead of Volume Up). This is because the name has been modified by opening the button properties and entering a new name.

When you have placed all the buttons you want on the keypad, you can test it using the *Keypad Test* button at the bottom of the keypad builder dialog. Before starting the test, select the BC4 and port you want to use.

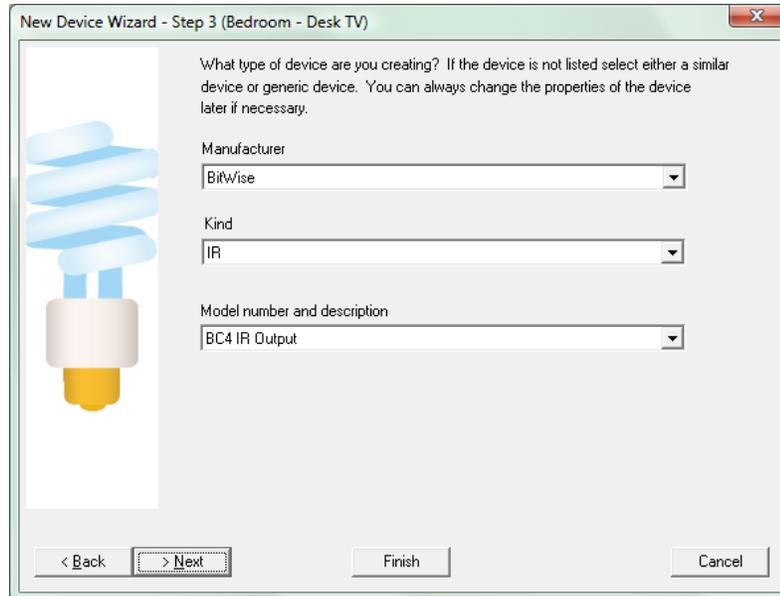


From the test keypad, pressing a button causes the IR sequence to be transmitted by the selected BC4 out the selected port.

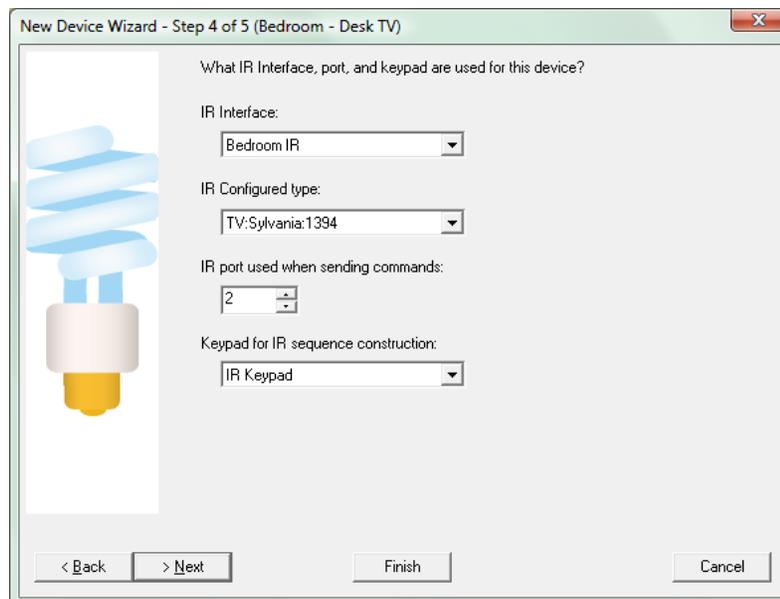
**Hint:** You can create more than one keypad for a device kind. You may want to do that to have a “full” keypad when you work with the device in the HCA UI and a simpler one when used in a client like *HCA for Android*.

## Creating a HCA device for an IR Output Port

Now that you have selected the device kind in the BC4 setup, and have created keypads to use with each device kind, you can finally create the HCA device to represent the IR output port. In step 3 of the New Device Wizard, select as the manufacturer *Bitwise* and type *BC4 IR Output*.



On step 4 the specifics of this port are configured.



You select:

- Which BC4unit and port this device represents. The BC4 is selected by the name you entered when it was identified to HCA in the *IR – Bitwise Controls IR Setup* dialog.
- Which kind of device. This is one of the eight you programmed the BC4 for.
- Which IR keypad to associate with this device. Keypads are selected by the name you used when you created it in the keypad builder.

The remainder of the New Device Wizard is the same as for all other devices.

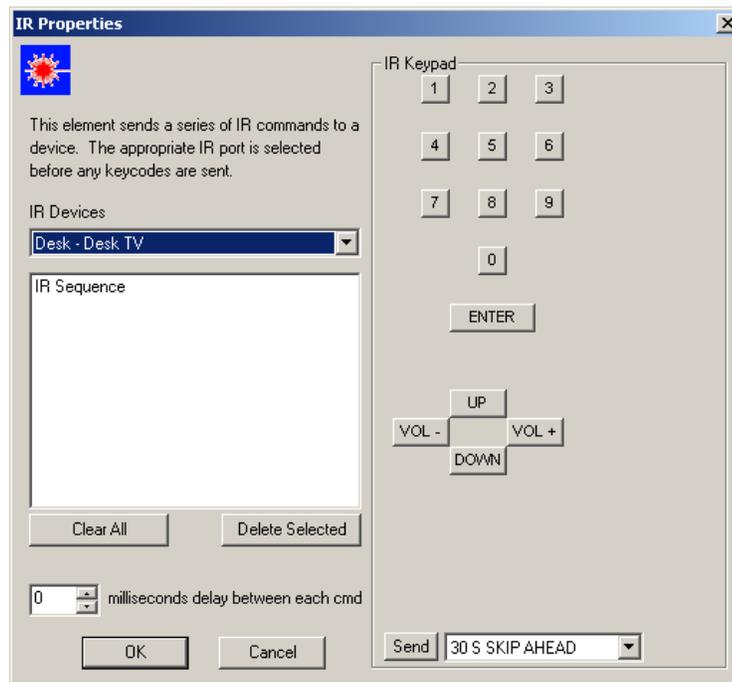
**Hint:** The ports are labeled on the back of the BC4 starting at 1. HCA uses the same numbering.

**Hint:** Rather than name the device with the BC4 name and port number you may want to name the device with what it controls. For example, you may want to use Desk TV, Home DirecTV tuner, etc.

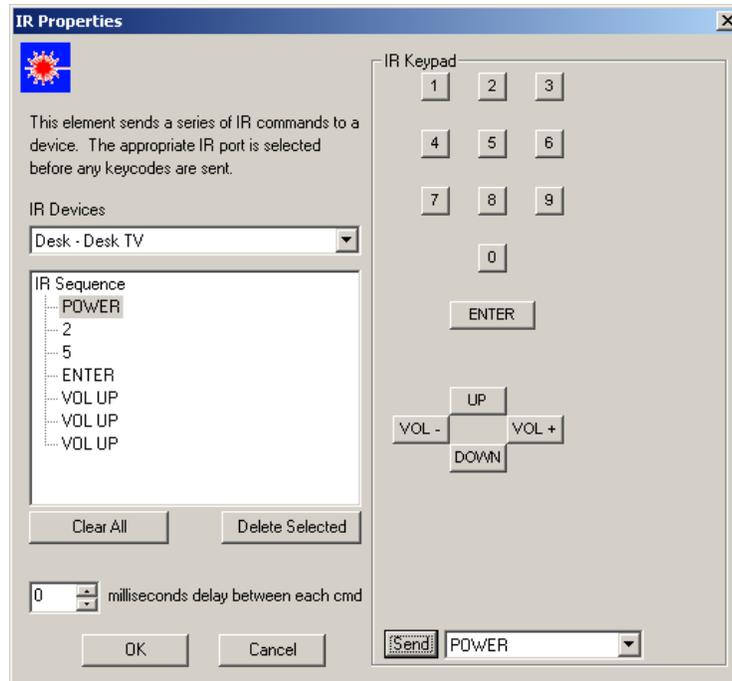
---

## IR sequences in programs

Now that all the setup steps are complete you can finally use the IR port in a program. The IR element is used for this:



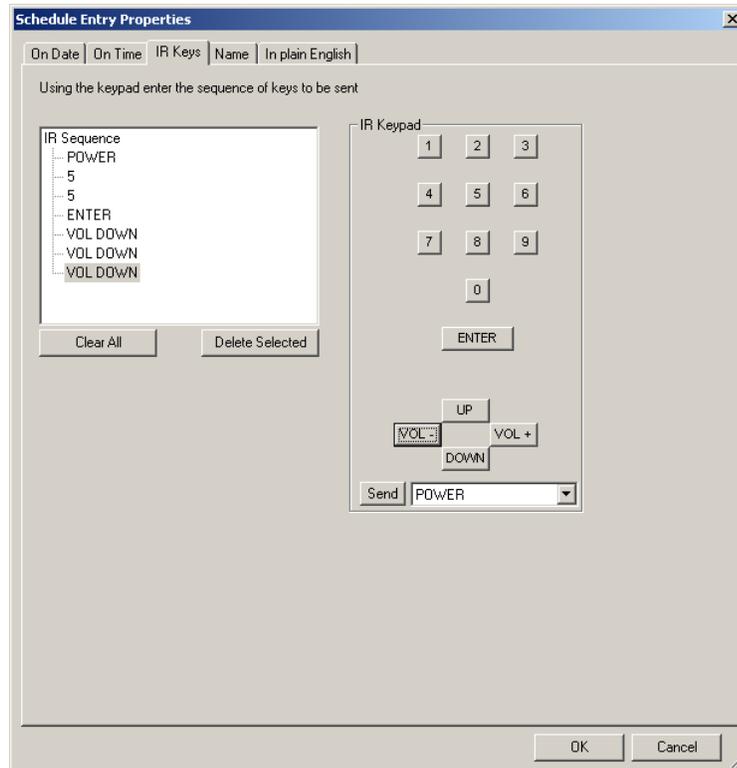
Use the keypad to enter the sequence you want to transmit.



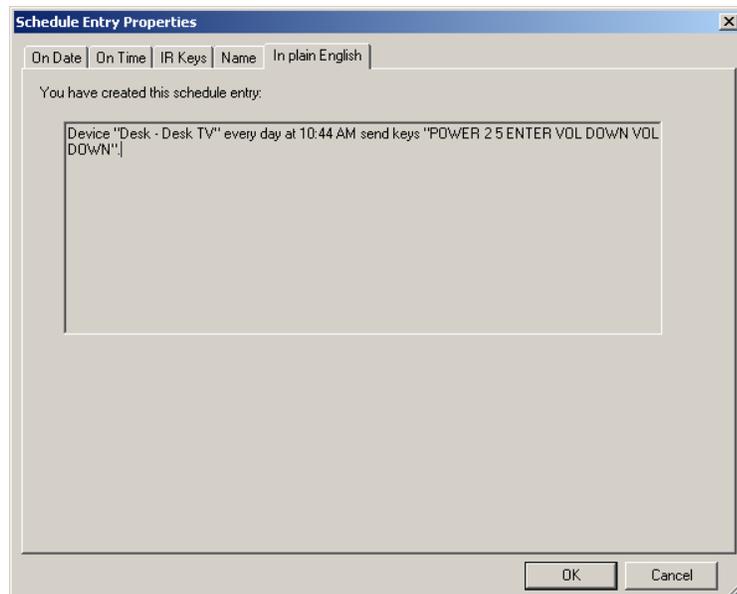
If you want to delete some keys in the sequence, select and use the *Delete Selected* Button. New entries are always inserted after the selected item so it is easy to enter missed keys in the middle of a sequence.

## IR sequences in schedules

Creating a schedule entry that sends an IR sequence is very similar to the IR Visual Programmer element. The schedule entry wizard has a step for IR sequences if you select an IR device. Also the Visual Schedules has properties for IR time bar markers.



Enter the sequence you want to transmit in the same manner as the Visual Programmer. The final tab (or wizard step) of the schedule entry shows in text the sequence to be transmitted.





## Appendix 6

### Wireless Interfaces

This appendix describes the W800RF32 and MR26 wireless receiver and covers these topics:

- What are the W800RF32 and the MR26?
- Use and configuration
- MR26
- W800RF32
- Creating wireless devices
- Using wireless triggers

The MR26 has been designated a legacy device and support is normally unavailable. Open the HCA Properties dialog and choose the legacy tab to enable support.

---

#### What are the W800RF32 and the MR26?

Both of these devices are receivers of wireless signals produced by motion sensors and wireless keypads. They receive the signal and send the command directly into the computer using a serial connection.

Until they were available, these signals were received by a wireless transceiver and retransmitted onto the powerline. The powerline transmission was then received by the computer and processed by HCA. Pictorially it looked like this:

Motion sensor → Transceiver → powerline → X10 interface → HCA

With a wireless interface it now looks like this

Motion sensor → Wireless Interface → HCA

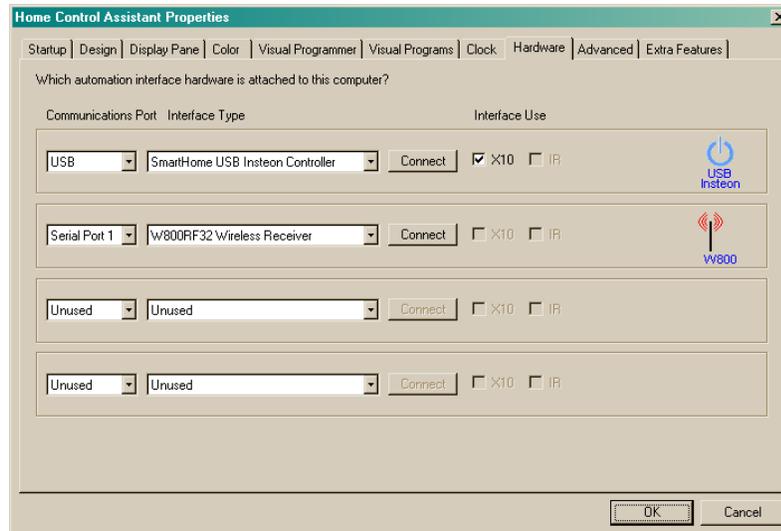
The advantage of this is that the wireless signal gets into the computer much quicker. If you have programs that start when you walk in front of a motion sensor, for example, the program starts much faster than if you used a Transceiver.

There is one major potential problem with using a wireless interface that you should be aware of. In the old method where you used a transceiver, the signal from the wireless sensor (for example a motion sensor) made it on to the powerline. This allowed you to set up a lamp, for example, that goes on as soon as the motion sensor detects motion. If the motion sensor was set to B7 for example, and a lamp module was also set to B7, the motion sensor's transmission would be received by the transceiver, and when put on the powerline the lamp would come on – as well as HCA receiving the signal.

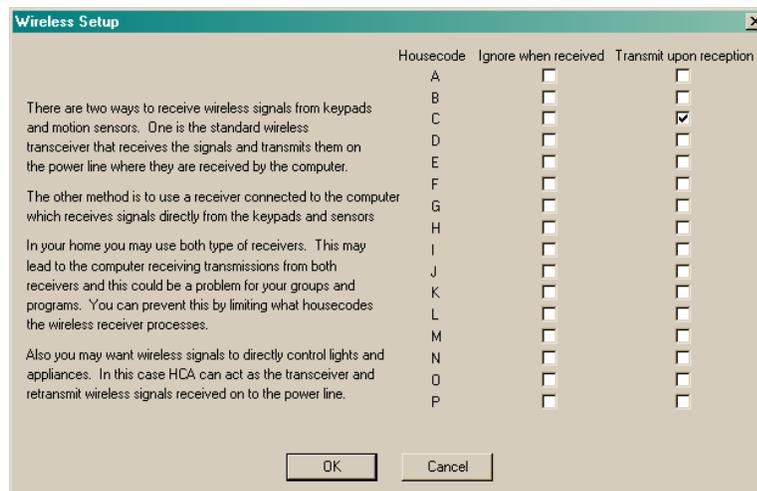
By eliminating the transceiver the signal gets to HCA quicker, but not onto the powerline. HCA has a way to fix this and is explained below.

## Use and Configuration

To use one of these interfaces attach it to an unused serial port on your computer. Then open the HCA properties dialog and select the hardware tab.



Once this is done, the next step is to configure how it works with HCA. Press the *Wireless* button in the ribbon *Interfaces* category and select *Configure* from the menu.



As this text in this dialog explains, you can have HCA both ignore transmissions from some house codes and also act as a transceiver for others.

While you may think that a Wireless Interface can replace all the X10 transceivers in your home – and it can – there may be a few problems yet to solve.

What problems? For example you have a wireless remote for housecode B and a transceiver set for housecode B and a lamp on B3. You send a B3 and the lamp comes on. This happens because the transceiver picks up the wireless broadcast and sends the B3 out on the power line. No computer involvement.

Now without the transceiver and using a wireless interface the B3 goes right into the computer bypassing the powerline so the light doesn't come on. To recover the function you want you could write a program that is triggered on the wireless reception, but there is a better way. Another option in this Setup dialog is to broadcast onto the powerline signals it receives, thus replacing the transceiver function. You can make this happen for selected housecodes. If you checked this option for housecode B, what would happen is this:

Wireless keypad → Wireless Interface → HCA → X10 interface → lamp

Thus HCA becomes the transceiver.

## MR26

While all this sounds great there are some technical problems. The MR26 comes with an antenna that is insufficient, and you will be disappointed in its reception range. If you feel capable of working on electronic projects there are ways to improve the MR26.

See the HCA technical notes directory on the support web site for more information on the MR26.

## W800RF32

The W800RF32 is a wireless interface from WGL and Associates. Like the MR26 it can receive all transmissions from X10 motion sensors (those that you set an X10 address using little push buttons) and wireless keypads.

More information about the W800RF32 is available at

<http://www.wgldesigns.com/>

The best news is that this device works very well and has an excellent range. This makes it much more usable than the MR26.

In addition to the X10 address motion sensors and keypads the W800RF32 can receive, it also can receive transmissions from a second series of keypads and sensors.

The X10 corporation makes a wireless security system that uses motion sensors, door and window closure contacts, and keypads. All of these items transmit wireless signals that can be received by the W800RF32 (but not the MR26).

Even though these are sold as "security" devices don't be convinced that they need be used for security. For example, you can use one of the door/window contacts to send a signal that turns lights on when you open a closet door, for example.

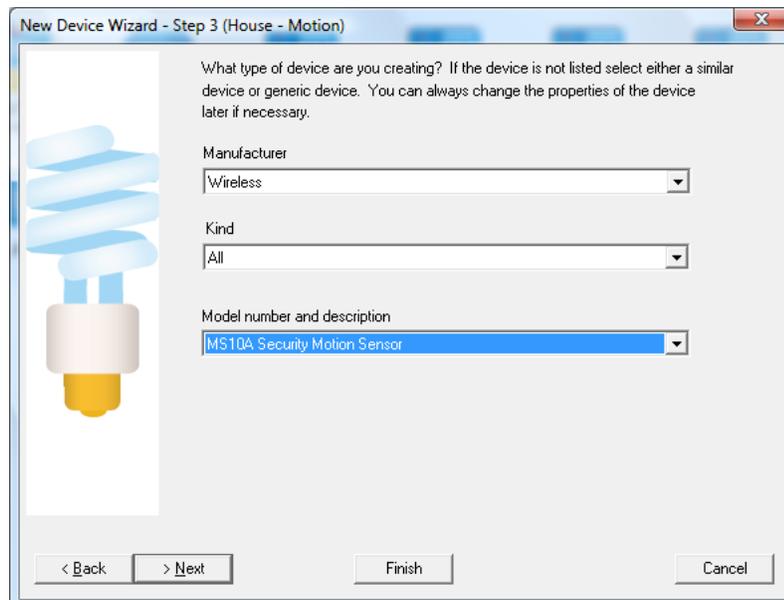
---

## Wireless Devices

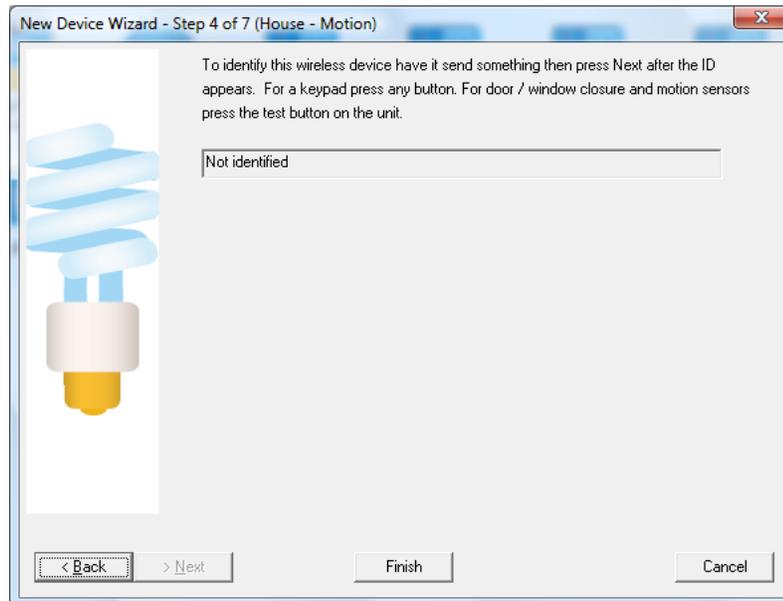
Before discussing the Wireless Component Inventory, don't get the concept of "security" motion sensors and the "security" keypads confused with other wireless keypads and motion sensors - those that transmit house and unit codes. If the motion sensor and keypad has a method for setting its X10 address, then its transmissions can already be received by HCA from either the MR26, W800RF32, or by plug in wireless transceivers. These kinds of motion sensors and keypads are added to your design by using the New Device wizard.

**Hint:** Before you start adding wireless components, you must have the W800RF32 hooked up and working.

To add wireless components that don't send a house and unit code, add them to your design as any other device. At step 3 of the wizard select as the manufacturer "Wireless" and choose the device type.



The next step of the wizard asks you to identify the device so that HCA can capture its assigned address.



The way these types of wireless devices work is that they transmit an identifying unique code as well as a command. The code that they transmit must be different than any other wireless sensor so there is a method to cause it to choose a new code.

Hint: Read the documentation that comes with the wireless security sensor or with the whole X10 wireless security system on how to select the codes.

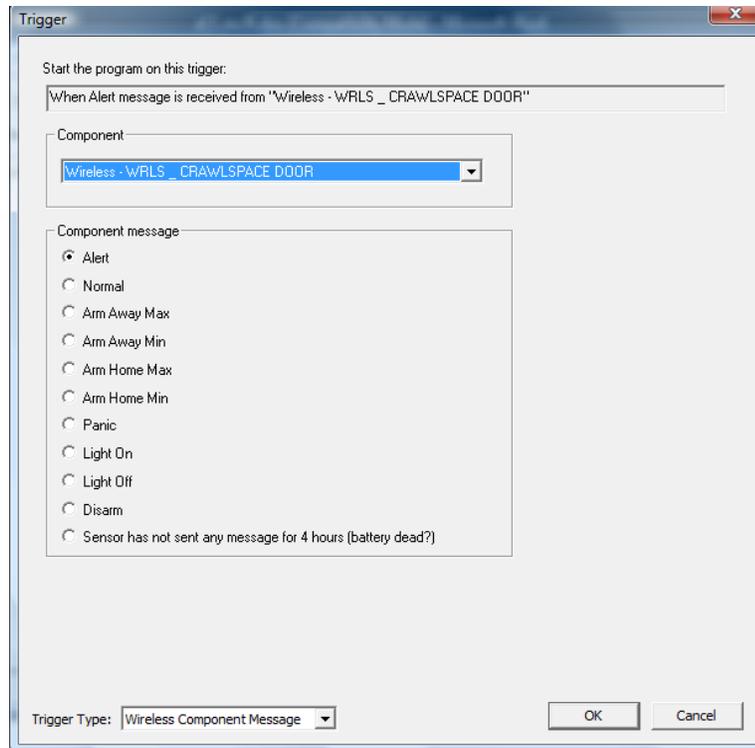
Once this step is on the screen, do something that causes the device to transmit. If it is a door/window contact closure, open and close the door. For a keypad, press any button on the keypad. For a motion sensor, all you need do is to walk in front of it.

When HCA receives the transmission through the W800RF32, it displays it in the lower box. Press Finish when done.

---

## Using Wireless Triggers

Once you add wireless components using the Inventory dialog described above you can now make use of their transmissions to trigger programs. To do this open the program's properties and select the Triggers tab. Select Wireless Component Message as the trigger type and press add.



To create a trigger, select the component and the type of message. The types of messages depend upon the type of sensor. These are:

#### **Door / Window Sensor**

When open sends an Alert message. When closed sends a Normal message

#### **Motion Sensor**

When motion detected sends an Alert message. After motion stops and a time delay passes, sends a Normal message.

#### **Keypads**

All the other commands are sent by the Security keypads.

The last choice in the dialog takes advantage of a nice feature of the motion sensors and the contact sensors. About every 90 minutes they send a message to "check-in". If they don't send a message within 4 hours the sensor should be checked as it might be broken or its battery dead.

Instead of selecting a component in the dialog you can also select the "Any" choice. This gives you the ability to create a program that responds to a message from any sensor.

**Hint:** By choosing *Any* for the component and the *Sensor has not sent a message in 4 hours* message choice you can create a program that alerts you when any sensor should be looked at.

**Hint:** Don't forget that these kinds of messages can only be received if you are using the W800RF32 interface.

## Appendix 7

### HCA Objects

This appendix describes the HCA Object model which can enable you to write scripts and Windows Programs that interact with HCA. These topics are covered:

- Introduction
- Example
- HCA Object model

This is a feature provided in HCA primarily for advanced users. HCA technical support will be unable to help you with your programs other than to provide any additional information on the HCA objects and methods documented here. Getting your programs and scripts to work is your responsibility!

---

#### Introduction

Object interfaces have been provided in HCA. Using these you can write Windows programs, using Visual Basic, Visual C, Java, and other languages, that access devices, programs, groups, schedules, and flags of a HCA design. Programs you write can also ask HCA to control devices, start programs, change schedules, etc.

One very important note: You can't use these interfaces unless HCA is already running on your computer. If you try to use them without HCA running, Windows will try and start HCA and that will not work.

There are two good ways to get started using these features.

- Use an object browser to examine the HCA type library. The type library is in the HCA program folder and is called HCA.TLB
- An example using these interfaces was written in Excel VBA. Excel was chosen since so many people have that program.

It is assumed that you are familiar with Visual Basic. There are many good books that you can purchase that will help you understand how to write programs. Excel itself comes with a Visual Basic help file and that might be a good place to start.

To work with other programming languages, look for information on how to call "Automation interfaces".

---

#### Example

Microsoft Excel contains as its macro language VBA or Visual Basic for Applications. An example of using the HCA Objects was created using Excel and VBA.

To work with this example, start Excel and load the file HCA.XLS. This file is in the samples folder of the HCA installation.

**Note:** If you have macros disabled in Excel for protection against viruses you must first start Excel then load HCA.XLS in order for Excel to display the dialog that offers you the option of enabling macros (you want to say Yes when it asks that). A double click on HCA.XLS in the Windows Explorer to start Excel and load the file at the same time may skip this dialog and you don't get the option of enabling macros.

Once you have this file loaded, follow the directions to run the macro and how to view the VB code.

**Hint:** Make sure that you get the Excel sample to run before you start your own programming task. That way you can be sure that all the necessary bits and pieces of Windows (DLLs) are available. Also review the code in the example to see how it works. If you write in to technical support about the VBA interface they will start out by asking if you have got the example to run and if you have looked at the code in it.

---

## HCA Object Model

The following documents the objects and methods provided by HCA. Listed below each object are the methods supported. The method name, its return and argument types, as well as a brief description are given.

---

### Error Handling

Methods that succeed return a non-negative result. Methods that fail return negative numbers in all cases. For example, if you pass the name of a device to the Device.On method and it is not the name of a device in your design, -1 is returned. -1 is always used for an invalid parameter. The other error codes are:

Return code	Use for
-1	Invalid Parameter
-2	Action not applicable. For example, using an IR method on a non IR device.
-3	Need Remote Access password
-31	Need Control Access password
-32	Need Design Access password
-4	Device, Program, Group or Controller suspended from remote control or disabled.
-5	Hardware needed to complete operation not available
-6	Did not work. Only for operations that confirm success

**You must check all methods for an error return!** If a method returns a VARIANT – used to return strings – you still need to test for an error by checking the type of the variant. If you are expecting a string and get a variant of I2 then it is an error and your program must handle that. Also for methods that return numbers – like a Count method – you should check for a negative number being returned and that will indicate an error.

---

## Controller and Device Objects

The HCA object model still shows a split between devices and controllers – a distinction made in the HCA UI prior to version 9. You will just have to determine what sort of object is a device and what sort of device is a controller and operate upon it using the correct object. In general, things that are like keypads and controllers and things like modules and switches are devices.

---

## Obsolete Methods

By examining the HCA Type Library you may see methods not listed here. These are considered obsolete or internal and should not be used.

## HCA Object

The HCA object contains methods that control HCA and get access to the overall elements of the current design.

Name	<b>GetHomeModeCount</b>
Description	Get count of home modes
Parameters	None
Return Value	Short: Count

Name	<b>GetHomeModeName</b>
Description	Gets the ith mode name
Parameters	Short i
Return value	VARIANT: Mode name

Name	<b>GetCurrentHomeMode</b>
Description	Returns the current mode
Parameters	None
Return Value	VARIANT: Mode name

Name	<b>SetCurrentHomeMode</b>
Description	Changes the home mode
Parameters	BSTR: New mode name
Return Value	Short: Result code

Name	<b>SetPassword</b>
Description	Provide a password used by any method that, because the design is password protected, needs a password.
Parameter 1	Short: Type
	0 = Modify access
	1 = Control access
	4 = Remote access
Parameter 2	BSTR: Password
Return Value	Short: Result code

Name	<b>RemovePassword</b>
Description	Clears a password previously stored by SetPassword
Parameters	Short: Type – same as SetPassword
Return Value	Short: Result code

Name	<b>X10Send</b>
Description	Sends an X10 address and command.
Parameters	Short: HC. (0 = HC_A 15 = HC_P)
	Short: UC. (0 = UC_1 15 = UC_16)
	Short: Cmd
	0 = All units off
	1 = All lights on
	2 = On
	3 = Off
	4 = Dim
	5 = Bright
	6 = All lights off
	7 = Hail request
	8 = Hail reply
	9 = not used
	10 = Status On
	11 = Status Off
	12 = Status Reply
Return Value	Short: Result code

Name	<b>X10SendAddress</b>
Description	Sends just an X10 address. The House Code and Unit Code are encoded in the same manner as the X10Send method.
Parameters	Short: HC
	Short: UC
Return Value	Short: Result code

Name	<b>X10SendCommand</b>
Description	Sends just an X10 command. The House Code and command is encoded in the same manner as the X10Send method.
Parameters	Short: HC
	Short: Command
Return Value	Short: Result code

Name	<b>X10SendPresetDim</b>
Description	Sends an X10 preset dim command. The House Code and Unit Code are encoded in the same manner as the X10Send method. The level is a number between 0 and 31, inclusive
Parameters	Short: HC
	Short: UC
	Short: Level
Return Value	Short: Result code

Name	<b>HCAVersionMajor</b>
Description	Returns the major version of HCA. For example, 9
Parameters	None
Return Value	Short: Major version

Name	<b>HCAVersionMinor</b>
Description	Returns the minor version of HCA. For example, 0
Parameters	None
Return Value	Short: Minor version

Name	<b>HCABuildNumber</b>
Description	Returns the build number of HCA. For example, 10
Parameters	None
Return Value	Short: Build number

Name	<b>HCAFlavor</b>
Description	Returns Flavor of HCA. 0 is HCA Limited, 1 is HCA Standard, 2 is HCA Plus, and 3 is HCA Pro.
Parameters	None
Return Value	Short: Flavor

Name	<b>ModificationTime</b>
Description	Returns the date-time when the currently loaded HCA design was last modified. This value is the number of seconds since midnight January 1, 1970 UCT.
Parameters	None
Return Value	Long: Time

Name	<b>ModificationTimeV</b>
Description	Same as ModificationTime but returns it as a VT_DATE variant
Parameters	None
Return Value	VARIANT: Time

Name	<b>FolderCount</b>
Description	Returns the number of folders in the current design
Parameters	None
Return Value	Short: Count

Name	<b>FolderName</b>
Description	Returns the ith folder name. The variant type is BSTR
Parameters	Short: i
Return Value	VARIANT: Name

Name	<b>FolderContentsCount</b>
Description	Returns the number of objects stored in the folder.
Parameters	BSTR: Folder name
Return Value	Short: Count

Name	<b>FolderContentsName</b>
Description	Returns the name of the ith object stored in the specified folder. The variant type is BSTR.
Parameters	BSTR: Folder name
	Short: i
Return Value	VARIANT: Name

Name	<b>ObjectType</b>
Description	Returns the type of the object
Parameters	BSTR: Name
Return Value	Short: Type
	1 = Device
	2 = Group
	3 = Controller
	4 = Program
	5 = Program Flag
	6 = Schedule
	7 = Display
	8 = Icon
	9 = Schedule Entry (Note: not all schedule entries have a name)
	10 = iButton
	11 = Magic Module
	12 = Wireless sensor
	13 = IR Keypad

Name	<b>SunriseTime</b>
Description	Returns the time for sunrise on the current day. The variant type is VT_DATE.
Parameters	None
Return value	VARIANT: Time

Name	<b>SunsetTime</b>
Description	Returns the time for sunset on the current day. The variant type is VT_DATE.
Parameters	None
Return value	VARIANT: Time

Name	<b>ProblemLevel</b>
Description	Returns the current “traffic light” color
Parameters	None
Return value	Short: level
	0 = Green
	1 = Yellow
	2 = Red

Name	<b>GetDesignPaneOrganization</b>
Description	Returns the current user selection for the HCA Window design pane
Parameters	None
Return value	Short: Organization
	0 = By folder
	1 = By type

Name	<b>GetDesignTitle</b>
Description	Returns the text of the home title set by Home – Properties. Variant type is VT_BSTR
Parameters	None
Return value	VARIANT: Title

Name	<b>RemoteAccessPasswordRequired</b>
Description	Returns TRUE if the current design has a remote access password set
Parameters	None
Return value	Short: Result

Name	<b>FormatString</b>
Description	Takes a string with replacement sections – like the ShowMessage visual programmer element – and evaluates the embedded expressions to create a final string which is returned.
Parameters	BSTR: String with replacement sections
Return value	VARIANT: Result

Name	<b>InstallRemoteAccessCommWindow</b>
Description	Provides to HCA a Windows handle where messages can be posted by HCA to communicate events. The windows message is always WM_USER + 10.
Parameters	Unsigned long: hWnd
Return value	Short: Result code

This table shows the wParam and lParam values the HCA sends to the comm. window.

wParam	lParam	Use
1	0	HCA is terminating

Name	<b>RemoveRemoteAccessCommWindow</b>
Description	Removes a previously installed comm. window handle
Parameters	None
Return value	Short: Result code

Name	<b>GetStartDisplay</b>
Description	Returns the name of the display or folder the user selected in HCA – Properties, Web tab. The return type is VT_BSTR. It may be "" if none was selected.
Parameters	None
Return value	VARIANT: Display name

Name	<b>GetRemoteAccessParameters</b>
Description	Returns the remote access parameters string selected in HCA – Properties, Web tab. The return type is VT_BSTR. It may be "" if none was selected.
Parameters	None
Return value	VARIANT: Display name

---

## Device, Group, Controller common methods

Some methods are the same in HCA.Device, HCA.Group, HCA.Controller and HCA.Program. These are documented here.

In the “Applies to” row the method applies to devices if a “D” is shown, applies to programs if a “P” is shown, applies to groups if a “G” is shown, and applies to controllers if a “C” is shown.

Name	<b>Count</b>
Applies to	DPGC
Description	Returns the count of objects, for example, devices
Parameters	None
Return Value	Short: Count

Name	<b>Name</b>
Applies to	DPGC
Description	Returns the name of the ith object. The variant has type VT_BSTR. Devices are numbered 0 .. count – 1
Parameters	Short: i
Return Value	VARIANT: Name

Name	<b>GetTreeIcon</b>
Applies to	DPGC
Description	Returns the icon used in the design pane for this object. The number returned is a value 0 to 'n'. On the object properties name tab, all the design pane icons appear. The number returned from GetTreeIcon is in the order they appear on that tab – top to bottom and left to right.
Parameters	BSTR: Object name
Return Value	Short: Tree icon number

Name	<b>IconFilename</b>
Applies to	DPGC
Description	Returns a path to the bitmap file for the icon used to represent the device, group, or controller in the HCA display pane. Works for user added icons and the HCA stock icons.
Parameters	BSTR: Object name
	Short: State (0 = off, 1 = on, 2 = dim)
	BSTR: Display name
Return Value	VARIANT: Path

Name	<b>IconXPos</b>
Applies to	DPGC
Description	Returns the X position of an icon for this object on this display
Parameters	BSTR: Object name
	BSTR: Display name
Return Value	Short: X Pos

Name	<b>IconYPos</b>
Applies to	DPGC
Description	Returns the Y position of an icon for this object on this display
Parameters	BSTR: Object name
	BSTR: Display name
Return Value	Short: X Pos

Name	<b>Suspend</b>
Applies to	DPGC
Description	Performs a Suspend All on this object
Parameters	BSTR: Object name
Return Value	Short: Result code

Name	<b>Resume</b>
Applies to	DPGC
Description	Performs a Resume All on this object
Parameters	BSTR: Object name
Return Value	Short: Result code

Name	<b>GetSuspend</b>
Applies to	DPGC
Description	Returns a bitmap that tells if the object is suspended
	0x01 = suspend schedule
	0x02 = suspend program
	0x04 = suspend remote
Parameters	BSTR: Object name
Return Value	Short: Bitmap

Name	<b>GetModeSuspend</b>
Applies to	DC
Description	Returns TRUE if the device or controller is suspended due to the current home mode
Parameters	BSTR: Device or controller name
Return Value	Short: Result

Name	<b>SupportsDim</b>
Applies to	DGC
Description	Returns 1 if this object supports dimming and 0 otherwise.
Parameters	BSTR: Object name
Return Value	Short: Result

Name	<b>SupportsStatus</b>
Applies to	DGC
Description	Returns 1 if this object supports status retrieval and 0 otherwise.
Parameters	BSTR: Object name
Return Value	Short: Result

Name	<b>On</b>
Applies to	DGC
Description	Sends an ON command to this object
Parameters	BSTR: Object name
Return Value	Short: Result code

Name	<b>Off</b>
Applies to	DGC
Description	Sends an OFF command to this object
Parameters	BSTR: Object name
Return Value	Short: Result code

Name	<b>DimToPercent</b>
Applies to	DGC
Description	Sets the object to this illumination level by percent
Parameters	BSTR: Object name
	Short: Percent
Return Value	Short: Result code

Name	<b>DimUpPercent</b>
Applies to	DGC
Description	Increase illumination level by percent
Parameters	BSTR: Object name
	Short: Percent
Return Value	Short: Result code

Name	<b>DimDownPercent</b>
Applies to	DGC
Description	Decrease illumination level by percent
Parameters	BSTR: Object name
	Short: Percent
Return Value	Short: Result code

Name	<b>State</b>
Applies to	DGC
Description	Return state of object
Parameters	BSTR: Object name
Return Value	Short: state
	0 = Off
	1 = On
	2 = Dim

Name	<b>DimPercent</b>
Applies to	DGC
Description	Returns the illumination level of the object as a percentage
Parameters	BSTR: Object name
Return Value	Short: Percent

Name	<b>RequestStatus</b>
Applies to	DC
Description	Requests status from a device that can respond to status requests. Returns the percent level the device is at like the DimPercent method. Can return -6 if the device doesn't respond to the status requests. Also updates the internal HCA device state.
Parameters	BSTR: Device name
Return Value	Short: Return code

Name	<b>UPBLink</b>
Applies to	DC
Description	Send a UPB Link command
Parameters	BSTR: Link name
	Short: op
	0 = Activate
	1 = Deactivate
	2 = Goto
	3 = Blink
	Short: Level
	Short: Rate
Return Value	Short: Return code

Name	<b>UPBBLink</b>
Applies to	DC
Description	Send a UPB Blink command
Parameters	BSTR: Device name
	Short: Rate
Return Value	Short: Return code

Name	<b>Scene</b>
Applies to	DC
Description	Control a scene
Parameters	BSTR: Scene name
	BSTR: Device name
	Short: op
	0 = Deactivate
	1 = Activate
	Short: Insteon confirm. If 1, send confirmation commands to each scene member
Return Value	Short: Return code

Name	<b>OnEx</b>
Applies to	DC
Description	Send an ON command to a device
Parameters	BSTR: Device name
	Short: Indicator
	Short: Fade rate
Return Value	Short: Return code

Name	<b>OffEx</b>
Applies to	DC
Description	Send an OFF command to a device
Parameters	BSTR: Device name
	Short: Indicator
	Short: Fade rate
Return Value	Short: Return code

Name	<b>DimEx</b>
Applies to	DC
Description	Send a DIM command to a device
Parameters	BSTR: Device name
	Short: Op
	0 = Set to percent
	1 = Decrease by percent
	2 = Increase by percent
	Short: Percent
	Short: Fade rate
Return Value	Short: Return code

The next methods are provided to allow applications to construct virtual keypads. This is best explained for controllers – which can easily be thought of as keypads – but applies to devices as well since there are some devices that are hybrids of devices and controllers. For example, some UPB devices have removable faceplates that allow a single rocker switch to be changed into a two rockers, or into two rockers and 4 buttons, or into 8 buttons. One rocker or button still controls the attached load and the other rockers and buttons act as transmitters.

A virtual keypad allows an application to remotely *tap* or *press* a button on a device or keypad. For a simple one-load controlling switch, clicking the virtual rocker or directly controlling the load will be the same. But if, for example, the device is an Insteon switch, then tapping the rocker may control many devices because of stored scenes.

NOTE: These methods apply to controllers and devices but not groups.

Name	<b>RockerCount</b>
Applies to	DC
Description	Number of rockers on the device. May be zero
Parameters	BSTR: Name
Return Value	Short: Count

Name	<b>ButtonCount</b>
Applies to	DC
Description	Number of buttons on the device. May be zero
Parameters	BSTR: Name
Return Value	Short: Count

When using the next three methods: `Keypress`, `KeyName`, and `KeyLabel` it is necessary to carefully determine that value for `iKey`. If I device has only buttons or only rockers it is simply the component (rocker or button) number. The 1<sup>st</sup> component is designated zero.

If a device has rockers and buttons then `iKey` still numbers the component. An example: A device has two rockers and 3 buttons then:

- `Keypress 0` (first rocker)
- `Keypress 1` (second rocker)
- `Keypress 2` (first button)
- `Keypress 3` (second button)
- `Keypress 4` (third button)

Name	<b>Keypress</b>
Applies to	DC
Description	Simulate a press of a rocker or button on a device. Controls the <code>ith</code> button or rocker.
Parameters	BSTR: name
	Short: <code>iKey</code>
	Short: half
	0 = rocker top
	1 = rocker bottom
	Short: action
	0 = single-click
	1 = double-click
	2 = hold
	3 = release
Return Value	Short: return code

Name	<b>KeyName</b>
Applies to	DC
Description	Returns the name of the <code>ith</code> button or rocker for this device. The name will be something like “A10” or “R1” – for rocker one – or “B1” for button one.
Parameters	BSTR: Name
	Short: <code>iKey</code>
Return Value	VARIANT: Name

Name	<b>KeyLabel</b>
Applies to	DC
Description	Returns the label of the <code>ith</code> button or rocker for this

	device. The key label for rockers will always be “On” for the top and “Off” for the bottom. For buttons, the label will be “On” if the button is on and “Off” if the button is off.
Parameters	BSTR: Name
	Short: iKey
	Short: Half (Must be 0 for buttons)
	0 = rocker top
	1 = rocker bottom
Return Value	VARIANT: name

The next two methods are different than Keypress described above in how the component is represented.. Keypress numbers the rockers and buttons sequentially.

RockerPress and ButtonPress number the component only within those component types. For example a device with two rockers and 3 buttons would be controlled as:

- RockerPress 0 (first rocker)
- RockerPress 1 (second rocker)
- ButtonPress 0 (first button)
- ButtonPress 1 (second button)
- ButtonPress 2 (third button)

Name	<b>RockerPress</b>
Applies to	DC
Description	Simulate a press of a rocker on a device. Controls the ith rocker.
Parameters	BSTR: name
	Short: iRocker
	Short: half
	0 = rocker top
	1 = rocker bottom
	Short: action
	0 = single-click
	1 = double-click
	2 = hold
	3 = release
Return Value	Short: return code

Name	<b>ButtonPress</b>
Applies to	DC
Description	Simulate a press of a button on a device. Controls the ith button.
Parameters	BSTR: name
	Short: iButton
	Short: action
	0 = single-click
	1 = double-click
	2 = hold
	3 = release
Return Value	Short: return code

---

## HCA.Device Object

In addition to the common methods, the device object contains these methods that work with HCA devices.

Name	<b>Kind</b>
Description	Returns the kind of device
Parameters	BSTR: Device name
Return Value	Short: kind
	0 = Other
	1 = X10 device
	2 = IR Device
	3 = Thermostat
	4 = Relay
	5 = UPB Device
	6 = Insteon device

Name	<b>Thermostat</b>
Description	Controls or receives thermostat information. The Name parameter supplies the name of the thermostat device.
Parameters	BSTR: Device name
	Short: Op
	Short: Data
Return Value	Short: Return code

Op	Data	Return value
0: Change setpoint	new setpoint	
1: Change mode	new mode	
2: Change fan	new fan	
3: Change economy	new economy	
4: Get current temperature	ignored	current temp

5: Get mode	ignored	current mode
6: Get fan	ignored	current fan
7: Get economy	ignored	current economy
8: Get heat setpoint	Ignored	Current heat setpoint
9: Get cool setpoint	Ignored	Current cool setpoint
10: Set cool setpoint	Setpoint	
11: Get current humidity	ignored	Humidity %
12: Capabilities for Set	Ignored	(See a)
13: Capabilities for Get	Ignored	(See b)
14: Heat setpoint range low	Ignored	temperature
15: Heat setpoint range high	Ignored	temperature
16: Cool setpoint range low	ignored	temperature
17: Cool setpoint range high	ignored	temperature

- The encoding for mode argument is 0 = off, 1 = heat, 2 = cool, 3 = auto.
- The encoding for the fan and economy argument is 0 = off, 1 = on.
- The setpoint should be provided in whatever units (F or C) that the thermostat is setup for.
- Not all thermostats support the same features. Operation #12 returns a bitmap of what Set operations are available for this thermostat.

Value	Use
0x01	Can change mode
0x02	Can change fan
0x04	Can change economy
0x08	Can change cool setpoint
0x10	Can change heat setpoint

- Not all thermostats support the same features. Operation #13 returns a bitmap of what Get operations are available for this thermostat.

Value	Use
0x01	Can get mode
0x02	Can get fan
0x04	Can get economy
0x08	Can get cool setpoint
0x10	Can get heat setpoint
0x20	Can get current temperature
0x40	Can get humidity

Name	<b>IR</b>
Description	Sends an IR command. The keyname is the name on the button on the popup keypad associated with the device. The final argument is a true/false value. For some IR interfaces this is needed and for others it is ignored.
Parameters	BSTR: Device name
	BSTR: Key name
	Short: First keypress of a sequence (0 = no, 1 = yes)
Return Value	Short: return code

The next set of methods is for IR Keypads. In HCA, each IR device has an associated keypad for direct control via the user interface, or for entering IR sequences in program elements and schedules. In HCA a keypad is constructed by a user placing buttons on a canvas, sizing them and positioning them as wanted. Each button has a name and a label. The name tells HCA what IR function to use and the label is just the text seen on the keypad. When using the IR method described above, the name, not the label, is what is passed as the 2<sup>nd</sup> parameter.

Name	<b>IRKeypadButtonCount</b>
Description	Returns the number of buttons on the keypad associated with the named IR device
Parameters	BSTR: Device name
Return Value	Short: Count

Name	<b>IRKeypadButtonName</b>
Description	Returns the name of the ith button
Parameters	BSTR: Device name
	Short: i
Return Value	VARIANT: Name

Name	<b>IRKeypadButtonLabel</b>
Description	Returns the label of the ith button
Parameters	BSTR: Device name
	Short: i
Return Value	VARIANT: Label

Name	<b>IRKeypadButtonType</b>
Description	Returns the type of the ith button
Parameters	BSTR: Device name
	Short: i
Return Value	Short: Type (0 = button, 1 = text)

Name	<b>IRKeypadButtonWidth</b>
Description	Returns the width of the ith button in pixels
Parameters	BSTR: Device name
	Short: i
Return Value	Short: Width

Name	<b>IRKeypadButtonHeight</b>
Description	Returns the height of the ith button in pixels
Parameters	BSTR: Device name
	Short: i
Return Value	Short: Height

Name	<b>IRKeypadButtonX</b>
Description	Returns the X position of the ith button in pixels from the upper left corner of the keypad.
Parameters	BSTR: Device name
	Short: i
Return Value	Short: Position

Name	<b>IRKeypadButtonY</b>
Description	Returns the Y position of the ith button in pixels from the upper left corner of the keypad.
Parameters	BSTR: Device name
	Short: i
Return Value	Short: Position

---

## HCA.Program Object

The program object contains these methods that work with HCA programs.

Name	<b>Start</b>
Description	Starts the program running
Parameters	BSTR: Program name
Return Value	Short: Result code

Name	<b>Stop</b>
Description	If running, terminates the program
Parameters	BSTR: Program name
Return Value	Short: Result code

Name	<b>On</b>
Description	Starts the program with an ON command
Parameters	BSTR: Program name
Return Value	Short: Result code

Name	<b>Off</b>
Description	Starts the program with an OFF command
Parameters	BSTR: Program name
Return Value	Short: Result code

Name	<b>IsRunning</b>
Description	Returns 0 if the program is not running, 1 if the program is running
Parameters	BSTR: Program name
Return Value	Short: Result

Name	<b>StartX10</b>
Description	Starts a program and provides the trigger that starts the program. This trigger can be tested within the program using the Test element
Parameters	BSTR: Program name
	BSTR: HCUC. housecode and unitcode of the trigger supplied by a string. For example, "A1", or "G10".
	Short: Command
	0 = All units off
	1 = All lights on
	2 = On
	3 = Off
	4 = Dim
	5 = Bright
	6 = All lights off
	7 = Hail request
	8 = Hail reply
	9 = preset dim – uses level parameter
	10 = Status On
	11 = Status Off
	12 = Status Reply
	Short: Level
Return Value	Short: Result code

Name	<b>StartMethod</b>
Description	When you right-click on a device in HCA, the popup menu contains either <i>Start</i> or <i>On</i> and <i>Off</i> . This method returns 0 if HCA would display <i>Start</i> and 1 if HCA would display <i>On</i> and <i>Off</i> .
Parameters	BSTR: Program name
Return Value	Short: Result

---

## HCA.Group Object

There are no additional methods for groups beyond the common methods shown above.

---

## HCA.Controller Object

In addition to the common methods, the controller object contains these methods that work with HCA controllers.

Name	<b>Kind</b>
Description	Returns the kind of controller
Parameters	BSTR: Controller name
Return Value	Short: Kind
	0 = Other
	1 = Keypad
	2 = Keypad with dimmer
	3 = Motion sensor
	4 = Input sense module

---

## HCA.Schedule Object

The Schedule object contains methods to work with schedules.

Name	<b>SetCurrent</b>
Description	Makes a schedule the current schedule
Parameters	BSTR: Schedule name
Return Value	Short: Result code

Name	<b>GetCurrent</b>
Description	Returns the name of the current schedule. The variant has type VT_BSTR
Parameters	None
Return Value	VARIANT: Schedule name

Name	<b>Count</b>
Description	Returns the count of schedules
Parameters	None
Return Value	Short: Count

Name	<b>Name</b>
Description	Returns the name of the ith schedule. The variant has type VT_BSTR. Schedules are numbered 0 .. count – 1
Parameters	Short: i
Return Value	VARIANT: Name

Name	<b>ParentName</b>
Description	Returns the name of the schedule's parent. The empty string is returned if the schedule has no parent
Parameters	BSTR: Schedule name
Return Value	VARIANT: Parent Name

Name	<b>EntryCount</b>
Description	Returns the number of entries in the named schedule
Parameters	BSTR: Schedule name
Return Value	Short: Count

Name	<b>EntryImage</b>
Description	Returns a text string of the image of the ith schedule entry in the schedule. This is the same text that HCA displays in the design pane
Parameters	BSTR: Schedule name
	Short: i
Return Value	VARIANT: Image

Name	<b>EntryPerformNow</b>
Description	Perform the ith schedule entry. This is the same operation as in HCA when you right-click on a schedule entry in the design pane and select <i>Perform Now</i> from the popup menu.
Parameters	BSTR: Schedule name
	Short: i
Return Value	Short: Result code

---

## HCA.Flag Object

The Flag object contains methods to work with Flags containing values of any type.

Name	<b>Count</b>
Description	Returns the count of flags
Parameters	None
Return Value	Short: Count

Name	<b>Name</b>
Description	Returns the name of the ith flag. The variant has type VT_BSTR.
Parameters	Short: i
Return Value	VARIANT: Name

Name	<b>Get</b>
Description	Returns the value of the flag
Parameters	BSTR: Flag name
Return Value	VARIANT: Value

Data	Variant type
Number	VT_R8
Date	VT_DATE
Yes/No	VT_BOOL
String	VT_BSTR

Note on error return: The Get method can return an error like most methods. You can tell if you get an error return by looking at the type of the variant. In the type is VT\_I4 then the variant contains an error code.

Name	<b>Set</b>
Description	Sets the value of the flag
Parameters	BSTR: Flag name
	VARIANT*: value
Return Value	Short: Result code

Variant type	Creates a flag of this type
VT_I1, VT_I2, VT_I4, VT_UI1, VT_UI2, VT_UI4, VT_INT, VT_UINT, VT_R4, VT_R8	Number
VT_BOOL	Yes / No
VT_DATE	Date
VT_BSTR	String

Name	<b>SetNumber</b>
Description	Sets the value of the flag
Parameters	BSTR: Flag name
	double: value
Return Value	Short: Result code

Name	<b>SetInt</b>
Description	Sets the value of the flag
Parameters	BSTR: Flag name
	int: value
Return Value	Short: Result code

Name	<b>SetShort</b>
Description	Sets the value of the flag
Parameters	BSTR: Flag name
	short: value
Return Value	Short: Result code

Name	<b>SetText</b>
Description	Sets the value of the flag
Parameters	BSTR: Flag name
	BSTR: value
Return Value	Short: Result code

Name	<b>SetDate</b>
Description	Sets the value of the flag
Parameters	BSTR: Flag name
	DATE: value
Return Value	Short: Result code

Name	<b>SetYesNo</b>
Description	Sets the value of the flag
Parameters	BSTR: Flag name
	bool: value
Return Value	Short: Result code

Name	<b>GetType</b>
Description	Returns the data type of the flag
Parameters	BSTR: Flag name
Return Value	Short: Result code or VT_BOOL, VT_I4, VT_R8, VT_DATE, VT_BSTR

Name	<b>GetNumber</b>
Description	Returns the value of the flag
Parameters	BSTR: Flag name
Return Value	double: value

Name	<b>GetInt</b>
Description	Returns the data type of the flag
Parameters	BSTR: Flag name
Return Value	int: value

Name	<b>GetShort</b>
Description	Returns the value of the flag
Parameters	BSTR: Flag name
Return Value	short: value

Name	<b>GetText</b>
Description	Returns the value of the flag
Parameters	BSTR: Flag name
Return Value	VARIANT: value

Name	<b>GetDate</b>
Description	Returns the value of the flag
Parameters	BSTR: Flag name
Return Value	VARIANT: value

Name	<b>GetYesNo</b>
Description	Returns the value of the flag
Parameters	BSTR: Flag name
Return Value	bool: value

NOTE: There is no way in the previous six methods to determine if the value returned is an error or data. It is expected that you would use the GetType method to determine which of these methods to call and any errors would happen at that point. Also note that these methods do no type coercion so you can't invoke GetDate for a flag that doesn't contain a date value. The result in that case will be indeterminate.

Name	<b>Evaluate</b>
Description	Evaluates the expression and returns the result using the same variant type as the Get method
Parameters	BSTR: text
Return Value	Short: Result code

Note on error return: An error could be returned in the normal VT\_I4 way. For example, password problems would be returned this way. But an error could also occur when the expression text is parsed or when the expression is evaluated. In either case an error string is returned (VT\_BSTR) with the first character of that string always a "~". A way to test for an error would be:

```

Dim value
value = hca.flag.evaluate(expressionText)
If (VarType(value) = vbString) Then
  If Left(value, 1) = "~" Then
    MsgBox "Expression evaluation error: " + Mid(value, 2)
  End If
End If

```

Name	<b>EvaluateEx</b>
Description	Evaluates the expression and returns the result as a formatted string
Parameters	BSTR: text
Return Value	VARIANT: See below

EvaluateEx returns a string of the form:

<a> , <b> , <c>

<a> is a number that is zero if there was no expression parsing or evaluation error. <a> is 1 if there was an expression parsing error. <a> is 2 if there was an expression evaluation error.

<b> is the type of the result. It using the same encoding as GetType.

<c> is a text string representing the result of the expression evaluation.

---

## HCA.Log Object

Name	<b>Open</b>
Description	Opens the HCA log for reading
Parameters	None
Return Value	Short: Result code

Name	<b>Close</b>
Description	Close the log previously opened with Open
Parameters	None
Return Value	Short: Result code

Name	<b>Count</b>
Description	Returns the number of entries in the log
Parameters	None
Return Value	Short: Count

Name	<b>Entry</b>
Description	Returns as a comma separated string the ith entry in the log. Entry(0) is newest and Entry(Count() – 1) is the oldest
Parameters	Short: i
Return Value	VARIANT: Log entry csv text

Name	<b>FilterCount</b>
Description	Returns the number of filters available
Parameters	None
Return Value	Short: Count

Name	<b>FilterName</b>
Description	Returns the name of the ith filter
Parameters	Short: i
Return Value	VARIANT: Filter name

Name	<b>SetFilter</b>
Description	Sets the filter that is used the limit the log entries returned by the Entry method.
Parameters	BSTR: Filter name
Return Value	Short: Result code

Name	<b>SetFilterObject</b>
Description	Creates a temporary filter like the “Show Log” operation when selected from the popup menu on a device, program, group, or controller. Returns 0 if the object name is found in the design and -1 if not.
Parameters	BSTR: device, program, group or controller name
Return Value	Short: Result code

Name	<b>InspectorCount</b>
Description	Returns the count of the number of messages displayed by the inspector. This is the same list displayed in HCA when you open the Troubleshooter and look on the Inspector tab.
Parameters	None
Return Value	Short: Count

Name	<b>InspectorMessage</b>
Description	Returns the text of the ith inspector message. The type of the variant is VT_BSTR
Parameters	Short: i
Return Value	VARIANT: Message text

Name	<b>InspectorMessageCheck</b>
Description	Check-off the ith inspector message. This is the same action as you can do when the troubleshooter is open and you click inside the <input type="checkbox"/> box on the Inspector tab.
Parameters	Short: i
	Short: Check
Return Value	Short: Result code

Name	<b>AddLogEntry</b>
Description	Adds an entry to the HCA log
Parameters	BSTR: Text
Return Value	Short: Result code

---

## HCA.Display Object

Name	<b>Count</b>
Description	Returns the count of displays
Parameters	None
Return Value	Short: Count

Name	<b>Name</b>
Description	Returns the name of the ith display. The variant has type VT_BSTR. Displays are numbered 0 .. count – 1
Parameters	Short: i
Return Value	VARIANT: Name

Name	<b>Type</b>
Description	Returns the type of the display
Parameters	BSTR: Display name
Return Value	Short: Type
	0 = Icon display
	1 = HTML display
	2 = Text display
	3 = Icon display with DXF
	4 = Icon display with image background

Name	<b>AddIconDisplay</b>
Description	Adds a new display to the design for showing icons
Parameters	BSTR: Display name
Return Value	Short: Result code

Name	<b>AddTextDisplay</b>
Description	Adds a new display to the design for showing text
Parameters	BSTR: Display name
	BSTR: Text
Return Value	Short: Result code

Name	<b>AddHTMLDisplay</b>
Description	Adds a new display to the design for showing HTML. Like the HCA UI, you supply the name of the new display and paths to the template and result file.
Parameters	BSTR: Display name
	BSTR: HTML Path
	BSTR: Template Path
Return Value	Short: Result code

Name	<b>Delete</b>
Description	Deletes a display by name
Parameters	BSTR: Display name
Return Value	Short: Result code

Name	<b>ChangeText</b>
Description	Changes the text displayed by a text display
Parameters	BSTR: Display name
	BSTR: Text
Return Value	Short: Result code

Name	<b>ChangeHTML</b>
Description	Changes the HTML displayed by a HTML display. The templatePath may be an empty string.
Parameters	BSTR: Display name
	BSTR: HTML path
	BSTR: Template path
Return Value	Short: Result code

Name	<b>IconCount</b>
Description	Returns the number of icons shown on the display
Parameters	BSTR: Display name
Return Value	Short: Count

Name	<b>Icon</b>
Description	Returns the name of the object (device, program, group, or controller) shown by the ith icon. Note that one device (or program or group or controller) can have more than one icon on a display.
Parameters	BSTR: Display name
	Short: i
Return Value	VARIANT: Name

Name	<b>IconAdd</b>
Description	Adds an icon for a device, program, group, or controller to a display. Note that this is only possible

	for Icon displays and not for Text or HTML displays.
Parameters	BSTR: Display name
	BSTR: Object name
Return Value	Short: Result code

Name	<b>GetText</b>
Description	Returns the current text shown for a text display
Parameters	BSTR: Display name
Return Value	VARIANT: Text

Name	<b>GetHTMLTemplatePath</b>
Description	Returns the path to the HTML template for a HTML display
Parameters	BSTR: Display name
Return Value	VARIANT: Path

Name	<b>GetHTMLPath</b>
Description	Returns the path to the HTML “result” file for a HTML display
Parameters	BSTR: Display name
Return Value	VARIANT: Path

Name	<b>GetDXFBackgroundPath</b>
Description	Returns the pathname for a display’s DXF background
Parameters	BSTR: Display name
Return Value	VARIANT: Path

Name	<b>GetImageBackgroundPath</b>
Description	Returns the pathname for a display’s image file background
Parameters	BSTR: Display name
Return Value	VARIANT: Path

Name	<b>GetCurrentDisplay</b>
Description	Returns the name of the current display
Parameters	None
Return Value	VARIANT: Display name

Name	<b>SetCurrentDisplay</b>
Description	Changes the current display to the named display
Parameters	BSTR: New display name
Return Value	VARIANT: Old display name

Name	<b>ChangeIcon</b>
Description	Change the icon and text for an icon on a display
Parameters	BSTR: Object name
Return Value	VARIANT: Old display name
	Short: Change Text
	0 = Don't change text
	1 = Do Change text
	BSTR: Text
	Short: Change Icon
	0 = Don't change icon
	1 = Do change icon
	BSTR: Icon name
	Short: Icon representation
	0 = Off representation
	1 = On representation
	2 = Dim representation

Name	<b>On</b>
Description	Sends an ON command to this room
Parameters	BSTR: Object name
Return Value	Short: Result code

Name	<b>Off</b>
Description	Sends an OFF command to this room
Parameters	BSTR: Object name
Return Value	Short: Result code

Name	<b>State</b>
Description	Return state of room
Parameters	BSTR: Object name
Return Value	Short: state
	0 = Off
	1 = On

Name	<b>IconFilename</b>
Description	Returns a path to the bitmap file for the icon used to represent the display in the HCA display pane. Works for user added icons and the HCA stock icons.
Parameters	BSTR: Object name
	Short: State (0 = off, 1 = on)
	BSTR: Display name
Return Value	VARIANT: Path

---

## Passwords

As described at the beginning of this appendix, errors are returned if the action performed by the method requires a password. Here is some additional information on that.

In a HCA design a user can set passwords that come in three different types. These are:

- Control
- Design
- Remote

*Control* access is for things like turning stuff on and off and starting programs. *Design* access is for making changes to the design. *Remote* access is being able to use the Object methods to do anything - even reading!

So if a user has assigned a *remote* access password and someone tries to access their HCA design via an application, the application will be unable to even access the design without the user entering the correct password. And if the design has a *control* password, the application can view but not change anything unless the *control* password is supplied.

Most of the methods in the HCA Objects check for passwords, but the password is not passed as a parameter to the method. Passwords are passed to HCA using the SetPassword method and remain in effect until HCA terminates, opens a new design, or the password is removed by using the RemovePassword method.

Internally to HCA, every method - except for the few listed below - start with this preamble:

```
if ((rcAccess = ValidateAccess(accessNeeded)) != 0)
    return (rcAccess);
```

What this function is asking is: "Should the access be allowed?". Each function knows what access type it needs – read access, control access, or design access. For a function like ON, control access would be needed. For a function like GetFolderCount it would need readAccess.

Internally, the ValidateAccess function does this:

1. If remote access password is enabled, then does the remote password equal the password previously passed to "SetPassword" for remote access? If not return -3
2. If requesting control access, then does the control password equal the password previously passed to "SetPassword" for control access? If not return -31
3. If requesting design access, then does the design password equal the password previously passed to "SetPassword" for design access? If not return -32
4. Else return 0

The reason for the different error codes is, for example, so that an application could call the On method, get a remote access error, popup the "Enter web access password" dialog, have the user enter it, call SetPassword with it, and then try ON again. This time it could get a -31 error and the application would again ask the user "Enter the control password", call SetPassword and then try the ON one more time and this time it will work.

When you call SetPassword you pass it which password you are setting (design, control, remoteAccess) and the password. It validates the password and returns an error (-3) if they don't match.

The 1st parameter to SetPassword is the type. Here is the encoding:

- 0: Design password
- 1: Control password
- 4: Remote access password

(Yes the encoding is strange - it's for legacy reasons)

An application must do these things:

- When first accessing the HCA design, call "RemoveAccessPasswordRequired" and if it returns 1 then you have to ask the user for the password and pass it to SetPassword (type = 4). If that function returns an error then you have to ask again.
- You **must** check status returns from all functions that can change things and if you get a negative number back then you have to ask for a *control* password (if you got -31) or a *design* password (if you got -32). It's strangely possible for a return of -3 if someone at the home computer changed the password while someone was "remoting" in.

All methods - except for these - check for passwords before they perform their action:

- HCAVersionMajor
- HCAVersionMinor
- HCABuildNumber
- HCAFlavor
- EnableTwoPartNames
- SunriseTime
- SunsetTime
- RemoteAccessPasswordRequired
- SetPassword
- RemovePassword

Lastly, it should be obvious with little thought about what functions require what kind of access. All of them - except the few listed above - require read access, some require control access (on, off, dim, etc) and just a few require design access (the ones in the Display object for creating displays, etc). The only ones that could be a question are the Evaluate method and the Set method in the flags object. Since they can do just about anything, they require design access.

## Appendix 8

# Universal Powerline Bus (UPB)

This appendix describes the features in HCA in support of the UPB technology and products available from Powerline Control Systems (PCS), Simply Automated Inc, and other manufacturers. These topics are covered

- What is UPB?
- UPB device setup and configuration
- Powerline Interface Module (PIM)
- Network import
- Generic UPB devices
- Device properties
- HCA support for scenes and device command features
- Program triggers for UPB events
- Hints and tips

---

### What is UPB?

UPB is a powerline carrier technology created by Powerline Control Systems (PCS). It allows commands to be sent over the powerline wiring in your home. Signals generated by a UPB transmitter, for example a keypad, can be received by the computer. HCA can also “listen in on” the communication between UPB devices and act on that communication, or simply log the activity.

To use HCA with UPB devices and keypads you need the UPB Powerline Interface Module (PIM). This interface lets HCA send and receive UPB messages using the powerline. HCA supports both the serial and USB version of the PIM.

If you are familiar with X10 devices (like the SceneMaster product line also from PCS) and the features in HCA to support them, you will be able to use UPB devices without much change to your thinking. Support for keypads is a bit different and is covered later in this appendix. If you are unfamiliar with the features of HCA for creating an automation solution, we suggest you review the other sections in of the HCA User Guide.

---

### UPB device setup and configuration

UPB devices are highly configurable but HCA does not provide configuration facilities. The reason for this is that UPB devices have so many settings and options that setup needs to be handled by its own Windows application. UPB manufacturers provide the application called UPStart for UPB configuration.

Documentation on how to use UPStart and instructions on how to configure all of the device settings is available from PCS.

Your first step before using HCA is to work with UPStart to configure your UPB network. You need not get it fully setup in all ways the first time but HCA has to have a network specification from UPStart before it can begin working with UPB.

Once you have installed and programmed all the devices using UPStart your next step is to export from UPStart a representation of the network. HCA imports this file to acquire all the information it needs to control all your devices and to receive signals from any device transmissions.

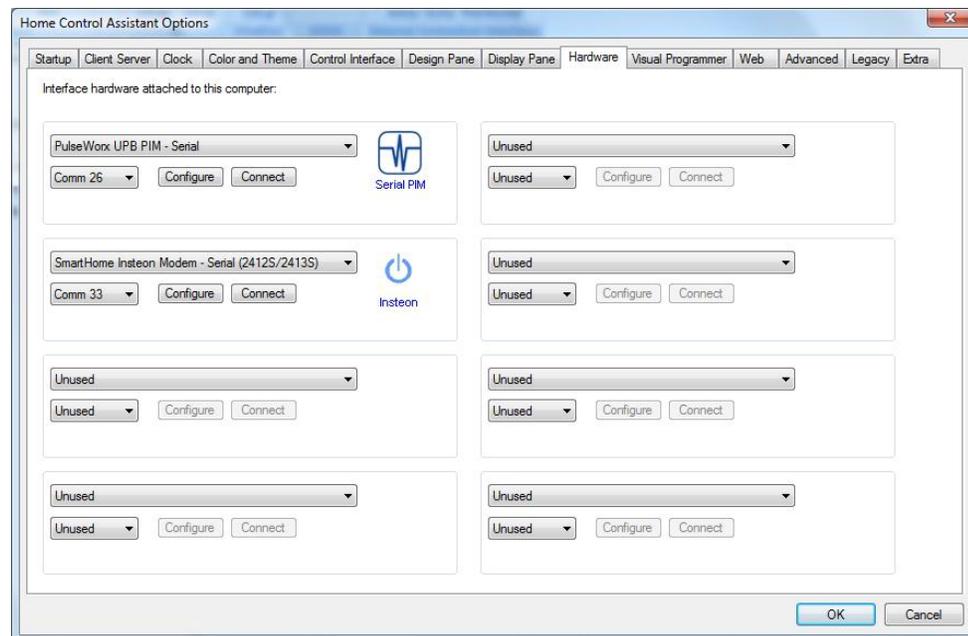
To create this export file, select from the UPStart menu: File – Export. You can store the export file in whatever folder that you want. All that is important is that it must be available for HCA to read either locally or over a network. Refer to the UPStart document for details on Export.

There are some tips and tricks described in the last section of this appendix that will help make your use of UPStart and HCA much simpler. Be sure to review that section.

---

## Powerline Interface Module (PIM)

Before HCA can send and receive UPB commands over the powerline, you must connect a Powerline Interface Module (PIM) to the computer HCA is executing on. Open the *HCA Options* dialog and choose the *Hardware* tab.



In the same way you identify all automation interfaces to HCA, all you need do is to select the interface type, communications port, and then press the Connect button. If HCA can read information from the PIM it displays a success message. Any problems with connection are noted in a failure message.

As you can see in this dialog you can attach up to eight automation interfaces to HCA simultaneously. This means that you can use a PIM for UPB messages, an X10 interface for X10 messages, a wireless interface for wireless messages, and even an IR interface for sending IR sequences. More on this later.

---

## Network Import

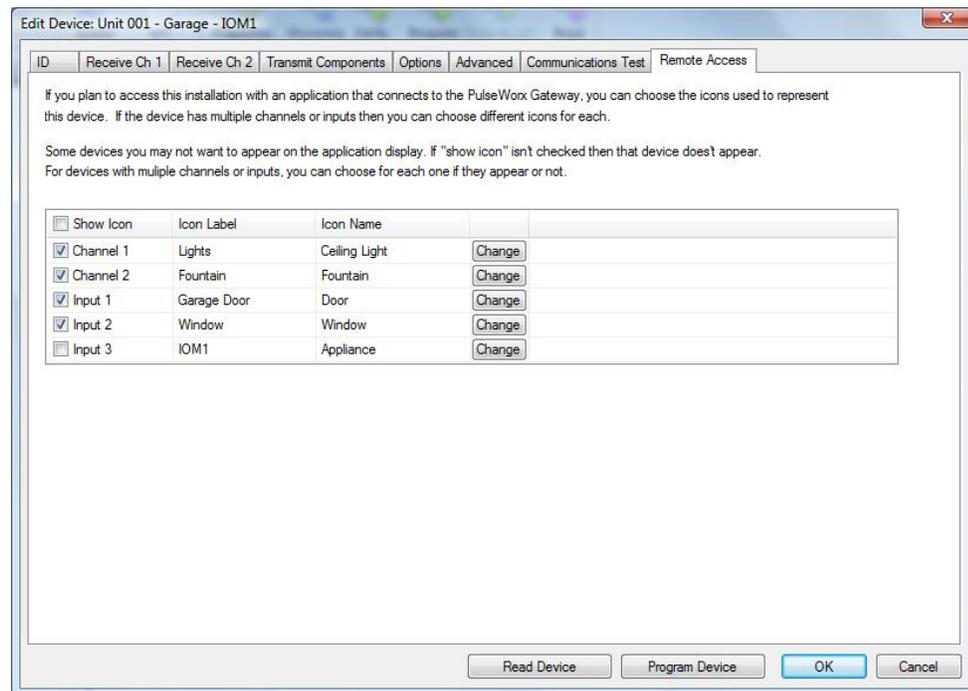
Now that HCA is talking to the PIM, and you have used UPStart to configure your devices and exported a definition of your network, the next step is to import that UPB network into HCA.

## Getting ready for export in UPStart

But before proceeding with the import of your UPB network into HCA you should first make some additional configuration in UPStart that will greatly improve the import into HCA.

In UPStart for each channel of a device you can specify the icon to represent the device as well as the text below the icon. Also for each channel you get to choose if there should be an icon seen at all. For example, if you are only using 1 input channel of a 3 input IOM you don't need to have icons for the other 2 inputs. This is all specified on the *Remote Access* tab of the device properties dialog.

Here is a screen image from UPStart. In this example, the configuration for this Input-Output module would cause 4 devices to be created in HCA – two for the loads and two for the inputs. No device would be created for input 3.



From the UPStart export file, these settings are also imported..

- Room icon choices. This is in the Application menu on the “Export” submenu.
- Keypad button names – specified on the “Engraving” tab in the properties for the PulseWorx keypads is now imported and used when showing the keypad in the Control UI popup.

Note that if you have previously imported your design, HCA does not change the HCA device name (the icon label) and icon selected for existing devices. The import doesn't want to override your icon choice you have made in HCA and also doesn't want to change the HCA device name which might break some HCA programs.

An excellent reason to take the time to define names for the parts of devices that have multiple channels comes when you use them in the Visual Program Test element.

As another example when using the input-output module, suppose that you were not using outputs 1 and 2 and only inputs 2 and 3. Also suppose you called input 2 “Washer” and input 3 “Dryer”.

In the test element you would then see these devices as choices.

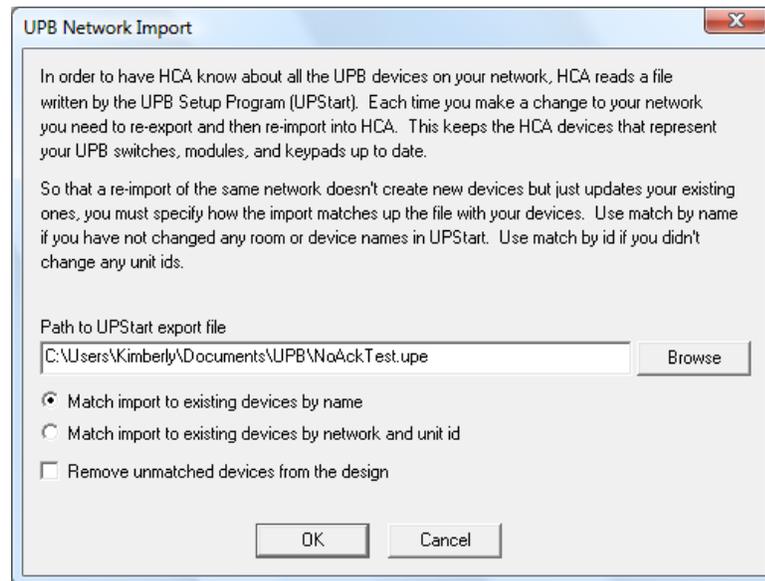
- Washer
- Dryer

This is more convenient than remembering "Input 2" or "Input 3".

---

## Importing a UPB network

To import a UPB network, press the *UPB Import* button in the ribbon *Protocols* category. When you do that, the following dialog displays:



Enter into the dialog three pieces of information:

- The path to the file created by UPStart for your network. This is the file that UPStart wrote when you selected File – Export. These files have a UPE type.
- The method that HCA uses to match what comes from the network export file with devices already in your design.
- If after the import, if any unmatched devices should be removed from your HCA design.

As described in the text of the dialog, each time you use UPStart to change your network it is essential that you export the network definition and then import that into HCA. If you don't do that, HCA may not operate correctly as it will have out of date information about how your UPB devices are programmed.

The first time you import, HCA creates new devices in your design to match what was in the import file. Subsequent imports don't want to create new devices – they want to update the existing ones. How does HCA know what needs to be updated and what needs to be added?

This is the purpose of the two match options seen on the preceding screen display. These options tell HCA how to match what it reads from the network import file with the existing devices in your design. It can do this in two ways:

- By name. HCA matches the existing devices by the room and device name from the UPB Id information

- By unit id. HCA matches the existing devices by the unique unit id given to any UPB device.

Which one should you use? It all depends upon what you changed in UPStart:

- If you modified device or room names, then have HCA match based upon the unit id.
- If you modified unit ids, then match based upon the names.
- If you modified some room or device names and some unit ids, choose the option that has the least impact on your design. For example, if you changed 4 names and 1 unit id, you'd probably match based on unit id, as only one mismatch occurs.

Once the import completes, HCA displays how many devices were added, updated, or removed.

It is important that HCA matches imported devices to existing devices so that any of your schedules and programs that use these devices don't get effected.

**Hint:** The UPB room name and device names matches well with the HCA concept of two-part names. The room name of each imported device is used for the folder, and the device name is used for the HCA device name.

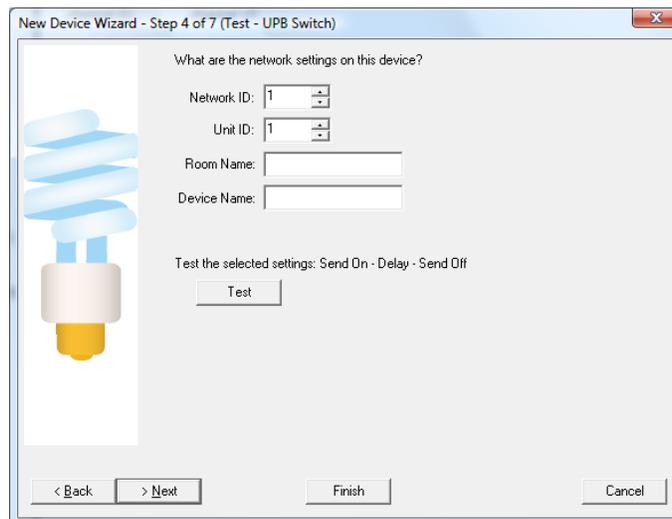
---

## Generic UPB Devices

The previous section described how you use the UPB configuration program called UPStart to add your devices to HCA using an import. This works well for almost all UPB devices.

There are, however, some UPB devices that UPStart doesn't configure. These are generally ones that do not access the powerline directly but rather connect to the powerline through the UPB Powerline Interface. This is the same unit as HCA uses to access the powerline for UPB. In this case the only way to add these sorts of devices to HCA is using the New Device Wizard.

In step 3 of the Wizard choose as the manufacturer "UPB" and select the device that most closely represents what you are adding. In the Wizard step 4 this dialog appears:



In this dialog you must enter the network id and unit id as well as the room and device names for the new unit. Once the device is added this way you can control it as a normal UPB device.

Why wouldn't you add all your UPB devices this way? Simply put, because HCA has no understanding of the properties of the device or what links are in its Receive Components Table. So if you can setup the device with UPStart do so and import from UPStart into HCA.

## Device Properties

Like almost everything in HCA, UPB devices have properties. Most of these are the same ones that other devices have in HCA: Name, Notes, Icon, Logging options, what displays the icon appears on, and restart options.

For UPB devices some UPB information is also shown. For example, selecting a UPB device, opening its properties dialog, and choosing the UPB Id tab shows the following:

The screenshot shows a dialog box titled "Test - Test Device channel (2)" Properties. The "UPB Id" tab is selected. The dialog contains the following fields:

- Room Name: Test
- Device Name: Device
- Unit ID: 1
- Network ID: 1
- Manufacturer ID: (PCS) Powerline Control Systems
- Product ID: (FMD2) Fixture Module - Dimmer
- Firmware Version: 05.53
- Channel: 2

At the bottom of the dialog are "OK" and "Cancel" buttons.

Some of the UPB Id information you are familiar with in UPStart is displayed here. While this information is useful, the most important information is on the Options tab.

The screenshot shows a dialog box titled "Kitchen - Hallway Properties". The "Options" tab is selected. The dialog contains the following sections and controls:

- Transmission:**
  - Confirm receipt of commands sent by ACK
  - Confirm receipt of commands sent by Status Request
  - Transmit count: Adjust count due to powerline noise amount
- Level and Ramp rate:**
  - Select defaults on level and ramp rate used when controlling a device from the UI and from a schedule. These could be the same defaults stored in the device but they need not be.
  - When using the Visual Programmer you can choose any of the available levels and rates. What is chosen here applies only to schedules and the User Interface.
  - When controlled ON use the last light level
  - When controlled ON use this light level: 100
  - When changing light levels use the default fade rate
  - When changing light levels use this fade rate: Snap!
  - To better work with linked keypad indicators, control this device ON using this link: Do not control by link
  - ... and OFF using this link: Do not control by link

At the bottom of the dialog are "OK" and "Cancel" buttons.

For a device these options can be set:

- Confirm receipt of commands by ACK. The UPB protocol contains an option that asks a device that is sent a command to respond back saying it was received. This is a very low overhead option. If the device doesn't ACK, then HCA retransmits the command.
- Confirm receipt of command by status request. This is an option built on top of the UPB protocol. Each time a device is sent a command to change illumination level (on, off, or dim) HCA follows up the command with a status request asking for the light level. HCA compares the two and if the device is not at the level it expects, it sends the command again.

Two important points about this. HCA doesn't send the status request immediately. It queues it for a later time when there are no pending transmissions. This means if three lights use this status confirm option and they are all sent a dim to 50% command, the status poll for the first happens after the 3<sup>rd</sup> light has been sent its 50% command. Secondly, use of these options adds a lot of extra powerline traffic. More traffic increases the chance of command collisions. Because of that, use this option carefully.

- Transmit count. As part of the UPB protocol, commands can be sent more than once and the devices know if these are repeats or not. You can use this to increase reliability of command reception. You can specify the number of repeats (1, 2, 3, or 4). The best way is to allow HCA to adjust the number of repeats based upon the amount of noise on the powerline. In general this means to be sent twice. If your UPB network has an active repeater HCA always sends each message a minimum of two times as needed by the repeater.
- Each command sent to a UPB device may contain the light level and fade rate. If the command sent doesn't specify the level or rate, the default values programmed into the device (by UPStart) are used instead. In this way you can program a light so that an ON command results in 90% illumination (or 80% or whatever). In addition you can program the amount of time – called the fade rate – it takes to change the light level.

Anytime you control a light, HCA can send a level and a rate that you choose or it can just let the default value in the device be used. That is what the options in the "Level and ramp rate" box are all about.

**Hint:** In the Visual Programmer you can have exact control over the level and rate for each command sent to the device.

- The last option changes how HCA sends messages to the device. Normally HCA just controls the device directly. However if you have keypad indicators that show the state of a device it would be better to control the device by a link. That way any indicator will also turn on or off as the device goes on or off.

---

## UPB Commands

If you are familiar with the X10 or Insteon powerline commands, UPB commands will take some getting used to. They are vastly different. You might be tempted to skip this section if all you want to do with your UPB devices is to make them go on, off, or dim. But UPB devices are capable of much more than X10 devices, so you may want to take a few minutes to fully understand the UPB commands what their effects are.

**Hint:** What follows is a very condensed and simplified explanation of UPB command. Refer to the documentation available on the PCS web site for more details.

There are five UPB commands that HCA works with. They are the Activate, Deactivate, Goto, Blink, and Request Status commands. But that's not the whole story. Most commands are sent in either of two formats: a direct command or a link command.

---

## Direct commands

The direct format is just about what you would expect. The command is sent to only one device and so affects only that device. The Goto, Blink, and Get Status commands can be sent as direct commands.

The Goto command is sent with a level (0 to 100%) and optionally a fade rate. If the fade rate is omitted, the default fade rate of the device is used.

The Blink command specifies the blink rate.

The Get Status command just asks the device to respond with its current level. For multi-channel devices, information on each channel is returned.

---

## Link commands

The link format takes a bit more explanation. Associated with any UPB device that can receive commands (like a switch or module) is what is called a Receive Components Table. This is similar to the stored presets in a SceneMaster or SwitchLinc device. Each entry in the Receive Components Table contains a level and, if the device is dimmable, a fade rate.

How would you use these presets? Below is a table representing three switch-controlled lights, with three different settings, for three different uses:

Link Name	Wall Light	Table Lamp	Big Light
TV Time	80%	50%	
Late Night	20%	20%	10%
Conversation	60%	80%	60%

Using UPStart you program the three switches for these lights. For the switches that control the Wall and Table lights, three presets are used. For the switch that controls the Big Light, only two are needed.

When do these different presets get activated? When the switch receives a Activate command that includes what is called a Link Id. What is a Link Id? It is just a number between 1 and 250. Since it can be difficult to remember numbers, in UPStart you should assign these link ids a name. Select from the UPStart menu: Network – Link Names.

In this example, when the "TV Time" link is received by the Wall Light it changes to 80%, when received by the Table Lamp it goes to 50% and when received by Big Light, nothing happens. Similar actions happen when the Late Night link and the Conversation link are received.

One way to think of Activating a link is to tell all of the devices in your UPB network this:

*When receiving an Activate command, if you have a link named \_\_\_\_ in one of your presets, then respond as that preset tells you to. If you don't have a link named \_\_\_\_ in one of your presets, then do nothing.*

The Deactivate command says a similar thing:

*When receiving a Deactivate command, if you have a link named \_\_\_\_ in one of your presets, then go off. If you don't have a link named \_\_\_\_ in one of your presets, then do nothing.*

And if that was the end of the story UPB devices would be very capable. But you can do more! The Goto and Blinks commands described above can also be sent in the link format. When sent like this, devices respond as:

*When receiving a link format Goto command, if you have a link named \_\_\_\_ in one of your presets, then respond as specified in the goto command ignoring the level and rate in the Receive Components Table. If you don't have a link named \_\_\_\_ in one of your presets, then do nothing.*

Using all these commands in the Link and Direct format you can do quite a bit with UPB commands.

Just to complete this discussion, the Activate and Deactivate commands can only be sent in the Link format. The Get Status command can only be sent in the Direct format. The Goto and Blink commands can be sent in either format.

---

## HCA support for scenes and device command features

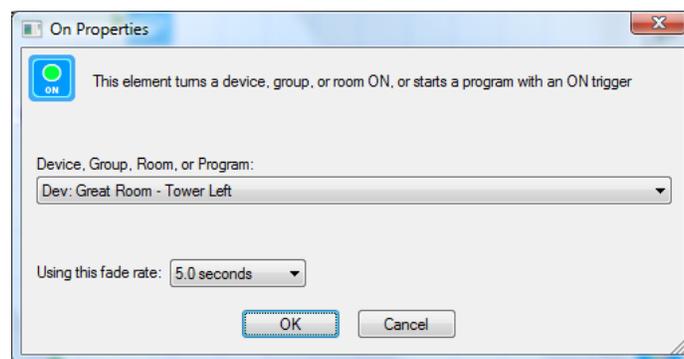
As explained above, when using UPB devices you have the option of working with a single device – by sending a command in a direct format – or with a set of devices – by sending a command in the link format.

---

## Direct commands

Using UPB devices in HCA is just about like using any other device type. They can be included in schedules and controlled by programs. About the only difference from other types of devices is that UPB devices can't be members of a group.

In addition to control by a schedule, you can control a device using the Visual Programmer On, Off, and Dim elements. When working with UPB devices, the Visual Programmer elements have an extra parameter:



In this example the UPB device is told to turn ON with a 5 second fade rate. The fade rate can also be specified when using the OFF and DIM elements.

In addition to the ON, OFF, and DIM commands the Visual Programmer also has a Blink element that sends the blink command to a UPB device. That element lets you set the blink rate.

Since all UPB devices respond to status requests, when using the Visual Programmer Test element to test for on, off, or dim, sends a status request to the device and performs the test based upon the status returned.

There is one additional method in HCA that works with UPB direct commands and it applies only to keypads. In this case what you control are the keypad indicators. To do this the Visual Programmer ON element is used. For example:



In this example, the Library Keypad F indicator is turned ON. To turn it OFF, the OFF Visual Programmer element can be used.

While this section is entitled "Direct Commands", a direct command may not always be used with the device selected in the ON, OFF, or DIM elements. If you have configured in the device properties, as explained above, to use a link then the command is sent in a link packet.

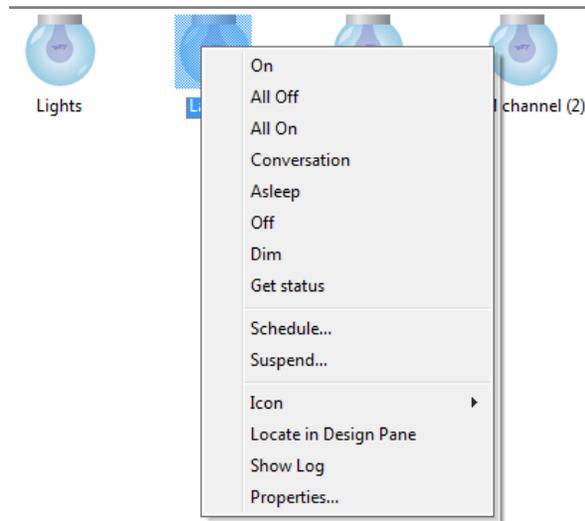
To better work with linked keypad indicators, control this device ON using this link:   
 ... and OFF using this link:

---

## Scene or Link commands

In addition to controlling a single device with a direct command, you can also control a set of devices with a single command by sending a link command - in the HCA world called a *Scene*.

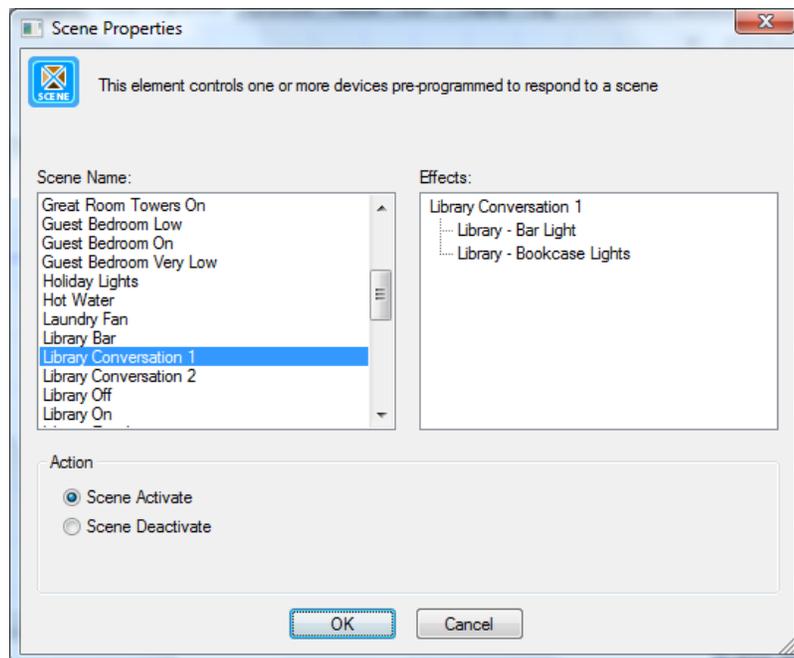
When used in HCA, the link names created in UPStart are called scene names. After you import a UPB network and right click on a device, in addition to the ON, Off, and Dim selections you also see all the scenes that the device responds to. For example:



In this example the UPB device responds to the scenes named "All On", "All Off", "Conversation", etc. These names are the same as seen in the link table in UPStart.

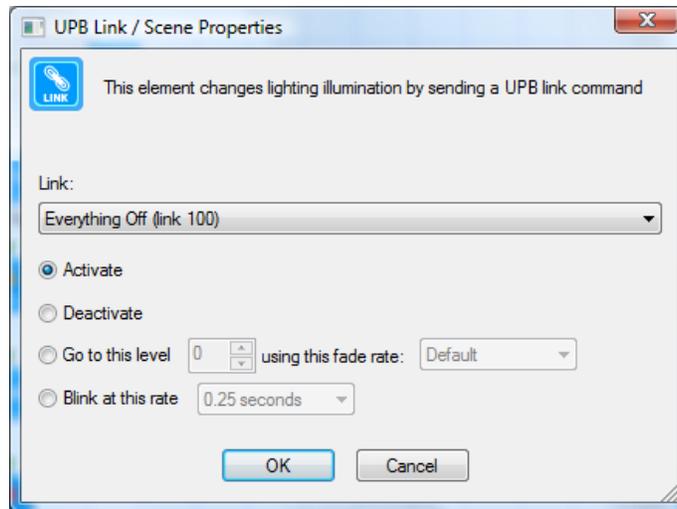
Selecting one of these scene names in the popup menu causes HCA to send a UPB Activate command using that scene. This causes the selected device *and any other device* that responds to that scene to change their illumination level to match what is programmed for that scene.

If you want to send scene commands in a program, you can use the Visual Programmer Scene element:



In this example, all devices that respond to the scene *Conversation*, changes their illumination level as setup in their preset for that scene.

The Visual Programmer Link element lets you send link commands as well:



All links imported from UPStart are listed. Choose a link and the command to send.

## Program triggers for UPB events

Keypads and input modules transmit UPB commands when a button is pressed (keypads) or a circuit opens or closes (input modules). HCA can respond to those actions and initiate one of your programs. In this way you can use a UPB keypad to start a program and that program can act with the full range of HCA services from turning on lights to sending email.

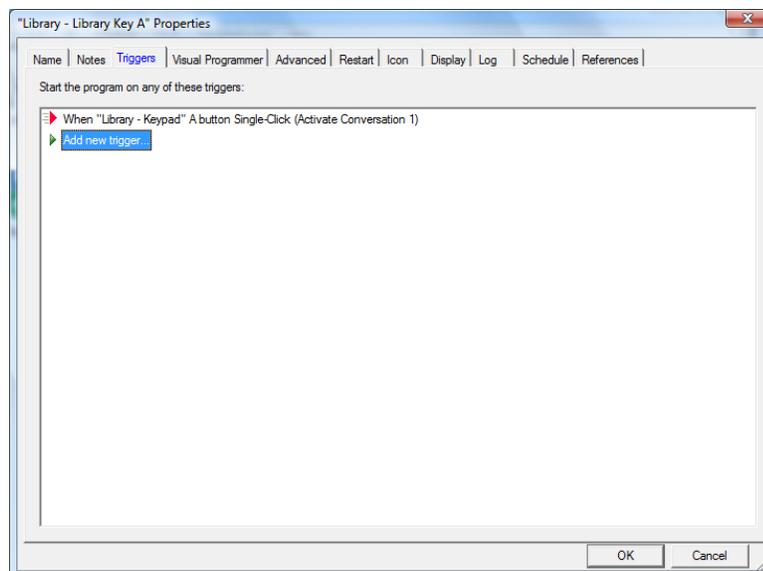
**Hint:** Everything in this discussion applies to both keypads and input modules like the doorbell sense module and the telephone sense module. It also applies to the commands that a rocker switch paddle can be programmed to transmit.

If you are familiar with how keypads work in the X10 world, you may find UPB keypads much more capable but more complex to work with. Each button depression on a UPB keypad sends a UPB command. What that command is and what it does depends upon what was programmed in to it using UPStart. Each button on a keypad and each action of that button (single-click, double-click, hold and release) can be programmed to send a different command.

At first glance UPB keypads can appear to be a difficult to work with. What you **want** to say when using a UPB keypad to start a program is this: *If the user single-clicks this button then run this program.* But that is not that way things work in the UPB world. HCA isn't told which keypad button was pressed and what action was done (single-click, double-click, etc) – HCA only sees the command it sends.

So how do you create a program trigger in HCA that describes a UPB command? Also what happens if, in using UPStart, you change what the command associated with a keypad button is?

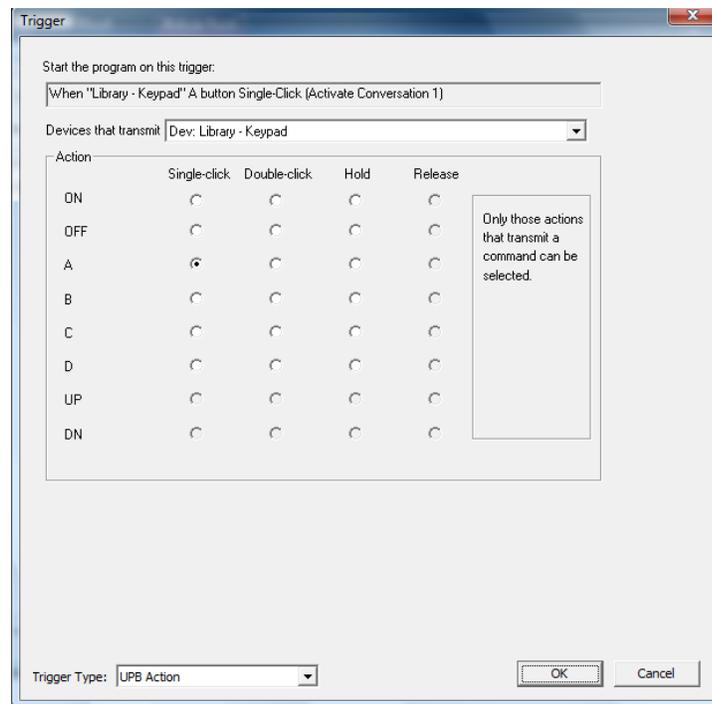
There are two ways UPB triggers are created for programs. Just like other program triggers it starts on the Triggers tab of the program property dialog.



There are two different types of UPB triggers you can create. UPB Action and UPB Powerline Message triggers.

## UPB Action Triggers

When creating a UPB Action Trigger, the dialog appears as:



This dialog lets you choose any of the available buttons for the keypad and the four different actions that can be done to that button. At the top of the dialog it shows not only what you selected – in this example, the Library Keypad A button single-click action – but also what that button was programmed to do – in this example, activate the *Library Bar Conversation* scene.

Using this dialog it *appears* that you are creating a trigger that starts a program when someone single-clicks the Library Keypad A button. But is it? The explanation above went to great lengths to say that this was impossible.

So what is *really happening* is that HCA creates a trigger that starts a program when it receives the *Activate Library Bar Conversation scene* command from the Library Keypad. It looks like you are selecting a button and an action, but in reality HCA creates a trigger based upon the command it knows (from the UPStart export) that button action sends.

So why all this discussion and why do you need be concerned if HCA handles all these details? Because if two buttons, or one button and two different actions, send the **same** command, HCA can't tell which button or action sent the command. For example, suppose that the A button single-click action and the A button double-click action are programmed for the same command. If you create a trigger that says: "When Library Keypad A button single-click" and someone single-clicks *or* double-clicks the button, your program starts. Is that wrong? No. Unexpected? Maybe. Why does it happen? You said in your trigger "single-click" but the double-click action sent the same command and HCA is triggering on the command **and not** the button action so HCA can't tell the difference between a single-click and a double-click in this case.

**Hint:** UPB triggers are tricky. Better read this section over again, and then experiment with UPStart and HCA before settling on a final UPB design.

There are two additional details when working with UPB triggers:

First, if you have a button configured to send different commands on each press of the button – known as a toggle - HCA creates two triggers: one for the Activate and one for the Deactivate. In UPStart, the *Super Toggler* mode does this.

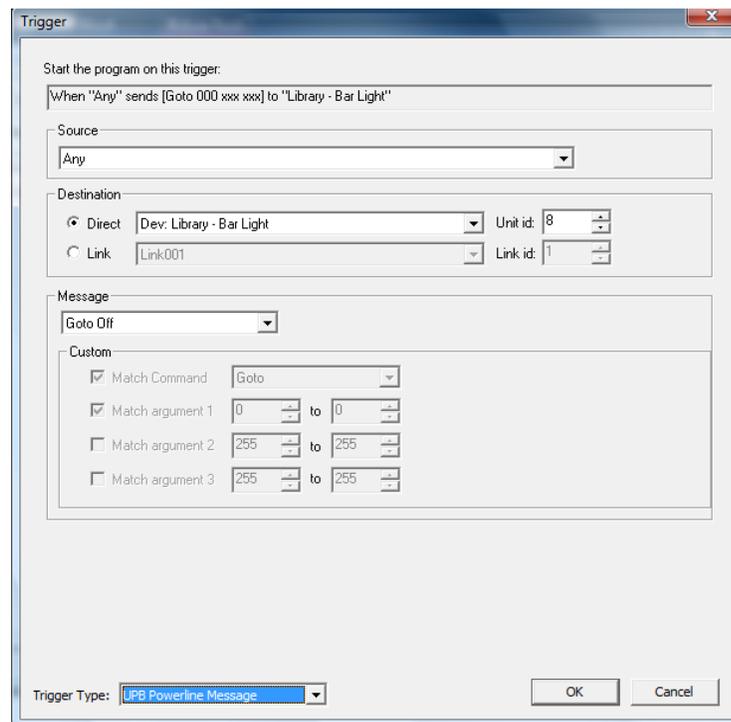
Second, what happens if you change the command associated with a keypad button and you already have a trigger defined for that button? When you re-import your design, as part of the import process the trigger is updated to match the new command programmed for that button and action.

For example, keypad button A when single-clicked sends the command: *Activate Late Night Scene* and the HCA program *Go to bed* has a trigger for keypad button A. Now suppose using UPStart you change button A to send *Activate Early Morning scene* when single-clicked. When you re-import the network file, as part of the import the trigger is updated. Your program still starts when keypad button A is single-clicked even though the command changed.

---

## UPB Powerline Message

Unlike the simpler UPB Action trigger, this type of trigger is more powerful but more complex. Creating a trigger of this type displays this dialog:



To use this type of trigger you must have a working knowledge of UPB powerline commands. This requires that you read and understand the UPB message and command documents available from Powerline Control Systems.

What you specify in this dialog is each component of the command that you want to trigger on. These components are:

1. The source device. This is specified by selecting one of the UPB devices in your network.
2. The destination. This could either be a link (select a link name) or a direct command (select a device).
3. The parts of the message. This is the command and its arguments. You can choose if the trigger must match the command and/or its arguments and the permissible range of values for those arguments. Using this drop down in the Message box you can select the more common commands or by using the controls in the custom box you can completely specify a UPB command.

---

## Hints and Tips

### Using UPStart

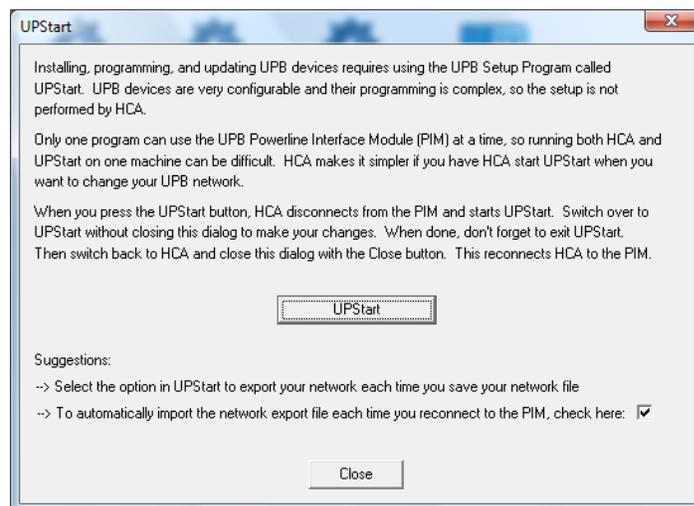
When just starting out with a UPB network you may often find yourself wanting to switch back and forth between UPStart and HCA. This can be a problem as only one application can use the PIM at one time. To make it simpler, HCA essentially allows you to “toggle” between HCA and UPStart if you use the following method:

1. First, in UPStart open the options dialog by selecting from the UPStart menu: Tools – Options. There are two options that you should enable.

**Reopen last file loaded when UPStart begins.** This makes it simpler to get back to your network file each time UPStart starts.

**Auto Export on File Save.** After you export from UPStart the first time, UPStart remembers the path you exported to. This option re-exports to that same file each time you do a save. This makes sure that the export file always contains up to date information.

2. Then, whenever you are in HCA and you want to work with UPStart, press the *Setup* button in the *Protocols* ribbon category. This dialog appears:



Press the UPStart button. At this point, HCA relinquishes its control of the PIM, then UPStart starts and its main window appears on your screen. Use UPStart to make your network and device modifications and when done, exit UPStart with File - Exit. Since in the first step above you selected the “Auto Export on File Save”, by exiting UPStart you are automatically saving and exporting your UPB network. Then switch back to HCA and close this dialog with the Close button.

Using this method allows UPStart to use the PIM – HCA disconnects before it starts UPStart - and then has HCA take back control of the PIM when UPStart is closed.

So you don't forget to import after you are done with UPStart, there is a checkbox at the bottom of the dialog that tells HCA to re-import when UPStart is done. If you use this technique to jump between HCA and UPStart, you probably want to enable this option.

### **Switch transmit**

One very nice feature of some UPB devices is that they can be told to transmit their current status each time they are locally controlled. For example, each time someone uses a switch paddle to turn the light on, off, raise or lower the illumination level, the switch transmits its new level.

This option is called *Report light level after rocker switch is pressed* and it is on the Options tab when editing the device using UPStart. Using this option makes sure that HCA is always up to date with the light level on the device and makes for much more accurate automation programs. It is a good idea to enable this option on all your switches.

### **Multiple Interfaces**

Don't forget that HCA can use multiple automation interfaces simultaneously. In this way you can use X10, wireless and IR devices along with your UPB switches and keypads in your automation design. For example, a HCA program, started from a UPB keypad trigger, can send commands to both UPB devices and X10 devices. Or a wireless motion sensor can be used to start a program that sends UPB commands.

While two or more simultaneous X10 signals can collide on the powerline – two or more simultaneous UPB signals can also collide – UPB and X10 signals can't collide with each other. X10 and UPB can coexist well in your home.

Review the sections of the User Guide and Appendixes on wireless, IR, and X10 devices for more information.



## Appendix 9

### Insteon

This appendix describes the features in HCA in support of the Insteon technology and Insteon products available from SmartHome. These topics are covered:

- What is Insteon?
- Insteon devices
  - Adding an Insteon device
  - Modifying an Insteon device
- Insteon PowerLinc interface
  - PowerLinc address database
  - PowerLinc Interface models and Insteon firmware
- Insteon Tools
  - Network Multi-Add
  - The Insteon linking model
  - PowerLinc swap
  - Network Capture
  - Network Map
  - Network Clean
  - Device Replace
  - Changing local level and ramp rate
- Device and keypad linking
  - Device Linking tabs
  - Multi-Way Wizard
- Scene Control without Scenes
- Insteon message triggers for programs
- What HCA knows of the Insteon network and what it doesn't know

---

#### What is Insteon?

Insteon is a powerline carrier technology created by SmartHome Design. Like X10 and UPB it allows commands to be sent over the powerline wiring in your home. Signals generated by an Insteon transmitter, for example a keypad, can be received by the computer. HCA can also “listen in on” the communication between Insteon devices and act on that communication, or simply log the activity.

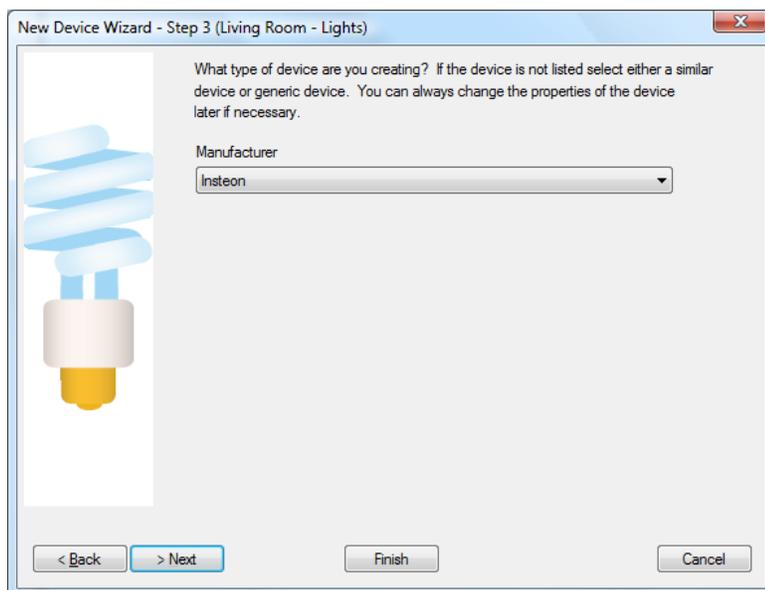
To use HCA with Insteon devices and keypads you need the USB Insteon PowerLinc (model 2413) available from SmartHome. This interface can also be used for X10 communication as well.

## Insteon devices

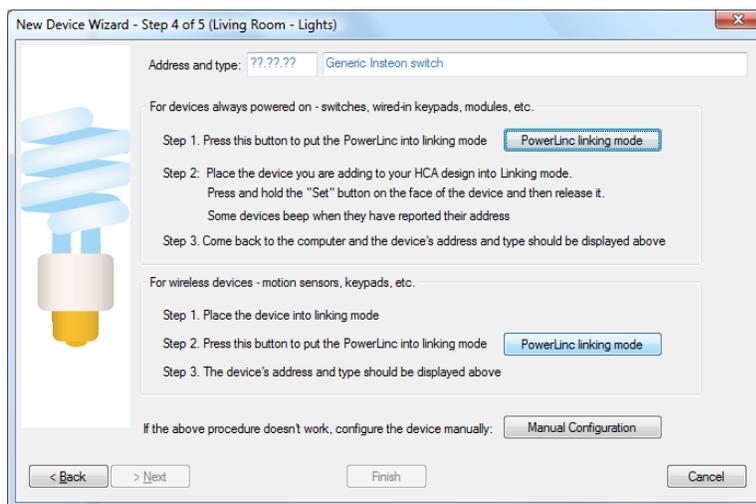
Insteon devices come from the factory with an address permanently already assigned. This address is a three part number and is on a sticker someplace on the device. When you add an Insteon device to your HCA design that address is captured from the device in a step of the New Device Wizard.

### Adding an Insteon Device

In step 3 of the New Device Wizard you choose the type of Insteon device. Select as the manufacturer "Insteon".



In the wizard step 4 the Insteon address of the new device is determined. There are two ways to do this.



The best method is to have HCA capture the address from the device when it is placed into linking mode.

As the text in the dialog says, if the device is wired-in, press the top "PowerLinc linking mode" button and the PowerLinc is placed in linking mode – the more recent versions of the PowerLinc beep when you do this. Next, go to the device and press and hold its "set" button until the link is made. Again, some more recent device types beep when you do this. When the link is complete the address and type of the device displays in the dialog.

For devices that are wireless, place the device in linking mode first – press and hold there "set" button" and when it is in linking mode then return to HCA and press the second "PowerLinc Linking mode" button. The link is made and the device address and type displayed in the dialog.

The second method is to enter the device address and type manually. **You should only do this as a last resort as the address capture method builds an important link in the device that is needed for HCA to control it.**

Insteon Device Manual Configuration: Living Room - Lights

What is the Insteon address of this device?

Address: 00 . 00 . 00

Note: Put wireless devices into linking mode to turn on their receiver before pressing the Test button

What is the type of this device?

Kind  
Switch

Model number and description  
Generic Insteon switch

The Test button in this dialog verifies the address you entered. If it can't communicate with the device then this popup gives you options.

Insteon Device Add

The device at that address didn't respond to an Insteon version request.

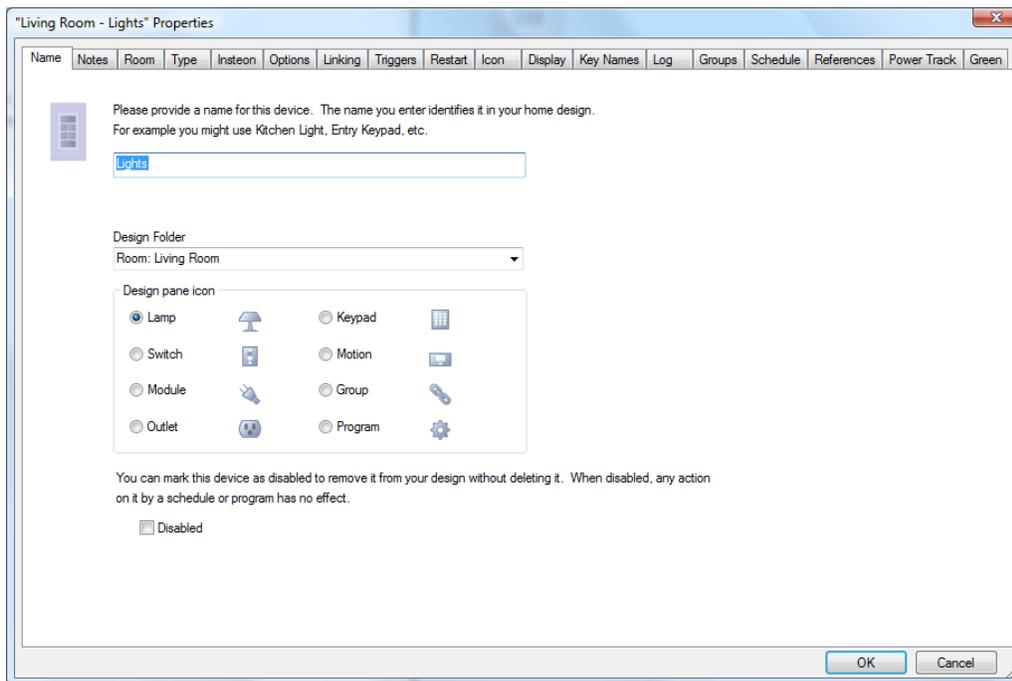
This could be because there is no device at the address you entered or the device is of a newer firmware version that will not respond until it is linked to the PowerLinc.

You have these options:

If you are sure the address entered was correct, then accept it and continue the wizard. If you think this is a device containing the newest firmware, try using the *Attempt to link* button.

## Modifying an Insteon Device

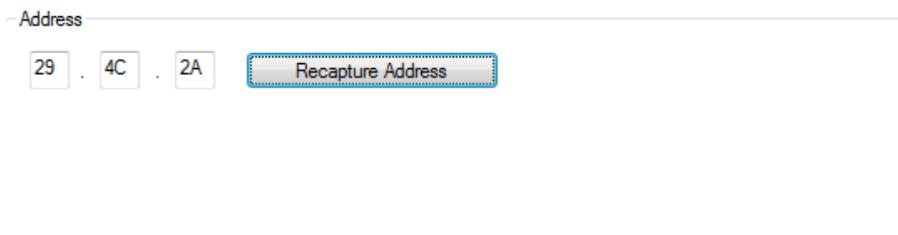
Like other device types, to modify the properties of a previously created Insteon device, select its icon or its name in the design pane, right-click and select Properties.



One very nice feature of Insteon devices is that they all work just about the same. All Insteon devices respond to the same commands in the same manner. As such, unlike some other device types there is no place in HCA to set any options that describe a device's function – the options tab only gives you options on how the icon operates.

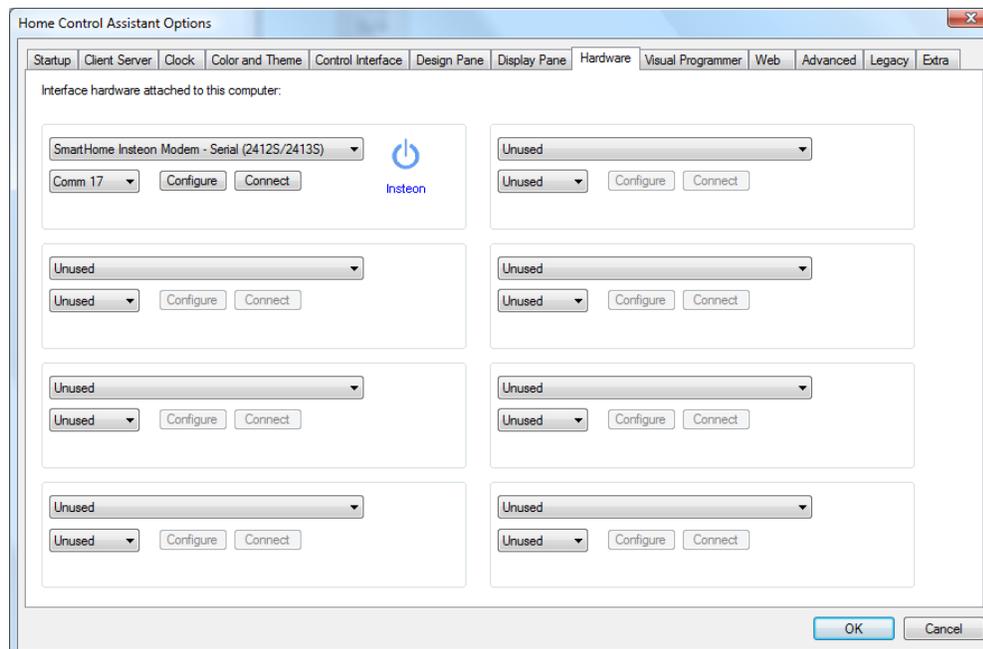
One final word on Insteon devices: Make sure the address captured by the New Device Wizard or in the device Property dialog matches the address on the device. Pay particular care that you enter it correctly or HCA will not be able to communicate with the device.

If you factory reset the device you must recapture its address. There is a button for this on the "Insteon" tab.



## Insteon PowerLinc Interface

Before HCA can send and receive Insteon commands over the powerline, you must connect an Insteon PowerLinc to the computer HCA is running on. Open the *HCA Options* dialog then choose the *Hardware* tab. When you do that this dialog appears:

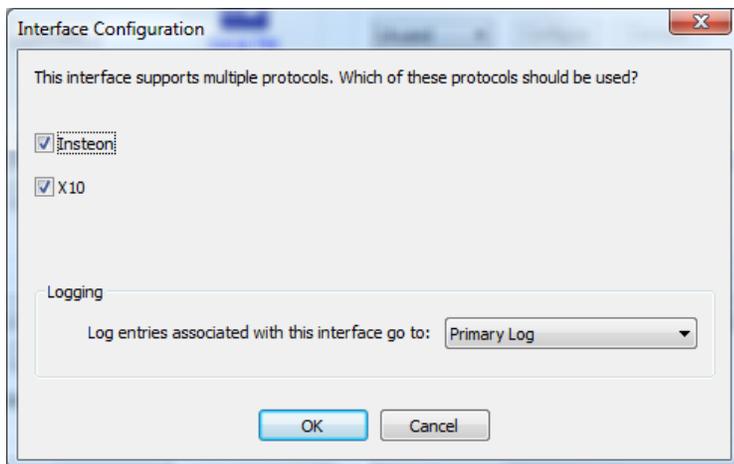


HCA Supports both the current Insteon PowerLinc (2413). HCA also supports the older model PowerLinc (2412 and 2414) but these are not recommended. The 2414 can't send and receive the commands needed by the most current Insteon devices. This is described in greater detail in a later section in this appendix.

**Hint:** There are many different PowerLinc controllers, make sure you select the correct one!

In the same way you identify all automation interfaces to HCA, all you need do is to select the interface type, communications port, and then press the Test button. If HCA can read information from the PowerLinc it displays a success message. Any problems with connection are noted in a failure message.

As you can see in this dialog, you can attach up to eight automation interfaces to HCA simultaneously. This means that you can use other interfaces as well for UPB and Wireless. One feature of the Insteon PowerLinc is that it can be used for X10 devices as well. To enable this press the Configure button to choose one or both of the supported protocols.



On this dialog you can also choose the log used for this interface.

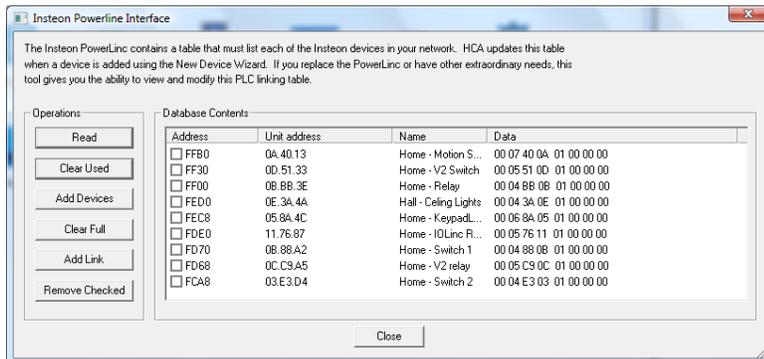
---

## PowerLinc Address Database

Unlike other power line technologies, the Insteon PowerLinc will only pass on messages from devices it knows about. This can help prevent you from receiving messages from other users near your home. But it may mean an extra step for you.

As devices are added to your HCA design they are automatically added to the database in the PowerLinc, so normally you need not be concerned about this. The problem arises if the PowerLinc is reset or you get a new PowerLinc to replace a failing unit, or you move your design to another computer with a different PowerLinc.

HCA can rebuild the PowerLinc table. Click on the *PowerLinc* button in the ribbon *Interfaces* category and select from the menu *PLC Utilities*. This dialog appears:



The operations are:

- **Read:** Read and display the PowerLinc database
- **Clear Used:** Clear any entries that are marked as used. Can only be done after a read.
- **Clear Full:** Clear all entries in the PowerLinc database. This can take several minutes.
- **Add Devices:** Add to the table the addresses of all of your Insteon devices. Use this operation if you replace the PowerLinc with a new one.

---

## PowerLinc Interface models and Insteon firmware

HCA supports all models of the PowerLinc interface but not all have the same capabilities. Also, not all Insteon devices have the same level of firmware in them. You may be working with a PowerLinc model and device firmware that can present challenges.

Some Insteon firmware has a requirement that the device's linking table include a responder link for HCA (actually the PowerLinc address but you know that) to have HCA even turn the device on or request its status.

During the New Device Wizard, when you get to the Insteon Address step there are two options as described above. The address capture method is key as it also builds the needed responder link in the device that the new firmware needs.

But there are problems. Here is what works and what doesn't and why

- Version 1 firmware is the newest Insteon devices.
- Version 2 is newer firmware
- Version 3 is the most recent – at the time of HCA 11 release – firmware.
- This firmware has the requirement as described above.

If you are using a 2413 PowerLinc, here are the operations that work with each device firmware version.

	Capture	Test
V1	ok	ok
V2	ok	ok
V3	ok	ok

If you are using a 2414 PowerLinc, here are the operations that work with each device firmware version.

	Capture	Test
V1	no	ok
V2	no	ok
V3	no	no

This means that if you are working with a device with version 3 firmware, and you are using the 2414 model PowerLinc you will have to manually link each device to the powerline or HCA can't even turn it on.

## Insteon Tools

This section describes a number of tools that work with Insteon devices. All of these tools are started from buttons in the *Insteon* section of the ribbon *Protocols* category.

### Multi-Add

One method to quickly add a number of already installed Insteon devices is to use Insteon Multi-Add. This dialog appears as:

The Insteon Multi-Add tool lets you add a number of already installed Insteon devices to your design.

Step 1: First press the "Start" button and then go to the first Insteon device you want to add and place it in linking mode. As soon as the device enters linking mode, the address of the device is captured. Then move to the next device and repeat the process. You can add up to 32 devices during this step. Not all wireless devices work with this tool.

You should take notes on the order of the devices you went to so when you return to the computer you can name them and assign to rooms or folders.

	Address	Model - Product	Version	Room Name	Device Name	
1						<input type="checkbox"/> Ignore
2						<input type="checkbox"/> Ignore
3						<input type="checkbox"/> Ignore
4						<input type="checkbox"/> Ignore

Step 2: Press the Get Types button to query each device for its version and type

Step 3: Replace the default device and room names with more descriptive ones. Select the type for any device listed as Unknown. If there was a problem with the discovery process - always a possibility - you can check the ignore box and an HCA device is not created for it.

Step 4: Press OK to create the HCA devices. Once created you may want to open their properties and change the icon and other HCA settings.

Here is how this works: Press the start button and then walk around your home putting devices into linking mode. Keep track of the location of the devices and the order you do them in!

Once the Start button is pressed the dialog tells you to begin.

The Insteon Multi-Add tool lets you add a number of already installed Insteon devices to your design.

Step 1: First press the "Start" button and then go to the first Insteon device you want to add and place it in linking mode. As soon as the device enters linking mode, the address of the device is captured. Then move to the next device and repeat the process. You can add up to 32 devices during this step. Not all wireless devices work with this tool.

You should take notes on the order of the devices you went to so when you return to the computer you can name them and assign to rooms or folders.

	Address	Model - Product	Version	Room Name	Device Name	
1						<input type="checkbox"/> Ignore
2						<input type="checkbox"/> Ignore
3						<input type="checkbox"/> Ignore
4						<input type="checkbox"/> Ignore

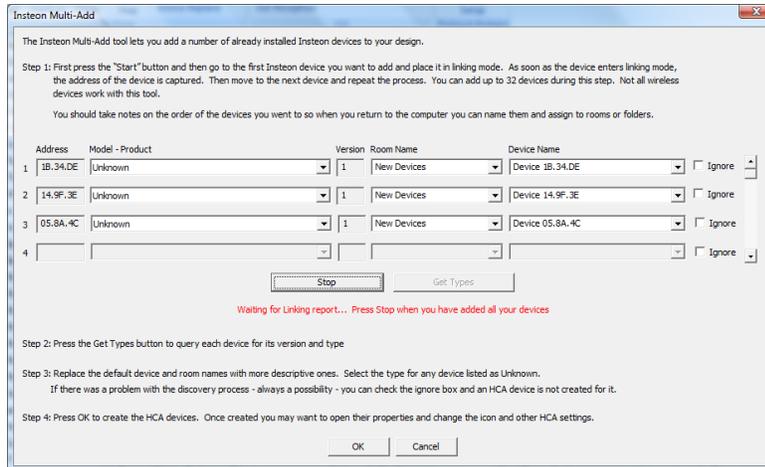
Waiting for Linking report... Press Stop when you have added all your devices

Step 2: Press the Get Types button to query each device for its version and type

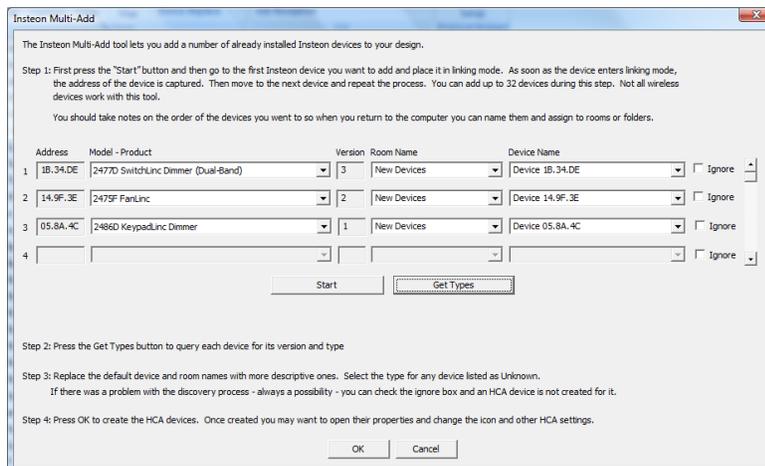
Step 3: Replace the default device and room names with more descriptive ones. Select the type for any device listed as Unknown. If there was a problem with the discovery process - always a possibility - you can check the ignore box and an HCA device is not created for it.

Step 4: Press OK to create the HCA devices. Once created you may want to open their properties and change the icon and other HCA settings.

Here is the dialog after three devices have been added. Each added device appears on its own row. That section of the dialog scrolls and you can add up to 32 devices at once although, as said below, you shouldn't try to do that many unless you are very fast and have a good memory.



When you have completed all your additions, press the *Stop* button and then press the *Get Types* button to start the next step of multi-add. HCA requests the status of each device and from that determines its type.



To complete the add operation, enter a device name and room name for each device and then close the dialog with OK.

### A few notes on using Multi-Add

- You have only a limited time to complete the process before the PowerLinc exits linking mode. It is best to do only a few devices at one time.
- If you find that you have added a device more than once or made some other error, just "tick" the ignore box next to the device to it and it isn't added to your design.

---

## The Insteon linking model

Described in the next sections are a number of tools in HCA to help manage your Insteon Network. These tools are for reading, writing, modifying, and displaying links between devices, so a short tutorial on Insteon links may be helpful.

If you are familiar with how keypads work in the X10 world, you may find Insteon keypads much more capable but more complex to work with. Each button push on an Insteon keypad sends an Insteon message. The destination of that message depends upon what was linked to that button.

Links can be created by a manual method. When you press and hold the keypad button until it goes into linking mode, the keypad sends out a message that says "My address is this and who would like to join my group 'n'?" Where 'n' is the button number.

At the lamp module you want to respond to that keypad button you press and hold its set button for a few seconds. That action sends a message to the keypad saying "Yes, I want to join your group n". The keypad then sends back a message saying "OK you're in" and then the lamp module acknowledges that message. Now linking is complete.

What happened is that the memory of the keypad and the lamp module were updated. The keypad added an entry to its internal memory that says "Group n has as one of its members the device addressed as xx.xx.xx". The lamp module updated its memory to say "If I receive an ON message from yy.yy.yy I should go to the level I was at when I was linked in".

In this example we used a KeypadLinc and a lamp module but the same things apply if you were linking with a SwitchLinc – as either a controller or receiver.

**Hint:** The preceding paragraphs are a bit simplified but it is correct enough for this discussion. See the SmartHome technical information for the complete story.

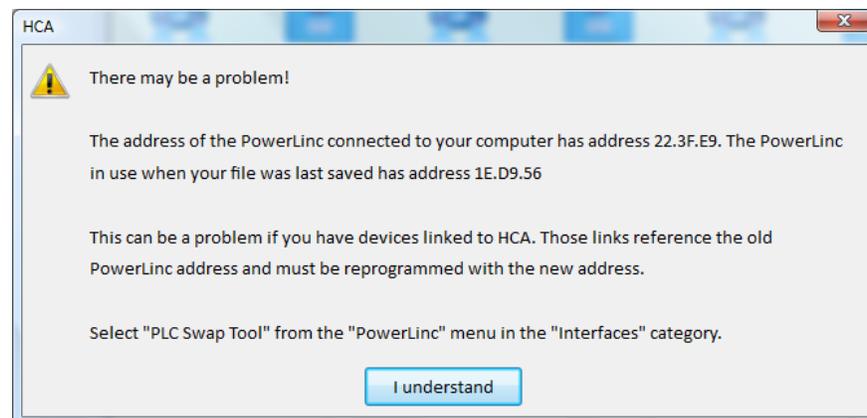
HCA can read and write these linking tables from all your devices.

This is important for two reasons: HCA can then understand all the connections between all your devices and, more importantly, be able to create new ones. This means that using HCA you need not spend time running around *pressing and holding* and *tapping set buttons*. In most cases, HCA can do it all for you.

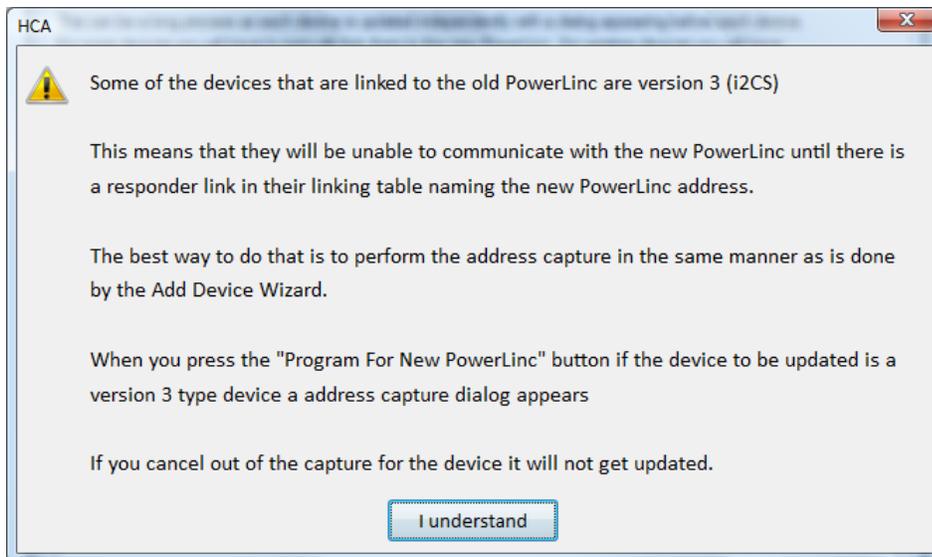
## PowerLinc Swap

The Insteon linking model described above presents a major challenge. Since each device that is linked to HCA is really linked to the PowerLinc used by HCA, if you change that PowerLinc all the links in all your devices that reference the old PowerLinc address will stop functioning. HCA tries to detect when the PowerLinc has changed and provides a tool to reprogram all your devices with the new PowerLinc address.

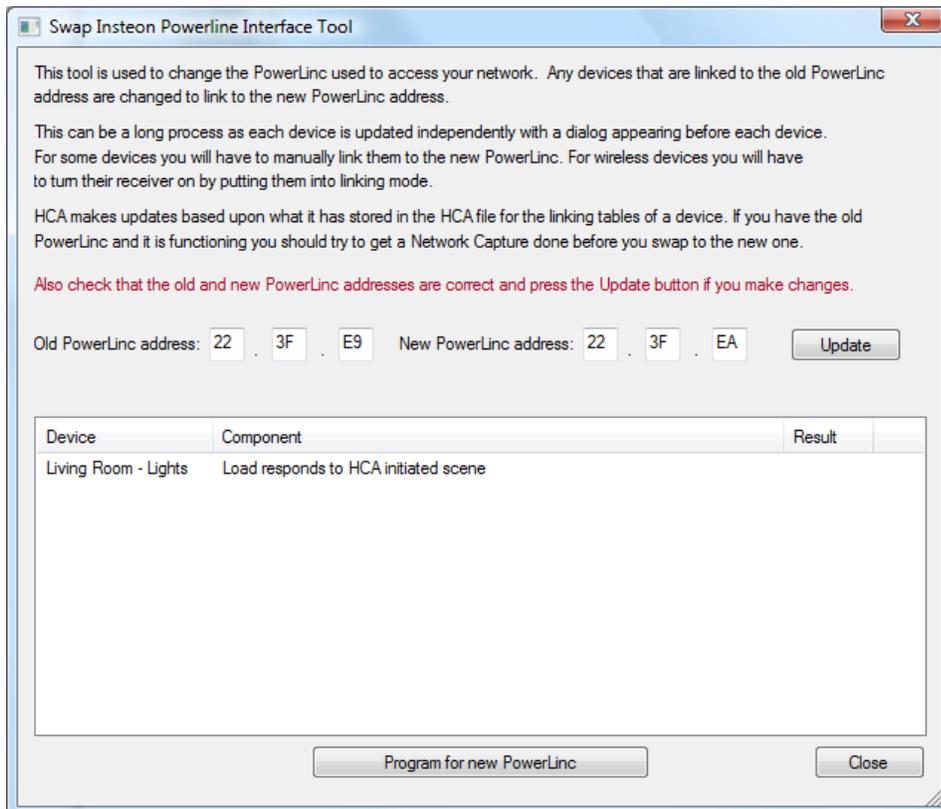
When HCA detects a PowerLinc change, this dialog appears when you load your design file.



And it is even a bit worse than that. In the section above that explains that a "responder link" is needed in most devices before HCA can control it, the existing responder link is to the old PowerLinc and this needs to be updated. The only method to do this is to recapture the address of all devices.



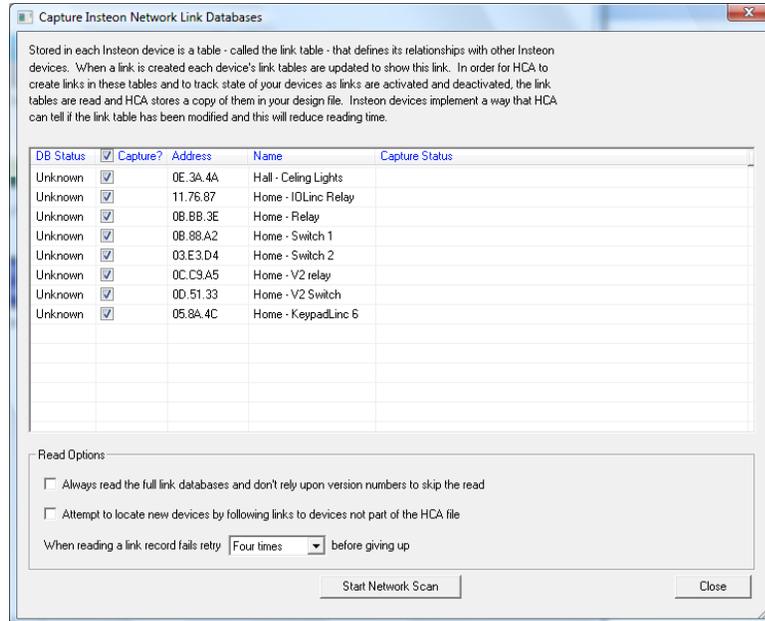
The Swap tool shows what needs to be programmed.



Carefully enter the old and new PowerLine address and then start the operation. Changing the PowerLine is not a simple operation. This isn't HCA's fault but just an outcome of the way that Insteon works.

## Network Capture

The Network Capture tool reads the linking databases out of each Insteon device. The Network Capture tool is started by pressing the *Network Capture* button in the ribbon *Protocols* category. This dialog appears:



This is the most important tool of all! All the other Insteon Tools – Network Map, Network Clean, Device Replace, and the Visual Scene Editor – rely upon having complete and correct knowledge of all devices' linking databases.

When HCA reads the database of each device and copies its contents to the HCA design file, it also reads the version number of the database. The version number is updated as links are added, deleted, or modified. This lets HCA skip the reading of a device if it can determine if the database already matches what it has in the design file.

The database version for scene-capable Insteon a device is stored in its memory and gets reset to zero on factory reset or power cycle. If you have a power outage, all of your devices will get reset to version zero.

HCA has to assume that any device with Database version zero has been reset or modified and forces a database re-read before programming it again. Upon first reading a device database with a zero version number, HCA writes to it in order to change the version to one so that it does not have to re-read it again. You may notice that all devices have a database version of one after the first network scan after a power outage.

Programming a device changes the database version number. The final version number is determined by the number of bytes written to it (not links, but bytes). The actual number of bytes written varies depending on what changes are being made to the links. HCA only writes the bytes necessary, which saves a lot of time.

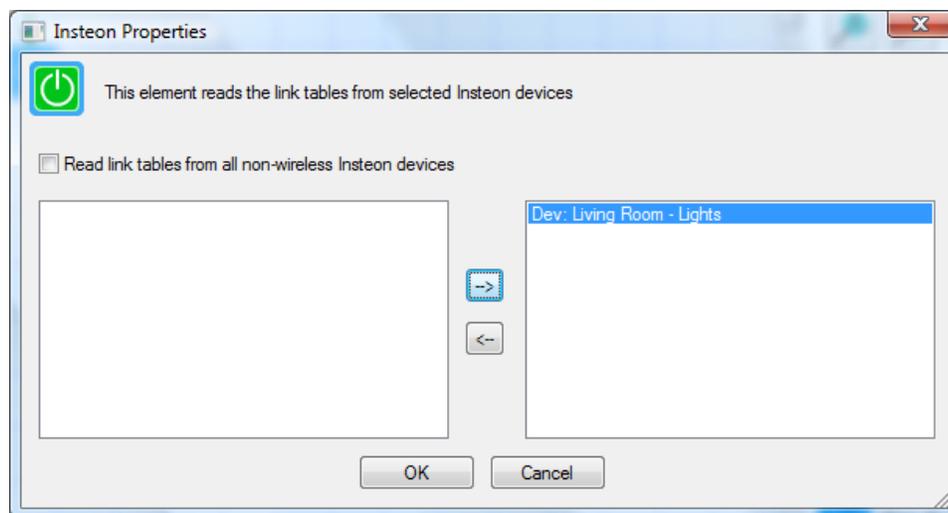
Occasionally the database number increments back to zero (roll-over), in which case HCA forces a re-read prior to programming the device again.

Reading out the database may take some time. If you have lots of devices you may want to choose only some of them to read. To do this, use the checkboxes in front of each device's name.

The same Remote Memory Access popup dialog used by all the other Insteon tools appears when you press the Start Network Scan button.

**Hint:** In some large networks the number of message repeats can be very large. This can really slow down the reading of linking databases. On the HCA Properties dialog Advanced tab is an option that sets a time limit on how long HCA will work at reading an entry from the linking database before giving up,. You may need to increase this timeout if you find that the Network read is reporting timeouts too frequently.

There is a visual programmer element that initiates the reading of a device linking table. You can use this in a program that you schedule to run infrequently – once a week or once a month – to make sure that the linking tables are up to date. It is not recommended that you don't do this too frequently as it takes a lot of network access time and the messages sent to read the devices can collide with other uses – like turning things on and off.



---

## Network Map

The network map is a textual representation of your Insteon network. Displayed are what each device controls and what it responds to. To open the network map, press the *Network Map* button in the ribbon *Protocols* category

Name	Component	Info	Scene Name
Computer Room - Ceiling Light	05.78.B8 ON button A button	Database version: 1 Controls 05.FC.8D ** Controls Computer Room - Desk Lamp Controls 05.FC.8D **	Ceiling Light.ON > (many) Ceiling Light.A > Desk Lamp
Computer Room - Desk Lamp	04.99.A3 Responds to	Database version: 1 Computer Room - Ceiling Light Button A 05.FC.8D (group 101) **	
Dining Room - Thermostat	0E.62.19	Database version: 25	
Family Room - Loveseat Lamp	00.4C.DA Responds to	Database version: 1 05.FC.8D (group 101) **	
Family Room - Remote Controller	0D.D5.56	Database version: 25	
Family Room - Sofa Lamp	00.4C.D3 Responds to	Database version: 2 05.FC.8D (group 101) **	
Full Bath - Exhaust Fan	05.A6.B8 Switch paddle	Database version: 1 Controls 05.FC.8D **	Exhaust Fan > (many)
Full Bath - Vanity Lights	07.26.F4 Switch paddle Responds to	Database version: 1 Controls 05.FC.8D ** 05.FC.8D (group 101) **	Vanity Lights > (many)
Kitchen - Ceiling Fan/Lights	07.27.82 Switch paddle Responds to	Database version: 35 Controls Kitchen - Ceiling Fan/Lights Slave 2 Controls Kitchen - Ceiling Fan/Lights Slave Kitchen - Ceiling Fan/Lights Slave Switch Paddle Kitchen - Ceiling Fan/Lights Slave 2 Switch Paddle 05.FC.8D (group 101) **	Ceiling Fan/Lights > (many)
Kitchen - Ceiling Fan/Lights Slave	07.27.2D Switch paddle Responds to	Database version: 28 Controls Kitchen - Ceiling Fan/Lights Slave 2 Controls Kitchen - Ceiling Fan/Lights Slave Kitchen - Ceiling Fan/Lights Slave Switch Paddle	Ceiling Fan/Lights Slave > (many)

This is a great method to document your network – a very handy thing in case something breaks and you need to re-establish some links. You can get a printout of this in one of two ways. The standard HCA printing features (*Application menu – Print - Printout Setup*) can print the network map. Or if you want to format it differently or use your own tools, you can save the map in CSV form using the Copy to Clipboard function. You can then paste it into other applications – perhaps an Excel worksheet..

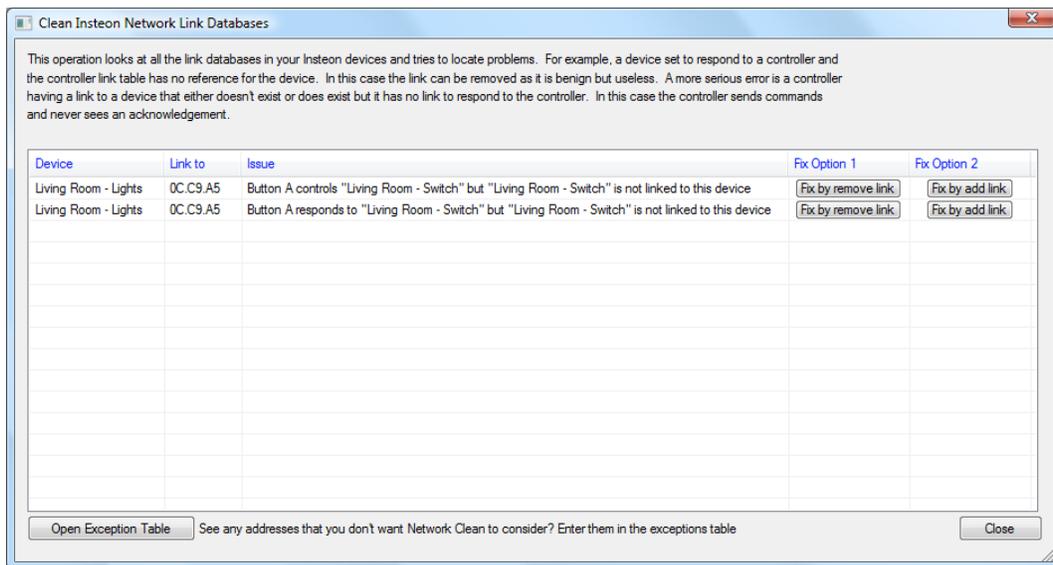
---

## Network Clean

It is easy for there to be problems with links in an Insteon network. If a device gets reset or you re-link a switch or keypad button and don't first remove the existing link you may wind up with problems.

For two devices to be correctly linked there must be a controller link in one and a responder link in the other. If one of the two is missing then the controller can't control the responder.

The Network Clean dialog analyzes your network and looks for problems. To perform this, press the *Network Clean* button in the ribbon *Protocols* category.



The Cleaning process corrects problems either by removing or adding links. For each error listed press the button for the kind of correction that you want to perform.

There may be some devices that, for whatever reason, you don't want the network clean tool to work with. These can be listed in the exception table. This has the effect of removing them from all consideration by the Clean tool.



**BIG Note:** Don't get too carried away with the network clean tool. While the rules of device linking say one thing there are exceptions. Insteon is not a perfectly consistent system. There are device types that seem to break these rules. Think before you act and, as has been said many times; if it isn't broken don't fix it.

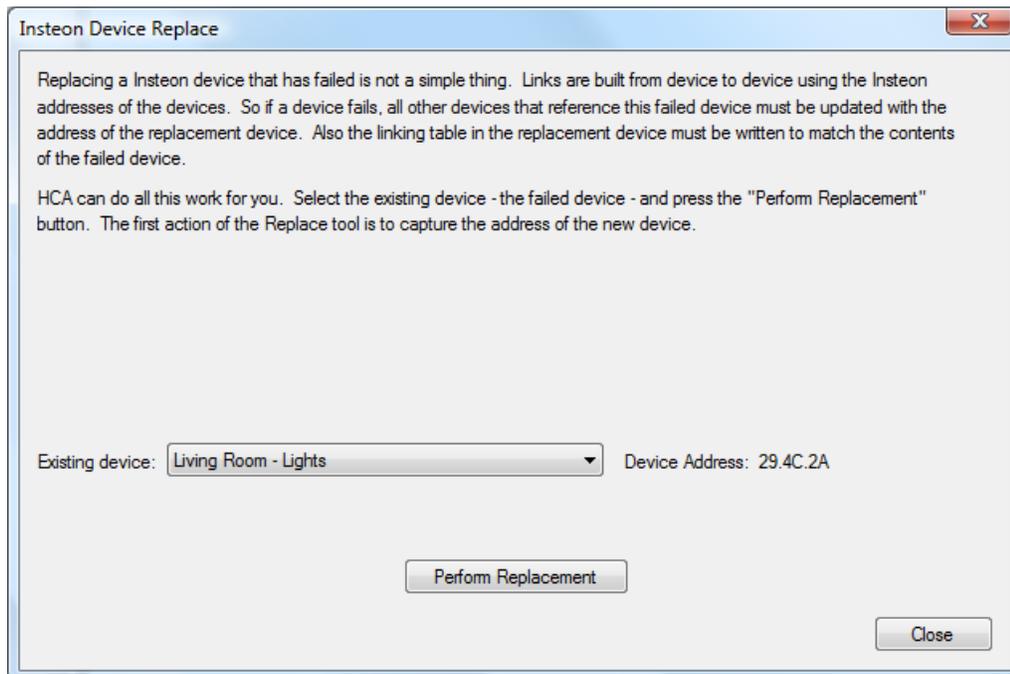
**Hint:** For the Network Clean tool to be able to detect and correct problems it needs to have the most current linking databases to analyze. Make sure that you use the Network Capture tool often.

---

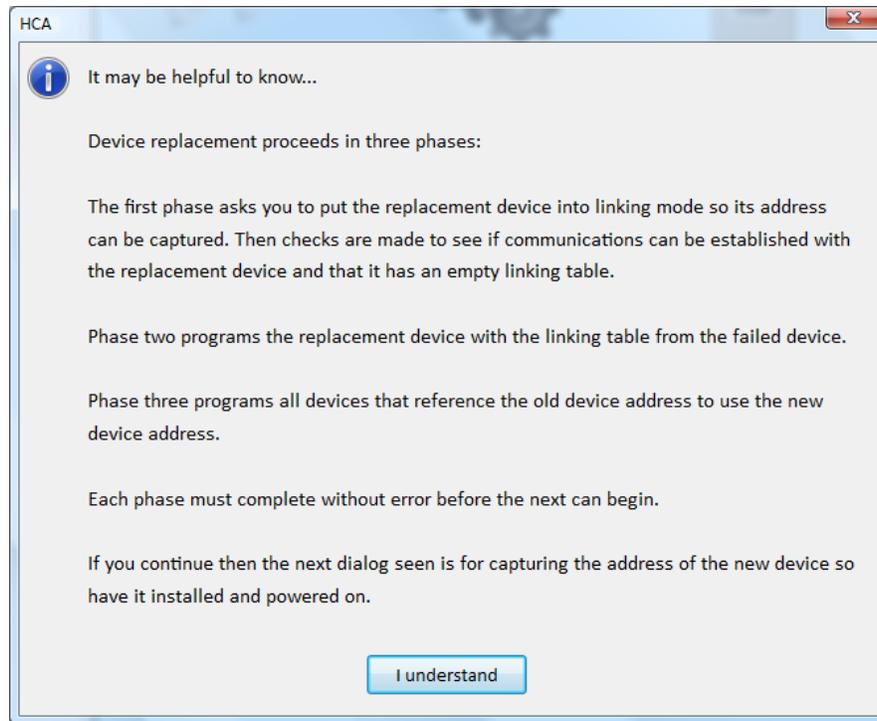
## Device Replace

A major problem in the Insteon linking model is what happens if a device fails. Not only do you have to replace the device but you have to reprogram it with all the stored links in it. And, as important, you have to reprogram any device that is linked to the failed device since those devices reference the old device's address. **This can be a long and complex operation.**

HCA can help with this. Press the *Device Replace* button in the ribbon *Protocols* category.



Starting the operation is simple: All you need do is to select the failed device – its Insteon address shows next to it - and press the Perform Replacement button. The procedure is explained:



Once the popup is closed then the operation proceeds as described. Each phase requires action on your part and subsequent popups show that.

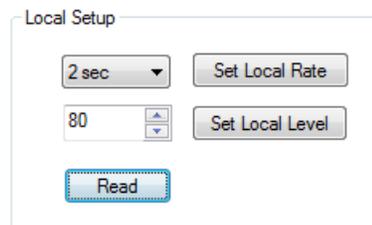
**Hint:** For the Device Replace tool to be able to correctly program the replacement device it needs to have the most current linking databases to analyze. Make sure that you use the Network Capture tool to keep HCA up to date. But you know that already!

---

## Changing Local ramp rate and Local Level

For some device types – switches and load controlling keypads– you can set the level and rate the load goes to when locally controlled. For switches when the paddle top is tapped, to what level and at what rate does it turn on? And for the paddle bottom, at what rate does it go off?

This is set from the Insteon tab of the device properties in the "Local Settings" box.



You can read the settings from the device, modify them, and then store the changes to the device.

**Note:** This is an area where Insteon is not consistent and not all device types support this nor do all device types – and firmware versions – operate the same. So if it works for your device then great. If not, you can always set these options manually – see the device documentation from SmartHome.

## Device and Keypad Linking

In the preceding section the subject of linking and linking databases has been covered in detail. But most of that description focused on reading and storing those tables into HCA. The section discusses the tools HCA has for creating links.

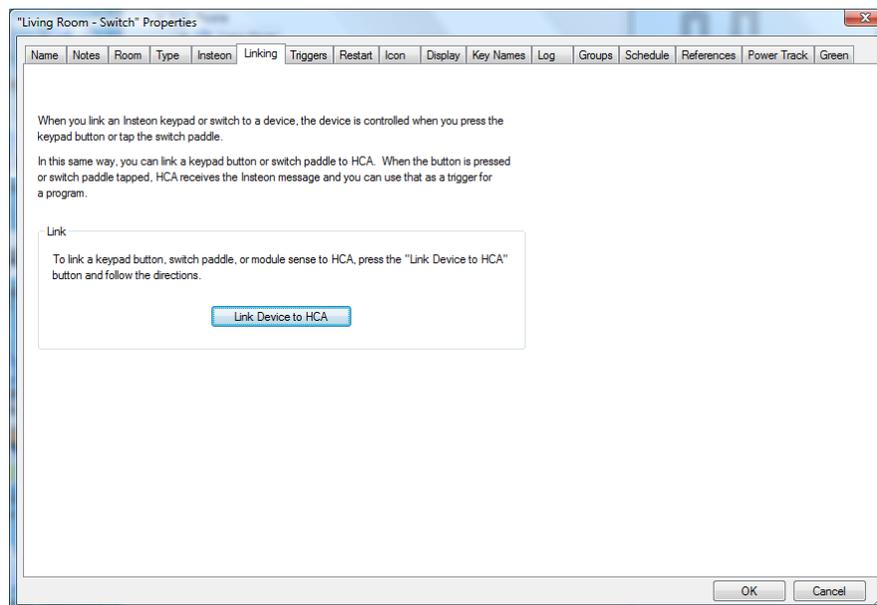
There are three methods:

- The linking tab in device properties. These are good for linking HCA to the device so that HCA can respond to signals from keypads and switches, and control KeypadLinc LEDs.
- The Visual Scene Editor. This visual tool is for creating links between devices that can be activated by HCA or by an Insteon device. This is covered in the VSE chapter of the User Guide.
- The Multi-Way wizard. This is a tool for creating multi-way associations. A multi-way association is a set of devices where if one is controlled (tapping the switch paddle or pressing a KeypadLinc button) all the others in the association are controlled as well.

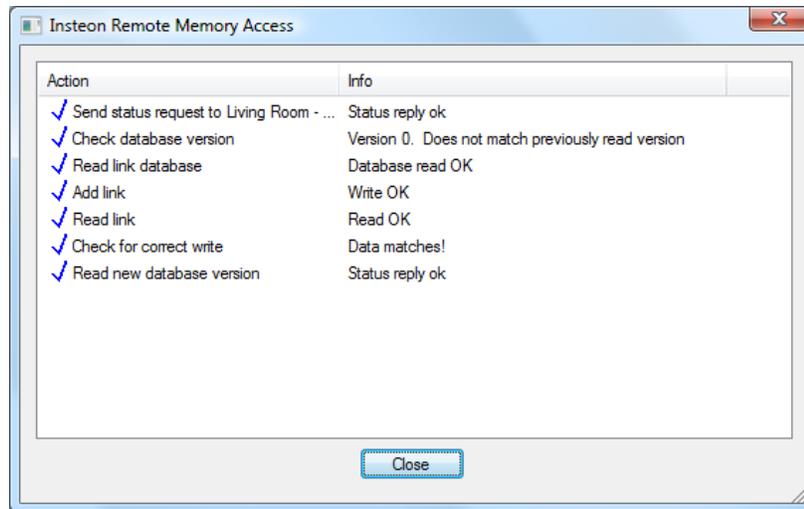
### Device Linking Tabs

While the Visual Scene Editor is a general tool that can be used to build links between devices and HCA, you may want to create links with HCA in a simpler manner.

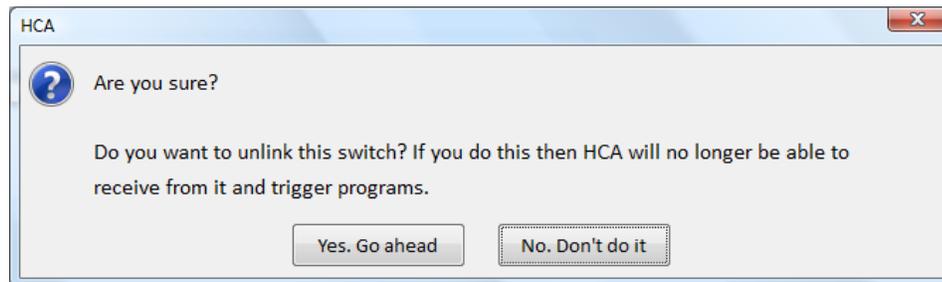
When you open an Insteon device property dialog, the linking tab appears as:



If you press the *Link Device to HCA* button HCA writes the necessary link to the device's linking table so that each time you tap the switch paddle, HCA receives a message. Once linked in this way, you can then create a trigger so that the Switch paddle starts a HCA program.

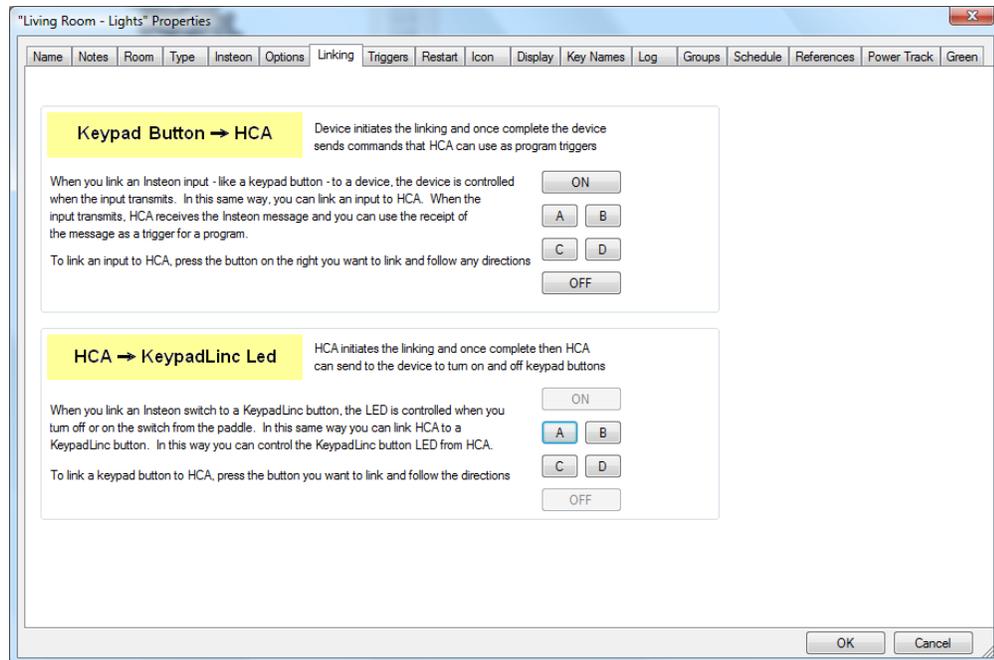


If the device is already linked the button text says *Already Linked to HCA*. If you press that button, this popup appears:



This gives you the ability to unlink an already linked device.

Linking a controller – like a KeypadLinc - is a bit more complex. The linking tab appears as:



This dialog is divided into two sections. The upper section is for linking buttons so that when pressed a message is sent to HCA. Like the switch paddle case, once a button is linked then a HCA program can be triggered by that button press.

The lower section of the dialog is for linking the KeypadLinc LEDs to HCA so HCA can send commands that turn the LEDs on and off. Unless HCA and the KeypadLinc LED are linked, HCA can't control the LED.

To create a link all you need do is to press the button and HCA will program the switch as needed. If the button or LED is already linked to HCA the txt on the button is HCA. Pressing it will offer you a chance to unlink the button or LED and HCA.

**Hint:** There is no magic here. The linking tab is only a quicker way to do what the Visual Scene Editor could also have done.

---

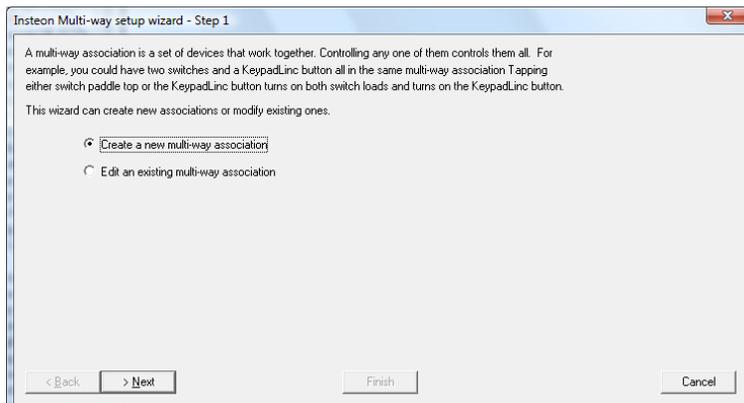
## Multi-Way Wizard

A Multi-way association is a collection of two or more devices that all work together. An example is best to illustrate this. Suppose you have a number of lights in your kitchen. There are separate switches but when you tap the paddle of one you want them all to come on. And it doesn't matter which one you tap – controlling any of them causes them all to come on. This collection of switches is a multi-way association. You can also add a KeypadLinc button – for the KeypadLinc load or not - to an association so that if you tap an associated switch, the KeypadLinc button illuminates to show that the association is on.

**The key concept of the multi-way wizard is that each member of the multi-way must be able to control all the others and respond to all the others. Since a module, for example, can only receive it can't be part of a multi-way.** Only devices that both transmit and receive can join a multi-way.

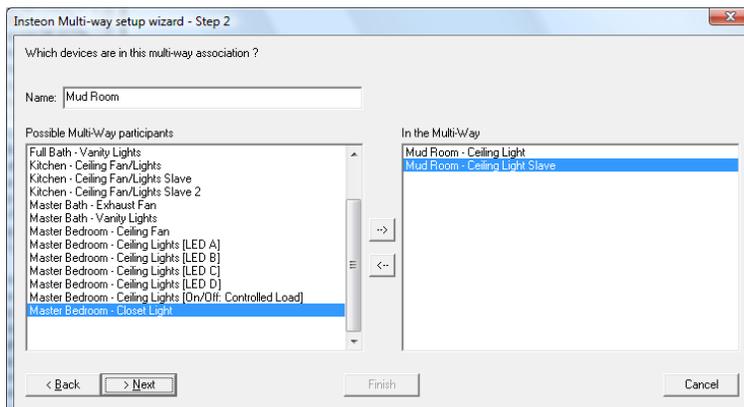
There is no magic in the Multi-Way wizard. All it is doing is creating links in each of the devices in the association. You could create this in the Visual Scene Editor but since the number of links can be large, the multi-way wizard may be more convenient.

To start the wizard, pres the *Multi-Way* button in the ribbon *Protocols* category. The first step of the dialog is:

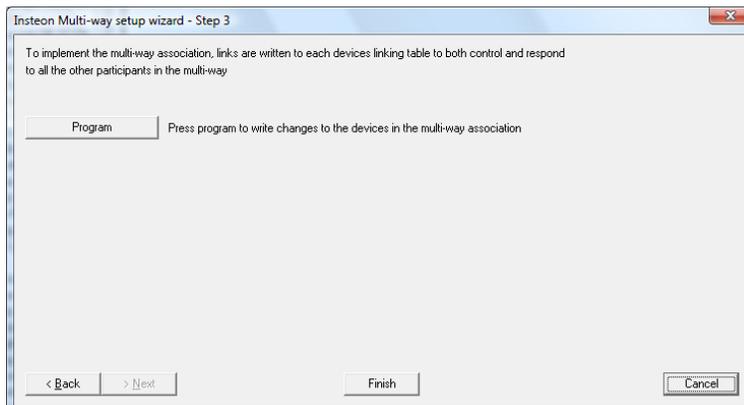


The multi-way wizard doesn't only create multi-way associations, you can also edit an existing association to add or remove members.

The next step in creating a new association is to select the devices you want to associate.

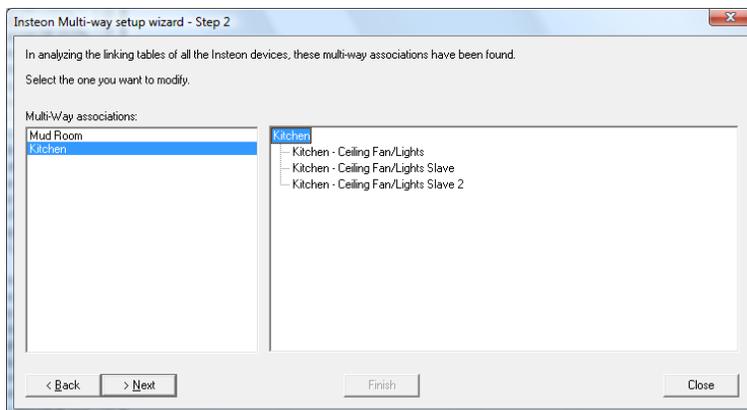


All you need do is to select what devices you want in the association. Use the → and ← buttons to move items between the two lists. When you have made your selections press the Next button.



Press the Program button and HCA will create all the necessary links.

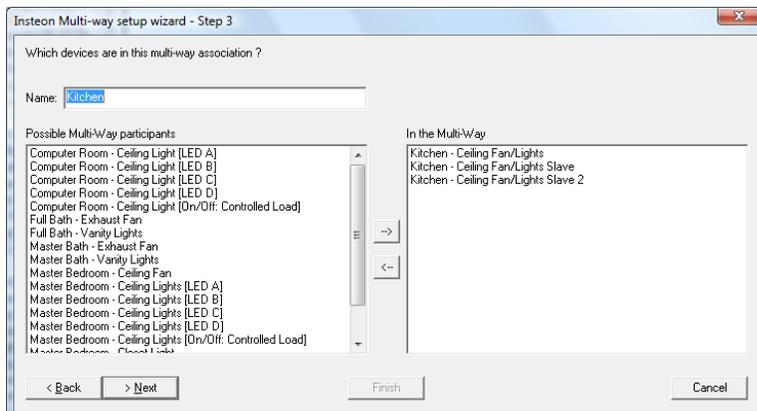
To edit an existing association, the second step of the dialog is:



HCA lists all the multi-way association it finds – by analyzing the linking databases of all your devices – and presents you with a list. When you select a multi-way in the left list, the right display shows what are the devices that make up the association. Select one and press Next.

**Hint:** If you want to rename the multi-way just click on the name in the right pane and HCA allows you to edit the name.

The next step of the edit is the same as when you are creating a new association:



You can add or remove devices from the association and then press Next. If you have made changes then the Program step appears and you can write those changes to your devices.

**Hint:** For the multi-way wizard to be able to edit existing associations it needs to have the most current linking databases to analyze. Make sure that you use the Network Capture tool often.

---

## Scene Control without scenes

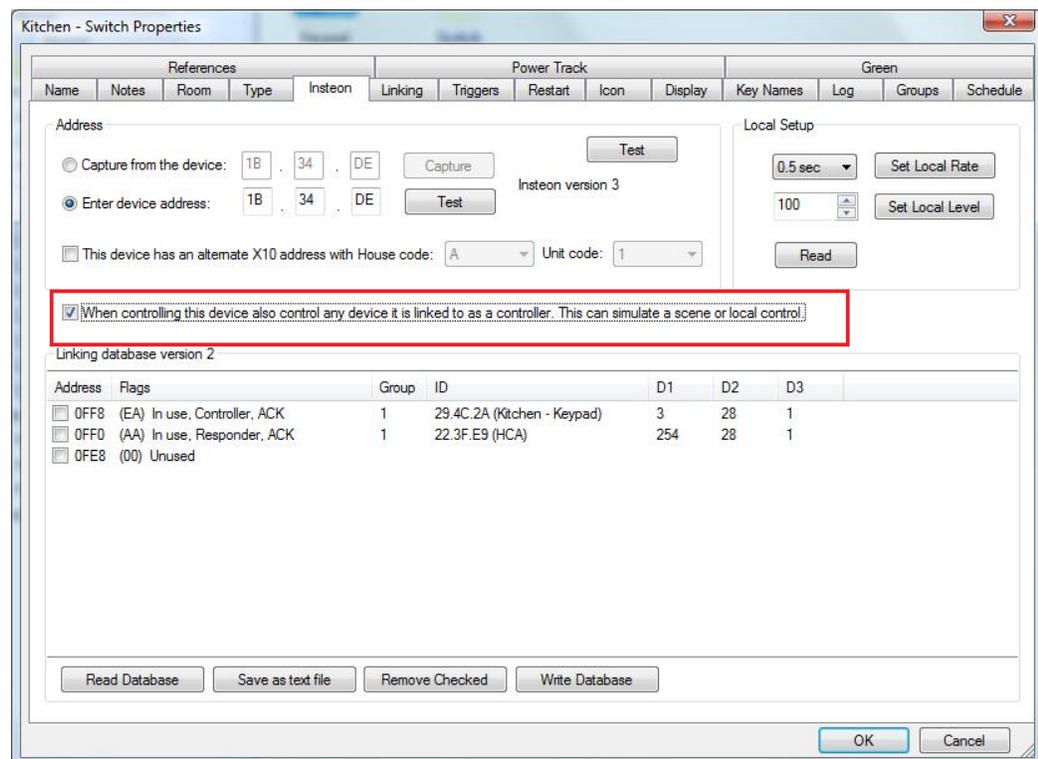
In the User Guide chapter on the Insteon Visual Scene Editor that tool was described along with how scenes can be invoked using the Scene Visual Programmer element and in schedules. There is an additional way to get "scene like" behavior.

One usual problem with Insteon devices because of its linking model is that no controller can pretend to be another controller. An example: Suppose you have a switch that controls a load linked as a controller of a KeypadLinc button LED. You tap on the paddle, the load comes on, and the KeypadLinc button turns on as well.

What happens if you use HCA to turn on the load? The load comes on but the KeypadLinc button does not. This is because HCA can't "pretend" to be the switch and have the KeypadLinc button LED respond.

There are three ways to solve this. One method is to create a scene where HCA was the controller and have the scene include the switch and the button LED. Another method is to create a program that controlled the switch and the button LED. In either case you have to remember to use the scene or program rather than control the switch directly.

The final method is to use an option that can be enabled on Insteon devices:



With this option enabled – and if HCA has the current linking table of the device – then when the device is controlled, HCA also controls any device that is linked to the device as a responder.

There is one major difference between what this option does and a "real" scene. With a scene all devices respond at the same time. They also respond using the ramp rate configured in the scene. When this option is enabled, HCA must send individual commands to each device and the ramp rates are not used. For keeping one device and an associated keypad indicator up to date this is an excellent method.



This dialog lets you choose any of the Insteon devices that transmit in your design.

The only other part that needs to be specified is the command to trigger on. You can create a trigger that responds to an On button press and another trigger that responds to an Off button press. Not all devices send all the commands listed so choose carefully.

**Hint:** Remember that triggers will not work unless the devices are linked to HCA. The trigger dialog presents all possibilities for triggers not just those linked. The Design Inspector, using the device linking tables, will look for any triggers that will not work as the necessary links are not in place.

---

## What HCA knows of the Insteon network

The Insteon network is expressed in the contents of the link databases in each device. For example, when you have linked a lamp module and a ControlLinc, when you press the linked button on the ControlLinc, the lamp comes on at a certain level and rate. That level and rate are established when you performed the linking.

If you have read all the linking databases of all your devices and you keep them up to date – that is, if you create links using the manual method you reread the network using the Network Capture dialog – then HCA can track the state changes of your devices fairly accurately. If the linking databases that HCA saves with your design file are different than what is in the devices then HCA can't keep its state correct.

Let's take an example: Suppose that you have a KeypadLinc linked to a SwitchLinc. You press a button on the KeypadLinc. The SwitchLinc comes on - but at what level? If HCA has the linking databases of the KeypadLinc and the SwitchLinc it can tell what level it went to and make sure that the HCA state is current with the SwitchLinc. Without the linking databases HCA could know that the SwitchLinc came on (due to the ACK message it sends) but not at what level.

Another case where the HCA state may get out of sync with Insteon devices is if you manually dim a device. Even if the device is linked to another device so that they both dim together, the Insteon commands don't provide enough information for HCA to know what level they are at when you stop dimming.

**Hint:** You can always request the status of an Insteon device to find out its actual state. This can be done from a Visual Program using the Test or GetStatus element. It can also be done from the user interface by right-clicking on the device and selecting 'Get Status' from the popup menu.

---

## Hints and Tips

### Multiple Interfaces

Don't forget that HCA can use multiple automation interfaces simultaneously. In this way you can use X10, wireless and IR devices along with your Insteon switches and keypads in your automation design. For example, a HCA program, started from an Insteon keypad trigger, can send commands to both UPB devices and X10 devices. Or a wireless motion sensor can be used to start a program that controls Insteon devices. In this way HCA can bridge all your automation technologies. You may want to look at the Protocol Bridge feature if you find you are using devices of different technologies that want to work together.

### Insteon PowerLinc Support for X10

While the Insteon PowerLine supports X10, if you have many legacy X10 devices, especially those that use the most primitive form of dimming, you may find that the Insteon PowerLine doesn't dim them as smoothly as you would like. This is due to the X10 implementation capabilities of the Insteon PowerLine. If this is a problem you could use an X10 interface like the CM15 for X10 sends and receives and use the Insteon PowerLine for Insteon messages only.

## Appendix 10

# Protocol Bridges

This appendix describes a feature in HCA that, with little setup, allows you to control devices that respond to one protocol with controllers that transmit a different protocol. For example, control an X10 device from a UPB keypad or an Insteon device with an X10 motion sensor. These topics are covered:

- What is a Protocol Bridge?
- Configuring a bridge
- Limitation of bridging

---

### What is a Protocol Bridge?

A Protocol Bridge is a feature in HCA designed to receive signals from a transmitting device and then retransmit them in the different protocol. For example, when HCA receives an ON command from an Insteon keypad, an X10 ON command can be transmitted to a specified house and unit code. This means that your Insteon keypad can be used to control an X10 light. Another user might be using an X10 motion sensor to turn on or off an UPB light.

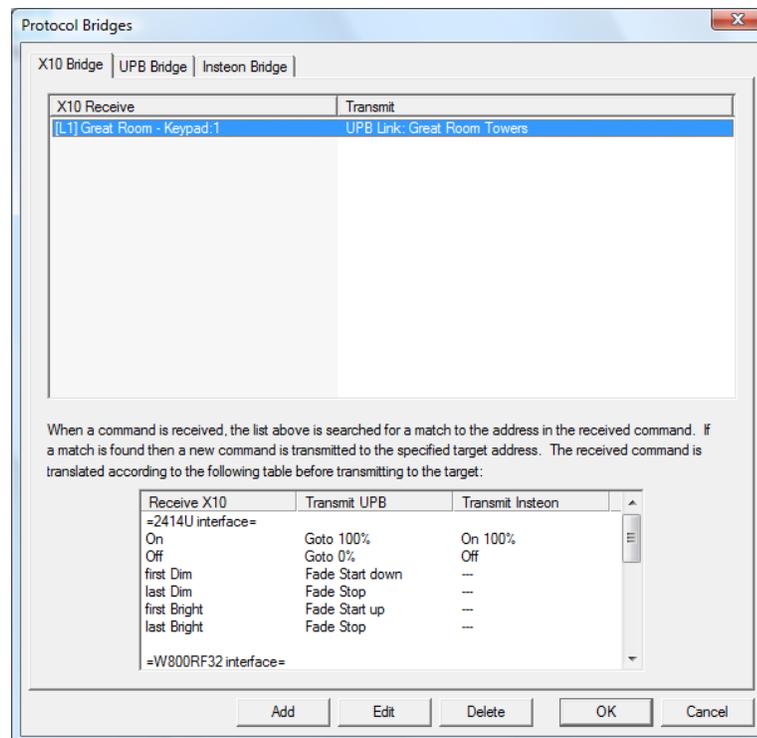
The information you provide on both sides of the bridge depends upon the protocol – X10, UPB, or Insteon.

Most of the functionality of a Protocol Bridge can also be done with an HCA program. For example, an HCA program can be started upon receipt of an Insteon ON command and the program can contain an ON element that sends to an X10 device. In the simple command translation case, configuring and maintaining a bridge is much simpler than an HCA program

The HCA bridge implementation is designed to act quickly. This means that as you press a key on a keypad you should see the effect of that key press very quickly on the target even though the keypad and target are not talking the same protocol. This is especially important when using a Dim or Bright buttons.

## Configuring a bridge

To configure protocol bridges press the *Bridge Setup* button in the ribbon *Protocols* category. This dialog appears:



As you can see on this dialog there is a separate tab for X10, Insteon and UPB. The top section shows the actions of the bridge you have configured. For example, in the above picture the bridge is configured so that when an L1 is received a UPB Link command is sent using the *Great Room Towers* link. What command is sent? This is where Protocol Bridges are smart. HCA does all the command translation for you!

The list at the bottom of the dialog shows the command translations. For example, an X10 ON command translates to a Goto 100% in UPB or an Insteon ON command with level 100.

Sometimes the translation depends upon the receiving or translating interface. What would this be? Some interfaces provide enough information to handle the “real time” aspect of a bridge and some don’t. An example may help explain this.

Suppose you press the ON button of an X10 keypad. The keypad is set up for housecode B and you press the button corresponding to a unit code of 2. That’s easy. HCA received the B2 ON and can easily translate that into an Insteon ON or UPB Goto. But now you press the Bright key and hold it down. What the keypad does is to send a stream of Bright commands on to the powerline. If HCA is given information each time one of those Bright commands appear on the powerline, HCA can send similar commands in the other protocols and the light should visibly get brighter.

However if the powerline interface saw the stream of bright commands, counted them, and when the button is released **then** HCA is informed, it wouldn’t be much good. Since HCA doesn’t know about the commands sent by the keypad until it’s too late. You couldn’t visually track the light level of the device and release the button when it matches what you want. And that is what makes the protocol bridges most useful.

Some interfaces report each Bright command and others just count and report at the end of the command stream. Because of this, it is a good idea to look at the translation table at the bottom of the dialog to see what the bridge can and can't do using your hardware.

**The bottom line on a bridge:** You specify the reception and transmission addresses and HCA handles the command translation.

The other items on this dialog are buttons to allow configuring a new item in the bridge, deleting an existing item, or modifying an item.

---

## X10 Bridge

An X10 bridge is configured as:

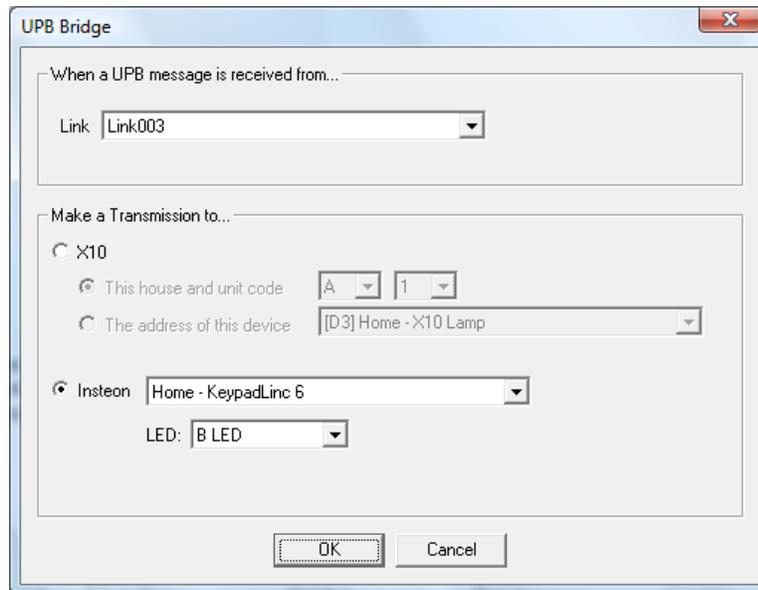
For the reception side of the bridge an X10 address is needed. This can be entered simply as a house and unit code or using the address of one of the devices already in your design.

The transmission side of the bridge can be UPB or Insteon. In the case of UPB you can transmit directly to a device or use a link. For Insteon all you need do is to name the target device.

---

## UPB Bridge

A UPB Bridge is configured as:



The screenshot shows a dialog box titled "UPB Bridge" with a close button (X) in the top right corner. The dialog is divided into two main sections. The first section, "When a UPB message is received from...", contains a "Link" dropdown menu with "Link003" selected. The second section, "Make a Transmission to...", has three radio button options. The "X10" option is selected. Under "X10", there are two sub-options: "This house and unit code" and "The address of this device". The "This house and unit code" option is selected, with "A" in a dropdown and "1" in a text field. The "The address of this device" option is unselected, with "[D3] Home - X10 Lamp" in a dropdown. Below these, the "Insteon" option is unselected, with "Home - KeypadLinc 6" in a dropdown. Under "Insteon", there is an "LED:" label and a dropdown menu with "B LED" selected. At the bottom of the dialog are "OK" and "Cancel" buttons.

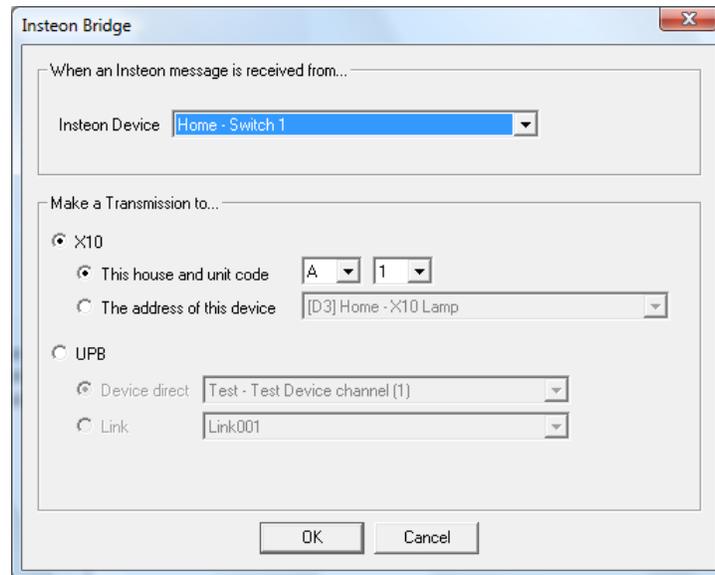
The receive side of the bridge is always configured as a link since all UPB device transmissions are by a link.

For an X10 transmission an address is needed. This is entered as a house and unit code or as the address of a device already in your design. For an Insteon transmission a device is selected.

---

## Insteon Bridge

An Insteon Bridge is configured as:



For the receive side of the bridge, select an Insteon device and if the device has more than one button, select a button. In this example an Insteon KeypadLinc controller is selected and the bridge specifies what happens when the C button is pressed.

For the X10 transmission side of the bridge an X10 address is needed. This can be entered as a house and unit code or as the address of a device already in your design.

For a UPB transmission, you must select if the transmission is to be direct to a UPB device or a link transmission.

---

## Limitation of Bridges

Most of the limitations of bridges come from the interfaces in use. Some reasons for this are described above in the section on configuring a bridge.

Another limitation comes when one interface is used to transmit on more than one protocol. For example, the Insteon 2414U interface is often used for X10 and Insteon transmissions. Because of this, it doesn't work well to bridge X10 and Insteon especially when X10 DIM and BRIGHT commands are on the powerline. Having the X10 commands reported to the computer quick enough at the same time Insteon commands are being sent just doesn't work well enough to be useful.

If you look at the command translation table on the X10 bridge tab you will see that the protocol bridge does much better when bridging X10 to Insteon when, for example, X10 receptions are coming from a whole house wireless receiver that when they come from the 2414U interface.

If your need for simple translations fit inside the limitations of what an HCA Protocol Bridge can do, then this can be a quick way to control newer devices with your older keypads or motion sensors.



## Appendix 11

### Z-Wave

This appendix describes support for Z-Wave interfaces and devices. Included are these sections:

- What is Z-Wave?
- Building a Z-Wave network
- Configuring the Leviton Z-Wave Interface
- Z-Wave Devices
- Z-Wave Visual Program Element
- Z-Wave Triggers

#### Before you begin with Z-Wave!

Z-Wave is a bit like Linux – lots of capability with little consistency between providers or documentation. Z-Wave can do a lot if you can just find the right piece of the puzzle and know where to put it.

Z-Wave is very different from other HCA supported protocols in that HCA provides only a method to send and receive commands from the interface. It does not attempt to encode or decode those commands. Because of this, HCA technical support can offer little beyond what is in this User Guide Appendix and the Z-Wave technical notes available on the support web site. Beyond this you are on your own.

---

### What is Z-Wave?

Z-Wave is a proprietary wireless mesh networking technology originally developed by the Danish company Zensys, now part of Sigma Designs. Because Sigma Designs considers documentation about Z-Wave technology proprietary, and makes such documentation only available under license, it can be a bit challenging to figure out how it works and how to control Z-Wave devices.

Unlike HCA support for UPB, Insteon, X10 and Wireless devices, there is very little built-in to HCA for Z-Wave. There is support for very simple one-way *on-off-dim* and *on-off* devices but that's all. All other device types you may want to add to your design – thermostats, two-way devices, door locks, etc – all have to have their protocol determined and then that protocol implemented in HCA programs you create. This is much different than other supported protocols!

Fortunately, there is a good deal of Z-Wave information on the Internet. A recommendation is to search for protocol and command information on your devices. On the HCA web site is a technical note that discusses using the HCA Z-Wave support for Z-Wave door locks.

---

### Building a Z-Wave network

While there are several Z-Wave interfaces available, HCA supports only the *Leviton Vizia RF+ serial to Z-Wave* interface. This interface is model number VRC0P.

Just as a UPB network is initially configured using a tool external to HCA (UPStart), a Z-Wave network is configured with a Windows program. This software, available for download from Leviton is called the *Vizia RF+ Installer Tool*. To use this program you need also a LEVVRUSB-1US USB stick for configuration.

This USB stick will be the “primary” controller in your network. The Leviton VRC0P will function as a “secondary” controller, in Z-Wave parlance, even though it will be the way HCA controls your Z-Wave devices.

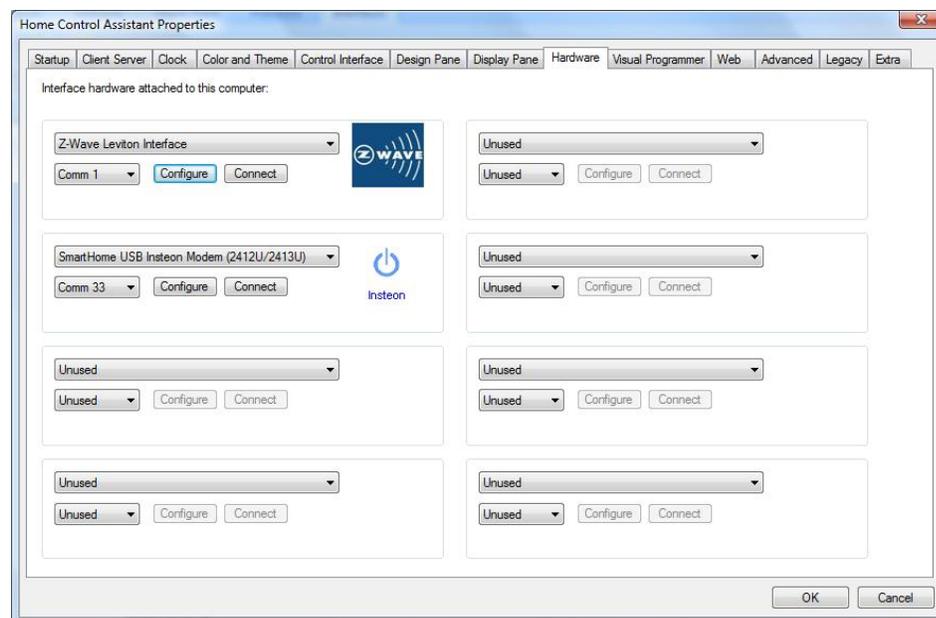
Instructions for creating a Z-Wave network using the Leviton USB stick and installation software can be found in the Help file included with the *Vizia RF+ Installer Tool* and on the Leviton support web site.

**Tip:** When you do the final network configuration with the Leviton tool, all Z-Wave devices need to be in their final location. Moving Z-Wave devices affects the network topology, and can adversely affect network routing if the network is not re-optimized.

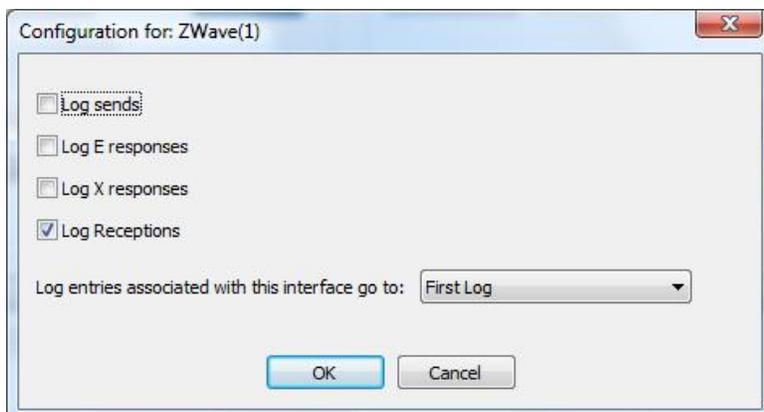
---

## Configuring the Leviton Z-Wave interface

The Leviton Z-Wave interface is identified to HCA like all interfaces from the hardware setup tab of *HCA Options*. The only piece of information needed is the serial port it uses.



The *configure* button sets the logging options:



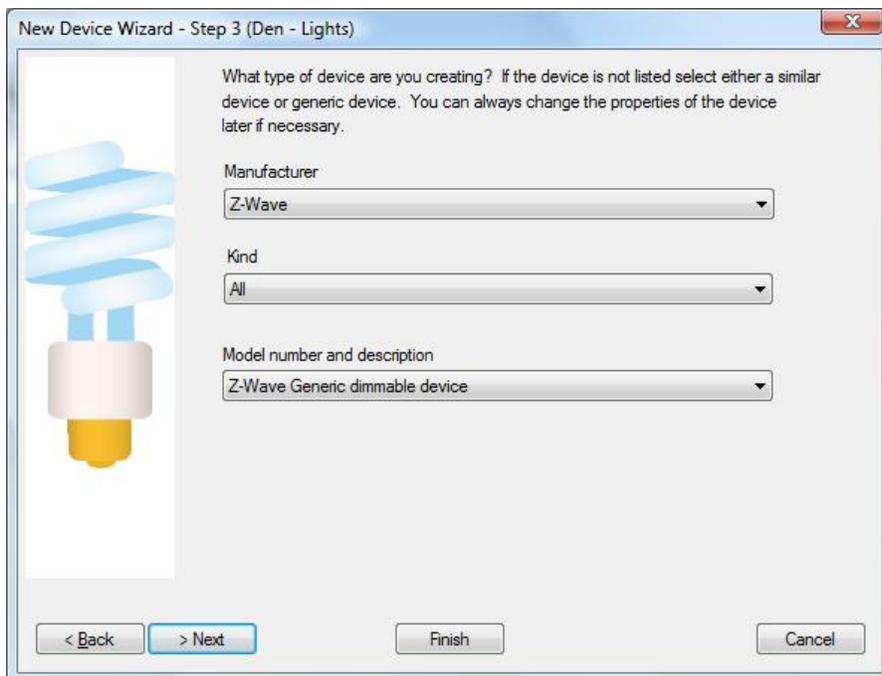
As described in the Leviton application note, each message sent to the interface usually generates an “E” response with an error code, followed by a “X” response that shows the transmission status. You can choose to log these or not.

**Tip:** The Leviton application note that describes the Leviton Z-Wave protocol is available on the HCA support web site in the technical notes section.

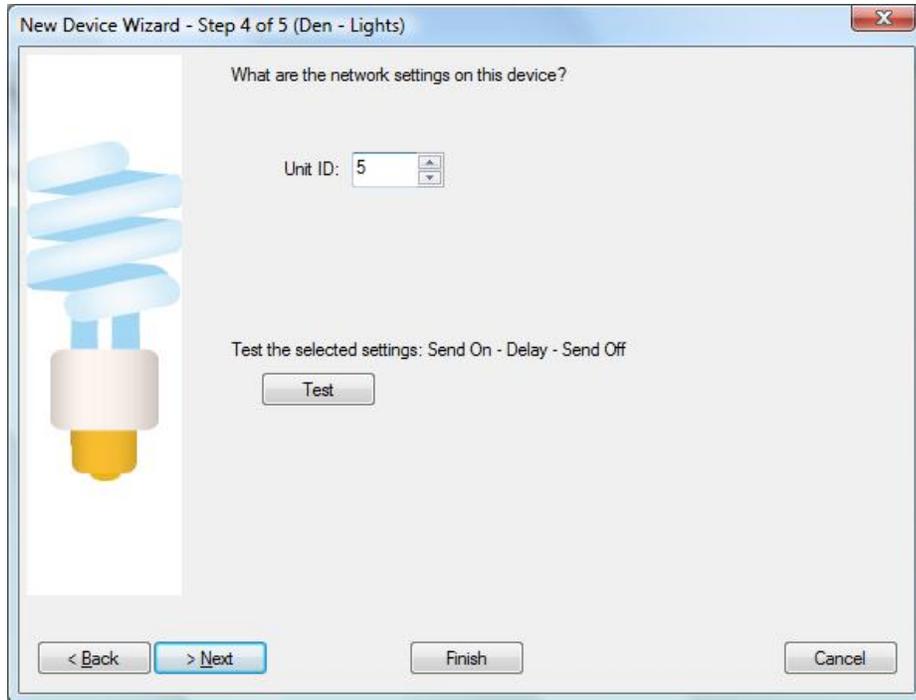
---

## Z-Wave devices

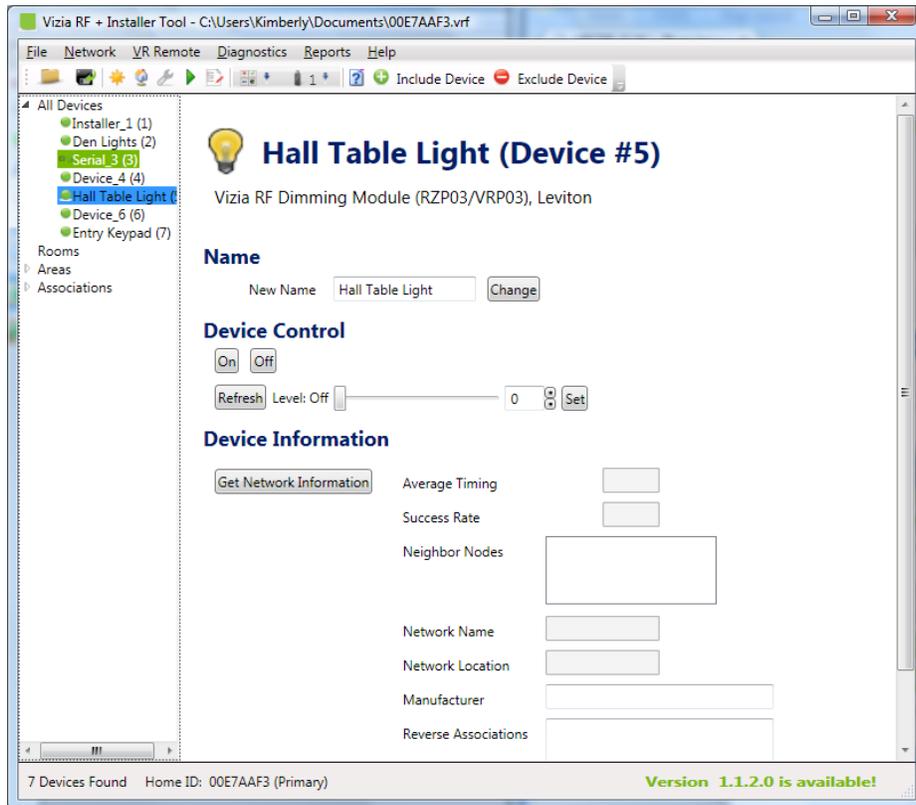
HCA directly supports only simple one-way *on-off-dim* and *on-off* devices. To use these devices select as their type *Z-Wave Generic dimmable device* or *Z-Wave generic non-dimmable device*.



In the next step of the wizard the unit id of the Z-Wave device is set:

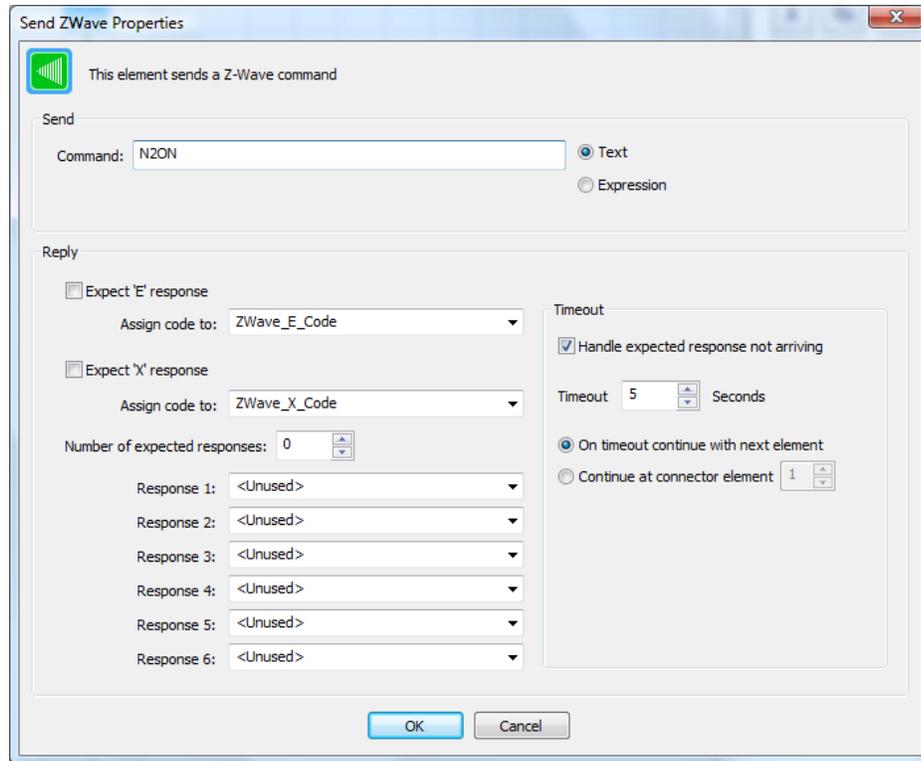


The unit id you enter into HCA comes from the Leviton setup tool:



## Z-Wave Visual Program Element

The real power of Z-Wave is not in the simple devices described in the above section but rather in the devices that you can implement. The key piece is the Z-Wave Visual Programmer element.



Enter the text to send into the *Send Text* box. This can be the actual text or the result of an expression evaluation if the *Expression* option is used.

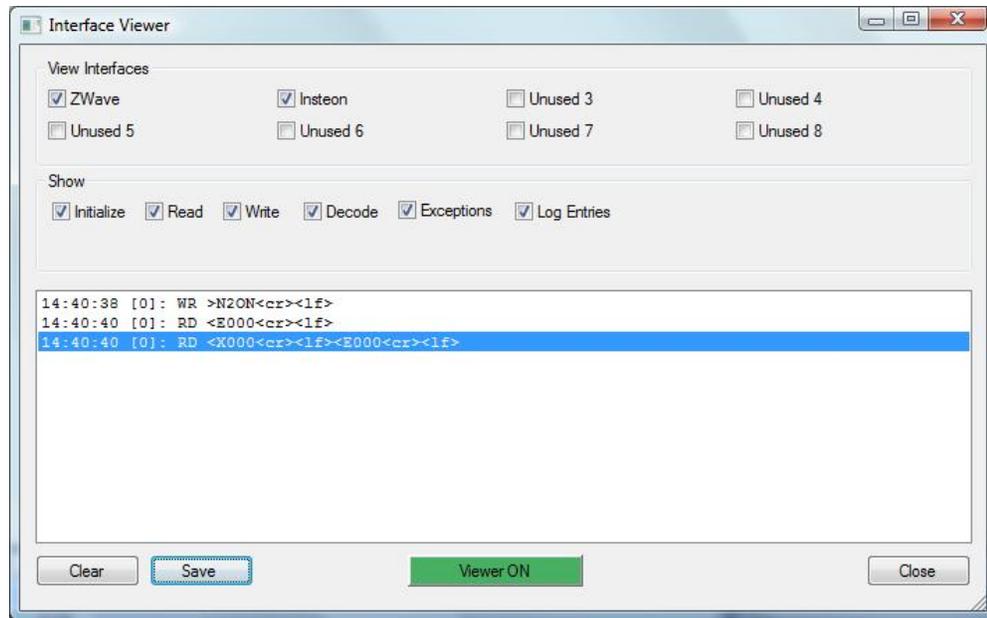
Depending upon the command you can expect “E” and “X” responses and if so tick those boxes and those messages are specifically decoded and the values assigned to the selected flags. If the command generates additional responses, enter the number of expected responses and then, for each response, select the flag that reception is assigned to. You can select an existing flag or type in the name of a new flag that is created.

## Interface Viewer

What does a very simple Z-Wave command look like? You can use the *Interface Viewer*.

Open the *Interface Viewer* from the *Interfaces* ribbon category. This viewer shows the messages sent to and received from one or more interfaces. This gives you a window into the communications at the lowest level.

In the image below, an ON command was sent to unit number 2. The interface replied back with an “E” response with code 0 (success) and an “X” response with code 0 (success).



The names of the eight interfaces that can be configured are the first set of checkboxes. Tick the ones that you want to see data from. Interfaces not configured are listed as *Unused*.

The *Show* box determines what sort of information you want to view. Tick or clear the boxes for the type of data you want to see.

You can turn the viewer on or off with the button at the bottom of the dialog. When green the viewer is enabled and when red it isn't.

In the dialog list, messages received from the interface are prefixed with RD, messages sent to the interface are prefixed with WR, and if the message is decoded – messages from some interfaces are decoded but Z-Wave messages aren't – they are prefixed with DC. Enclosed in []'s is the number of the interface the line is for. The first interface is numbered 0 and the last is numbered 7.

This interface is at a very low level so you may see, because of the nature of network and serial reads, a single reception broken across multiple RD lines.

You can clear the list with the *Clear* button and save the list into a text file with the *Save* button.

## Triggers

Messages received from the Z-Wave interface can be used as a trigger for a program. To create a trigger of this type, select as the trigger type *Zwave Reception*.

There are several parts to a *ZWave Reception* trigger and these are covered in the next sections.

## Pattern

Unlike, for example, Insteon triggers where the reception from an Insteon interface is decoded into a source device and command, and Insteon triggers specify that source and command, the *ZWave Reception* trigger is based only on the text of the reception itself.

A *ZWave Reception* trigger contains a pattern that is matched against the received text. If the pattern matches the received text then the program is triggered. If the reception doesn't match the pattern then the program is not triggered.

The pattern is specified as a *Regular Expression*. A regular expression is a sequence of characters that forms a search pattern. The full description of a regular expression is beyond the scope of this User Guide but here are the fundamentals.

A regular expression is composed of a series of characters and metacharacters. Characters that are not metacharacters match the corresponding changes in the text. Meta characters describe one or more possible characters to match in the text. The metacharacters are:

Character	Use
.	(dot) Matches any single character. For example, a.c matches "abc", or "axc" or "a6c" etc.
[ ]	A bracket expression. Matches a single character that is contained within the brackets. For example, [abc] matches "a", "b", or "c". [a-z] specifies a range which matches any

	lowercase letter from "a" to "z". These forms can be mixed: [abcx-z] matches "a", "b", "c", "x", "y", or "z", as does [a-cx-z].
[^ ]	Matches a single character that is not contained within the brackets. For example, [^abc] matches any character other than "a", "b", or "c". [^a-z] matches any single character that is not a lowercase letter from "a" to "z".
^	Matches the starting position within the string
\$	Matches the ending position of the string
*	Matches the preceding element zero or more times. For example, ab*c matches "ac", "abc", "abbbc", etc. [xyz]* matches "", "x", "y", "z", "zx", "zyx", "xyzy", and so on.
{m, n}	Matches the preceding element at least m and not more than n times. For example, a{3,5} matches only "aaa", "aaaa", and "aaaaa". This is not found in a few older instances of regular expressions
\	The following character is a character and not a metacharacter

Here are some examples:

- `.*` matches any text.
- `.at` matches any three-character string ending with "at", including "hat", "cat", and "bat".
- `[hc]at` matches "hat" and "cat".
- `[^b]at` matches all strings matched by `.at` except "bat".
- `[^hc]at` matches all strings matched by `.at` other than "hat" and "cat".
- `^[hc]at` matches "hat" and "cat", but only at the beginning of the string or line.
- `[hc]at$` matches "hat" and "cat", but only at the end of the string or line.
- `\[.\]` matches any single character surrounded by "[" and "]" since the brackets are escaped, for example: "[a]" and "[b]".

Since it can be complex to compose a regular expression, in the dialog is a *Validate* button that checks the syntax of the regular expression. You must use the *Validate* button to check the pattern before creating the trigger.

However, just because the syntax is correct that doesn't mean that the pattern matches strings as you expect. Only testing shows that.

**Tip:** Use your favorite search engine or Wikipedia to find the full syntax of regular expressions.

---

## Flag assignment

ZWave Reception triggers are also different than other triggers in an additional aspect. The text of the reception can be assigned to a flag for subsequent decoding in the triggered program.

Select the name of an existing flag or type in the name of a new flag. If the program is triggered – when the pattern matches the reception – the received text is assigned to the flag before the program starts. This lets you do further decoding of the reception in the program that is started.

This is an optional feature of the trigger. If you don't want to use this, set the flag as *<Unused>*

---

## Trigger of last resort

You can also create a trigger that starts a program if no trigger on any program matches the Z-Wave reception. Of course, only one program can be designated with this trigger.

To enable this option tick the *Trigger on a reception not otherwise handled* checkbox.

Trigger on a reception not otherwise handled



## Appendix 12

### Generic Serial, Generic IP, and Generic Server

This appendix describes support for interfaces that implement a protocol and action that is not directly supported by HCA.

*Generic interfaces* are supported only to the extent that HCA can send to them and receive from them and, optionally, decode them only to the extent of breaking receptions into individual messages. Data sent must be assembled by your own implementation in one or more Visual Programs. Data received must be processed by your own implementation in one or more Visual Programs.

Generic serial and IP (network) interfaces are configured and used in approximately the same manner. Because so much is common between them both are covered in this appendix.

Not all interfaces that are attached to the computer using a serial port – real or virtual – are a *Generic Serial Interface* as covered in this appendix. For example, the SmartHome Serial Insteon interface isn't a *Generic Serial Interface* because HCA includes an implementation of its protocol.

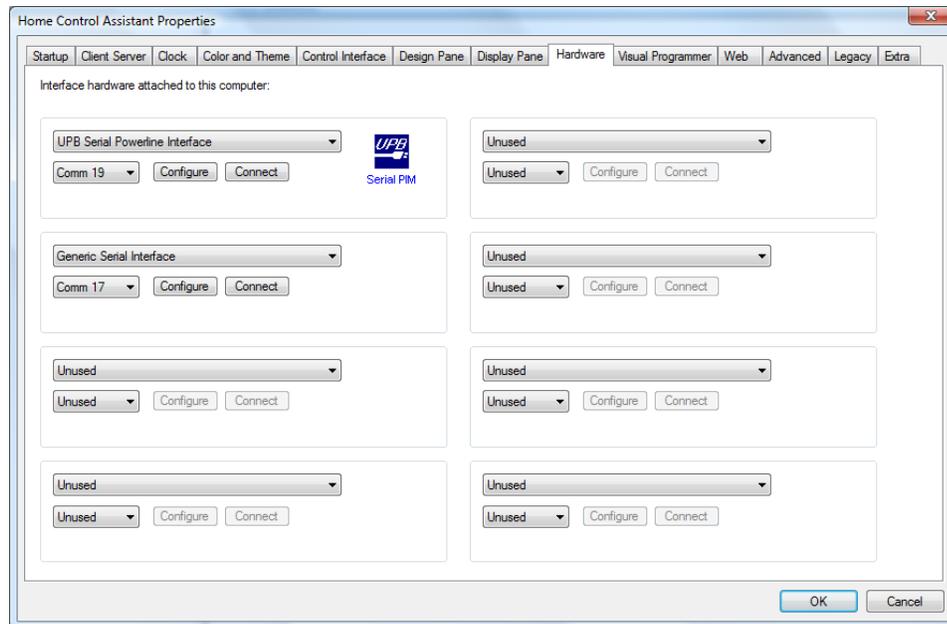
Not all interfaces that are accessed using a network connection are Generic IP Interfaces as covered in this appendix. For example, the Global Cache IR Interface has a network address and communication is made using TCP but it isn't a *Generic IP Interface* because HCA includes an implementation of its protocol.

This appendix covers these topics:

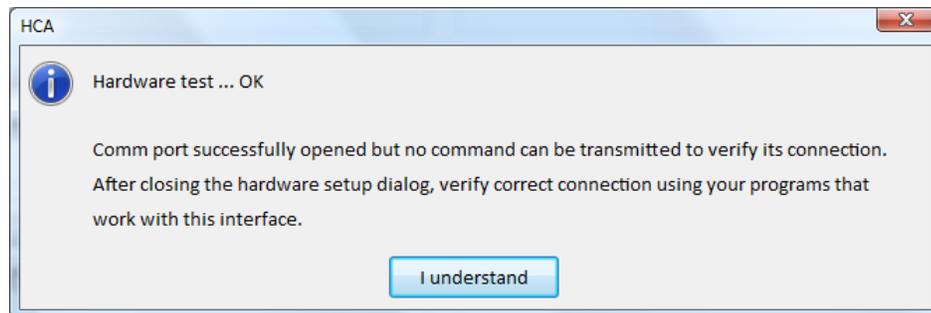
- Configuring a generic serial interface
- Configuring generic IP interfaces and Generic Server interfaces
  - Client interfaces
  - Server interfaces
- Visual program Port I/O element
- Port Reception Triggers
- Working with Generic Interfaces

## Configuring a serial interface

A generic serial interface is configured from *HCA Options* on the *Interfaces* tab.

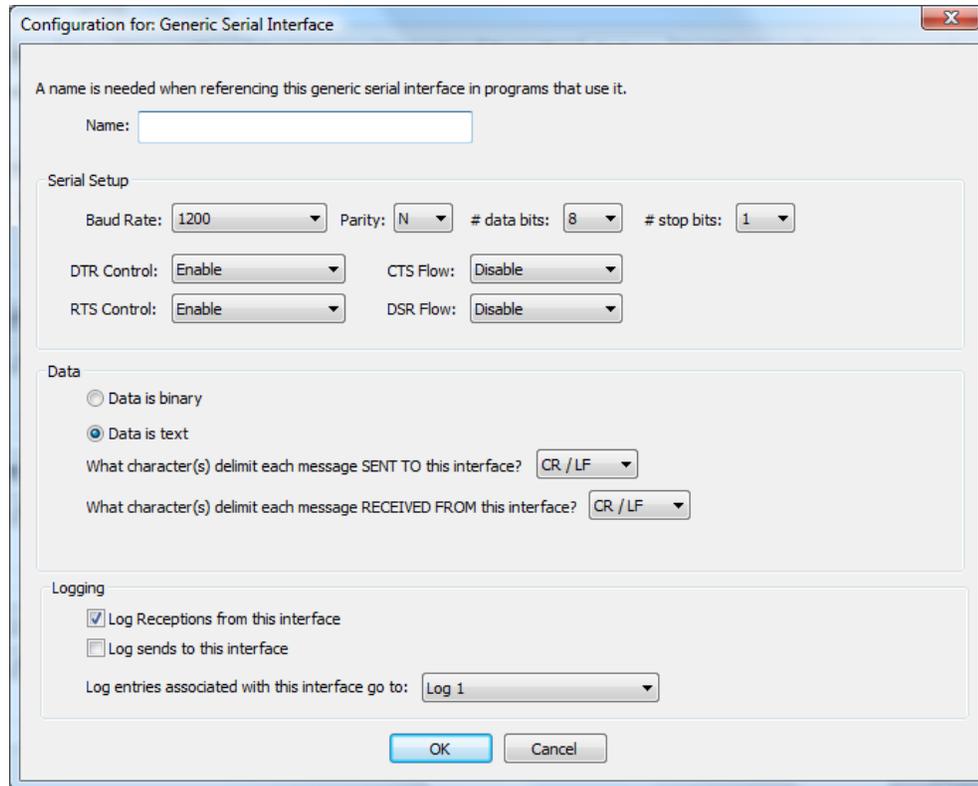


Choose “Generic Serial Interface” as the interface type, select the serial port to use, and press the *Connect* button. If the serial port can be opened a popup message appears:



As the dialog says, the only test performed is that the serial port can be opened. There doesn't even have to be an actual interface connected to that serial port for the test to pass.

Serial communication has a number of settings: Baud rate, flow control, etc. These settings for this interface can be configured. Press the *Configure* button to open the settings dialog.



Since you can have more than one serial interface in use simultaneously, by giving each a name you can use that name to identify which interface you want to work with in a program.

The sections of this dialog are:

### Serial Setup

Baud rate, parity, # of data bits, # of stops bits

These are the standard serial port settings and they depend upon the interface you are using.

DTR Control, RTS Control, CTS Flow, DSR Flow

These are the serial flow control settings and the choices depending upon the connected interface.

### Text Data

Each message sent to and from the serial interface is decoded to a very minimal degree. This decoding needs to know how messages are separated from each other. In this way when you use the Visual Program *Port IO* element to receive a message, the decode mechanism knows when a message has been completely received. It is completely received when the message terminator(s) are received.

When a message is transmitted a terminator can be added. The *Sent To* terminator can be the same or different for received messages.

The choices for the send and receive terminators are:

- CR – ascii hex 0d
- LF – ascii hex 0a
- CR followed by LF
- LF followed by CR

- None
- Other

In the case of the CR-LF and LF-CR then two characters are the message terminator. In the case of *Other* then you can enter the character to use as the terminator. Entry is made as a hex number of the ascii character code so non-printable characters can be used.

If *None* is used as the *Sent To* terminator then no terminator characters are automatically added to each message sent. With the *Sent To* terminator set to *none* then programs that use the Port IO element must include the necessary terminating characters in each send.

If *None* is used as the receive terminator then no decoding of receptions into separate messages happens. With the receive terminator set to *None*, whenever the *Port IO* element configured for a receive executes, then anything currently in the receiver buffer is what is read. If the buffer is empty then the element immediately times out.

### Binary Data

Each message sent to and from the serial interface is decoded to a very minimal degree. This decoding needs to know how messages are separated from each other. In this way when you use the Visual Program *Port IO* element to receive a message, the decode mechanism knows when a message has been completely received. Unlike text data, binary data doesn't have a terminating character. HCA determines when a complete message is received based upon your selection in the message type dropdown. There are several choices and which you use depends upon the hardware being interfaced with. As an example, suppose the data being sent was the 3 bytes 0x40 0x00 0xef. These are the options and how the message data is expected.

- Fixed length  
0x40 0x00 0xef
- Byte length prefix. Record length not including length byte  
0x03 0x40 0x00 0xef
- Byte length prefix. Record length including the length byte  
0x04 0x40 0x00 0xef
- Word (high – low) length prefix. Record length not including the length word  
0x00 0x03 0x40 0x00 0xef
- Word (low –high) length prefix. Record length not including the length word  
0x03 0x00 0x40 0x00 0xef
- Word (high –low) length prefix. Record length including the length word  
0x00 0x04 0x40 0x00 0xef
- Word (low - high) length prefix. Record length not including the length word  
0x04 0x00 0x40 0x00 0xef

When programs receive the data using the Port I/O element the data is in a text string. Continuing the above example the data would be in the string "4000ef". Data is supplied in the same manner when sending data to the serial interface.

If the "binary send formatted same as reception type" option is enabled, then if one of the length prefix options is selected, the length prefix byte(s) are prefixed to the data being sent.

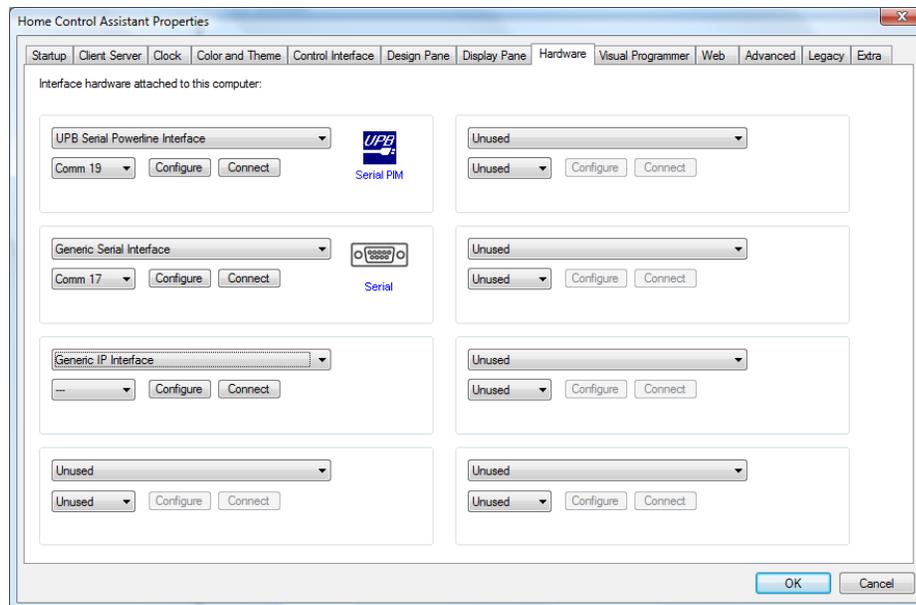
### Logging

As with all interfaces you can select what is logged and the log that is used. Select the log to use and if sends or receives or both are logged.

The checkbox to configure for printable characters is used in logging of the sends and receives. If the data is printable characters then it can be logged as text otherwise a hex representation of the received bytes is used.

## Configuring an IP interface

An IP interface is connected from *HCA Options* on the *Interfaces* tab.



Like the serial interface, before you can test the connection it is necessary to configure it. Press the *Configure* button.

In this dialog you configure both the connection and the format of the data being transmitted. The data type – binary or text – and all the options for them are the same as in the generic serial interface.

Since you can have more than one IP interface in use simultaneously, by giving each a name, you can use that name to identify which interface you want to send to or receive from in a program.

There are two ways to configure a Generic IP Interface: As a client or server. The next section explains the differences.

---

## Client or Server

Before discussing the details of the connection it is necessary to determine if an interface is used for a client connection or as a server.

When HCA opens a network interface, HCA can operate as a *client* of that interface or HCA can operate as a *server*.

When creating a connection to an interface as a client, what you are saying is that there is some physical interface someplace on the network. Using HCA programs you send messages to this interface and receive responses back.

When configuring an interface as a *server* what you are saying is that HCA **is** the interface. Other programs (Windows programs, browsers, mobile applications) open the IP address of the computer that HCA is running on with the port number specified in the interface configuration. Those programs then send requests to HCA and receive responses back. HCA programs that you create receive those messages, process them, and reply with responses.

Why would you want to configure an interface to act as a server? Using a server you can provide an interface for mobile apps, windows programs, and browser addins to send messages into HCA for action. What are those messages? That’s up to you! You get to design whatever messages are needed to accomplish whatever your task is.

---

## Configuring an IP Interface

The parameters to configure an IP interface:

### Address

When configuring the interface as a client then you need to specify the IP address of the device you are communicating with. That interface is located by a 4-part network address given in the form A.B.C.D. You can supply the address in that format or, if a DNS entry exists for the interface, then you can use its name. Upon connection, a lookup is done on the name to translate it into the 4-part network address.

In addition to a network address, a port number is needed as a network interface may open different ports for different purposes.

### Timeouts

In any communication you always have to handle the case where the communication fails. The *During connect allow* timeout setting is used during connection to the interface. If after the given number of seconds a connection is not established, then HCA gives up and shows an error.

### Auto Disconnect

Some interfaces allow only a single connection at a time. This can be a problem if HCA opens the connection and leaves it open forever. Doing that may prevent other users of that interface from connecting to it.

If the *Disconnect* option is enabled, then HCA connects to the interface when it needs to use it, and then leaves it open for the specified number of minutes and if not used again during that time then it disconnects. The next time it needs to use the interface then it reconnects.

If you use this option then you will not receive *Asynchronous* messages from the interface. Here is what that means: Suppose you are connecting to an interface that controls a hard-wired lighting system. You can send commands to it to turn on and off the lights. You send a message saying “Lights on” and it replies back with “OK”. That a *synchronous* message protocol. You send something and it replies. The message from the interface (its reply) is always preceded by a message sent to it (your request).

But also suppose that the interface can notice when someone pushes a button to turn the lights on. It could send to HCA a message to report what occurred. This would be an asynchronous message. You didn’t ask it to do anything; it just sends you a message. The message you received (the report) isn’t preceded by a message you sent to it. That is why this is an *asynchronous* message.

If the interface is not connected when the interface has something to report in this way, it will not get received by HCA since HCA doesn’t have an open connection to it.

If you want to receive these kinds of messages – reports for actions that HCA didn't initiate – then you can't use the disconnect option.

## Telnet

If the interface you are using operates as a Telnet server and if you want to connect to it that way, then tick the *Use Telnet protocol* checkbox and enter the user name and password needed. Each time HCA connects to the interface it automatically performs a telnet login using that user name and password.

**Tip:** Make sure you use the appropriate port. Telnet is often on port 23.

---

## Configuring a Generic Server

To create a generic server, on the hardware connection tab of HCA Options, select *Generic Server* and then press the *Configure* button.

The main parameter is the port number. This port number must **not** be the port number the HCA Server uses and HCA clients – and mobile applications – connect to.

Like the Generic IP Interface and the Generic Serial interface, you can specify the data and how the data is handled – binary or text. If you are making the connection to the generic server using a WebSocket then the WebSocket protocol frames and un-frames the messages so the delimiters are not needed.

**Note:** There is a technical note more completely describing the use of a Generic Server with an example. Look for the *User Applications* note in the technical notes area of the support web site.

## Send and Receive

As described in the introduction to this appendix, HCA manages sending to and receiving to the interface. But the contents of the data sent and received is opaque to HCA – all decoding has to be done in programs you create.

Sending and receiving from the serial or IP interfaces uses the Visual Program *Port I/O* element. This element, depending upon its configuration, can perform three different actions: Send, Receive, or Send and then Receive.

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

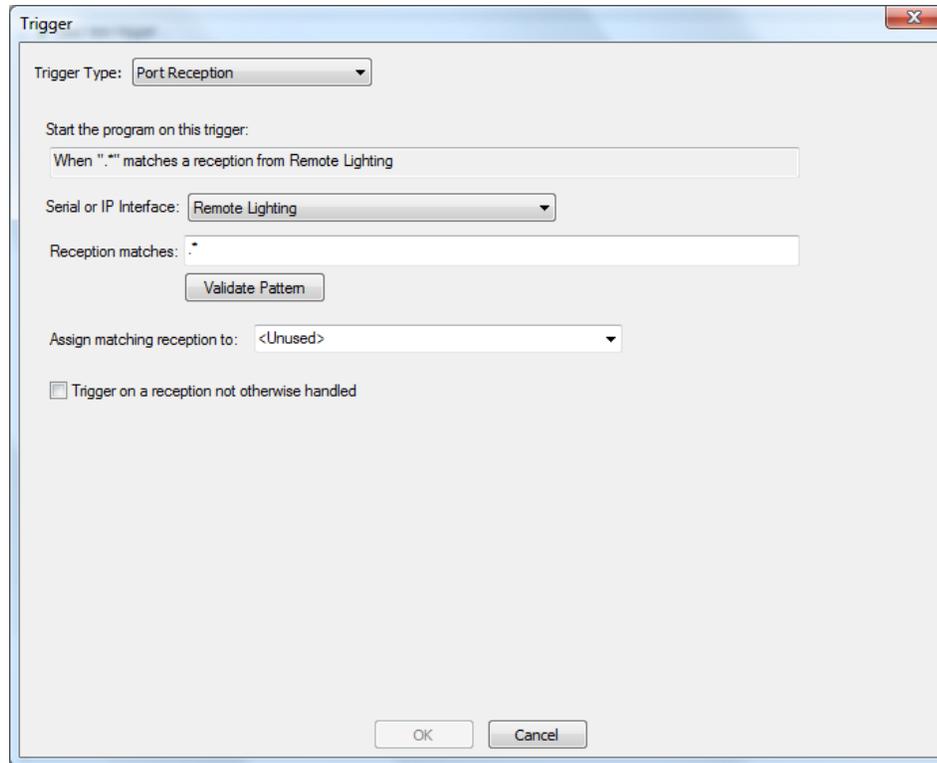
If sending enter the text to send into the *Send Text* box. This can be the actual text or the result of an expression evaluation if the *Expression* option is used.

If receiving enter the number of expected responses and then, for each response, select the flag that reception is assigned to. You can select an existing flag or type in the name of a new flag that is created.

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 maximum 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 connector element with the selected number.

## Triggers

Asynchronous messages received from an interface can be used as a trigger for a program. To create a trigger of this type, select as the trigger type *Port Reception*



There are several parts to a Port Reception trigger and these are covered in the next sections.

As you review the next sections, remember that if you are working with binary data the text is rendered into characters. For example, a reception of the bytes 0x64 0x00 0xef assigns the string "6400ef" to the chosen variable.

---

## Pattern

Unlike, for example, Insteon triggers where the reception from an Insteon interface is decoded into a source device and command, and Insteon triggers specify that source and command, the Port Reception trigger is based only on the text of the reception itself.

A Port Reception trigger contains a pattern that is matched against the received text. If the pattern matches the received text then the program is triggered. If the reception doesn't match the pattern then the program is not triggered.

The pattern is specified as a *Regular Expression*. A regular expression is a sequence of characters that forms a search pattern. The full description of a regular expression is beyond the scope of this User Guide but here are the fundamentals.

A regular expression is composed of a series of characters and metacharacters. Characters that are not metacharacters match the corresponding changes in the text. Meta characters describe one or more possible characters to match in the text. The meta characters are:

Character	Use
.	(dot) Matches any single character. For example, a.c matches "abc", or "axc" or "a6c" etc.
[ ]	A bracket expression. Matches a single character that is contained within the brackets. For example, [abc] matches "a", "b", or "c". [a-z] specifies a range which matches any lowercase letter from "a" to "z". These forms can be mixed: [abcx-z] matches "a", "b", "c", "x", "y", or "z", as does [a-cx-z].

[^ ]	Matches a single character that is not contained within the brackets. For example, [^abc] matches any character other than "a", "b", or "c". [^a-z] matches any single character that is not a lowercase letter from "a" to "z".
^	Matches the starting position within the string
\$	Matches the ending position of the string
*	Matches the preceding element zero or more times. For example, ab*c matches "ac", "abc", "abbc", etc. [xyz]* matches "", "x", "y", "z", "zx", "zyx", "xyzzy", and so on.
{m, n}	Matches the preceding element at least m and not more than n times. For example, a{3,5} matches only "aaa", "aaaa", and "aaaaa". This is not found in a few older instances of regular expressions
\	The following character is a character and not a metacharacter

Here are some examples:

- .\* matches any text
- .at matches any three-character string ending with "at", including "hat", "cat", and "bat".
- [hc]at matches "hat" and "cat".
- [^b]at matches all strings matched by .at except "bat".
- [^hc]at matches all strings matched by .at other than "hat" and "cat".
- ^[hc]at matches "hat" and "cat", but only at the beginning of the string or line.
- [hc]at\$ matches "hat" and "cat", but only at the end of the string or line.
- \[. \] matches any single character surrounded by "[" and "]" since the brackets are escaped, for example: "[a]" and "[b]".

Since it can be complex to compose a regular expression, in the dialog is a *Validate* button that checks the syntax of the regular expression. You must use the *Validate* button to check the pattern before creating the trigger.

However, just because the syntax is correct that doesn't mean that the pattern matches strings as you expect. Only testing shows that.

**Tip:** Use your favorite search engine or Wikipedia to find the full syntax of regular expressions.

---

## Flag assignment

Port Reception triggers are also different than other triggers in an additional aspect. The text of the reception can be assigned to a flag for subsequent decoding in the triggered program.

Select the name of an existing flag or type in the name of a new flag. If the program is triggered – when the pattern matches the reception – the received text is assigned to the flag before the program starts. This lets you do further decoding of the reception in the program that is started.

This is an optional feature of the trigger. If you don't want to use this, set the flag as *<Unused>*

---

## Trigger of last resort

You can also create a trigger that starts a program if no trigger on any program matches the reception. Of course, only one program can be designated with this trigger.

To enable this option tick the *Trigger on a reception not otherwise handled* checkbox.

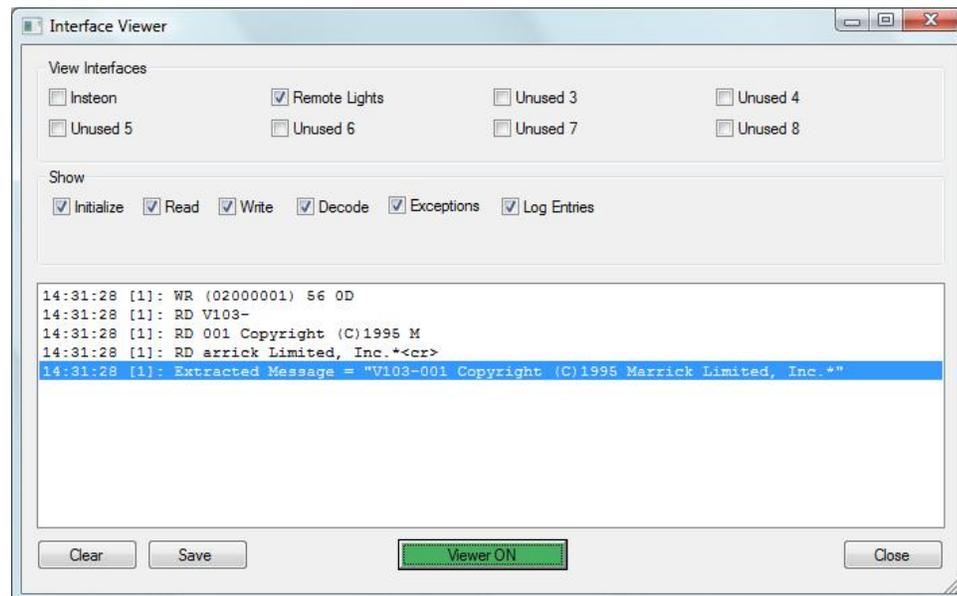
Trigger on a reception not otherwise handled

---

## Working with Serial and IP Interfaces

It can be complex working with these Generic interfaces since all the encoding of messages you send and decoding of messages received is implemented in your programs.

Open the *Interface Viewer* from the *Interfaces* ribbon category. This viewer shows the messages sent to and received from one or more interfaces. This gives you a window into the communications at the lowest level.



The names of the eight interfaces that can be configured are the first set of checkboxes. Tick the ones that you want to see data from. Interfaces not configured are listed as *Unused*.

The *Show* box determines what sort of information you want to view. Tick or clear the boxes for the type of data you want to see.

You can turn the viewer on or off with the button at the bottom of the dialog. When green the viewer is enabled and when red it isn't.

In the list messages received from the interface are prefixed with RD, message sent to the interface are prefixed with WR, and if the message is decoded – messages from some interfaces are decoded but messages from the Generic Interfaces are not – they are prefixed with DC. Enclosed in []'s is the number of the interface the line is for. The first interface is numbered 0 and the last is numbered 7.

This interface is at a very low level so you may see, because of the nature of network and serial reads, a single reception broken across in multiple RD lines.

You can clear the list with the *Clear* button and save the list into a text file with the *Save* button.

