HCA Tech Note 101

Using the HCA Server as a bridge from Alexa to the world

You can use the HCA Server to control all the devices and programs that HCA supports – Insteon, UPB, X10, IR, etc. But you can also use the HCA Server as a bridge between Alexa and a device that Alexa doesn't talk to and that HCA doesn't have built-in support for. This technical note described the process.

If you have something that you want to control by Alexa, there are several questions: Can you communicate with it over a serial port or an IP connection and do you know what those commands look like? Or, can you control it using a Windows program that you have written?

If you can answer "yes" to those questions, then you can use HCA as an Alexa bridge and doing so is easy. To start, create a program – named whatever you want – as the HCA side of the bridge. Make sure that the program is given a carefully chosen Alexa name as you will say:

Alexa, turn on <name>
Alexa, turn off <name>

If what you are controlling has an ON and OFF state, then add to the program Generic ON and OFF triggers. When constructing the program, you can test to see how it was started and perform different actions for on and off. If what you are controlling doesn't need that, or if you will create separate programs for each action, then there is no need to add the triggers. You will still have to tell Alexa to "turn on" the program because that is the only syntax that Alexa understands.

Once the program or programs are created, you need to ask Alexa to discover your devices – which in HCA includes programs.

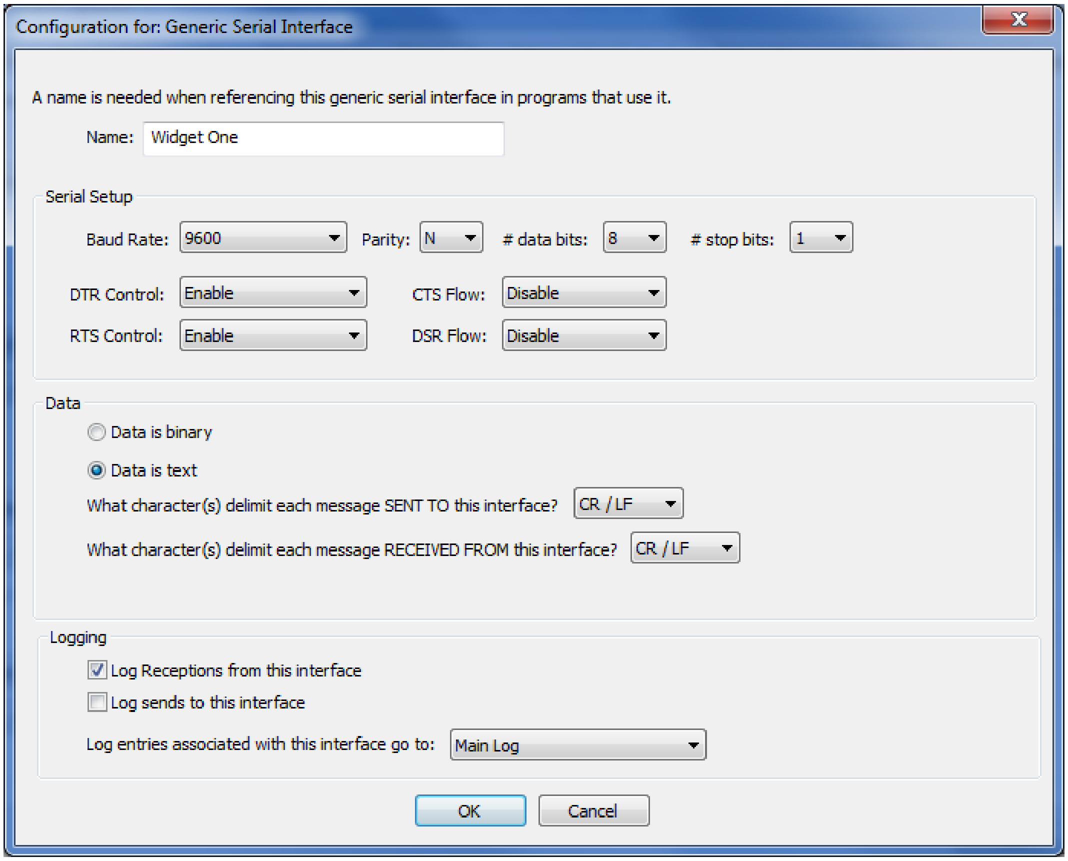
If you are new to HCA, the Getting started series also has Alexa information:

Getting Started with HCA – Client Server

Getting Started with HCA – Amazon Alexa

If you can control your device by a serial connection

One of the HCA interface choices is called a "Generic Serial Interface". Configure one of the interfaces on the hardware setup dialog as this type of interface. Then press the *Configure* button to set all its parameters.



A name is needed when referencing this generic serial interface in programs that use it.

Name: Widget One

Serial Setup

Baud Rate: 9600 Parity: N # data bits: 8 # stop bits: 1

DTR Control: Enable CTS Flow: Disable

RTS Control: Enable DSR Flow: Disable

Data

○ Data is binary

● Data is text

What character(s) delimit each message SENT TO this interface? CR / LF

What character(s) delimit each message RECEIVED FROM this interface? CR / LF

Logging

☑ Log Receptions from this interface

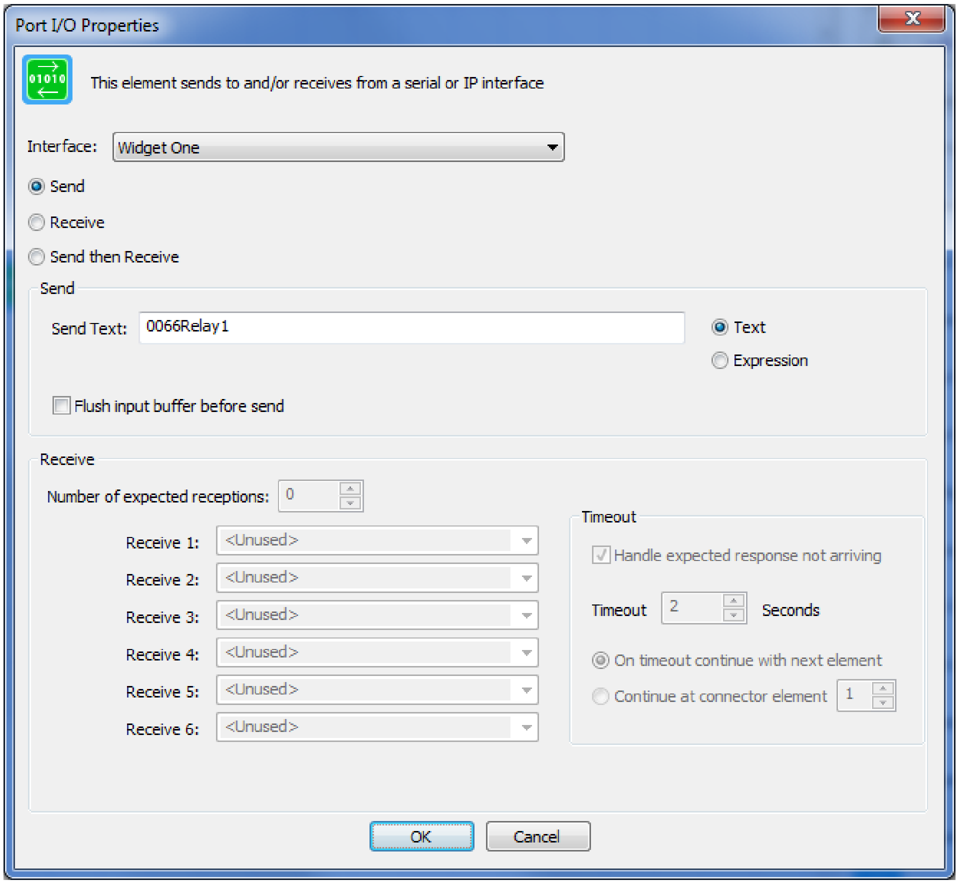
☐ Log sends to this interface

Log entries associated with this interface go to: Main Log

In this dialog the usual serial settings of baud rate, parity, etc are specified.

HCA can both receive from and send messages to the serial port but it has to know how to delimit messages being sent and how to parse a single message out from the data stream when receiving. There are selections for this and the choices depend upon if the message is text or binary. How messages are formatted should be part of whatever information you have on the commands you will be sending to the device.

Once you have made all the choices then close the configuration dialog. The next step is back in the program you created. The Port/IO Visual Programmer element can send to, receive from, or send and receive messages from a generic serial interface.



Port I/O Properties

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

Interface: Widget One

● Send

○ Receive

○ Send then Receive

Send

Send Text: 0066Relay1

● Text

○ Expression

□ Flush input buffer before send

Receive

Number of expected receptions: 0

Receive 1: <Unused>

Receive 2: <Unused>

Receive 3: <Unused>

Receive 4: <Unused>

Receive 5: <Unused>

Receive 6: <Unused>

Timeout

☑ Handle expected response not arriving

Timeout 2 Seconds

● On timeout continue with next element

○ Continue at connector element 1

The configuration of this element determines if it sends or receives. For receptions, the format of the message as specified in the interface configuration lets HCA parse the data received from the serial port into separate messages. Each message is saved in one or more variables.

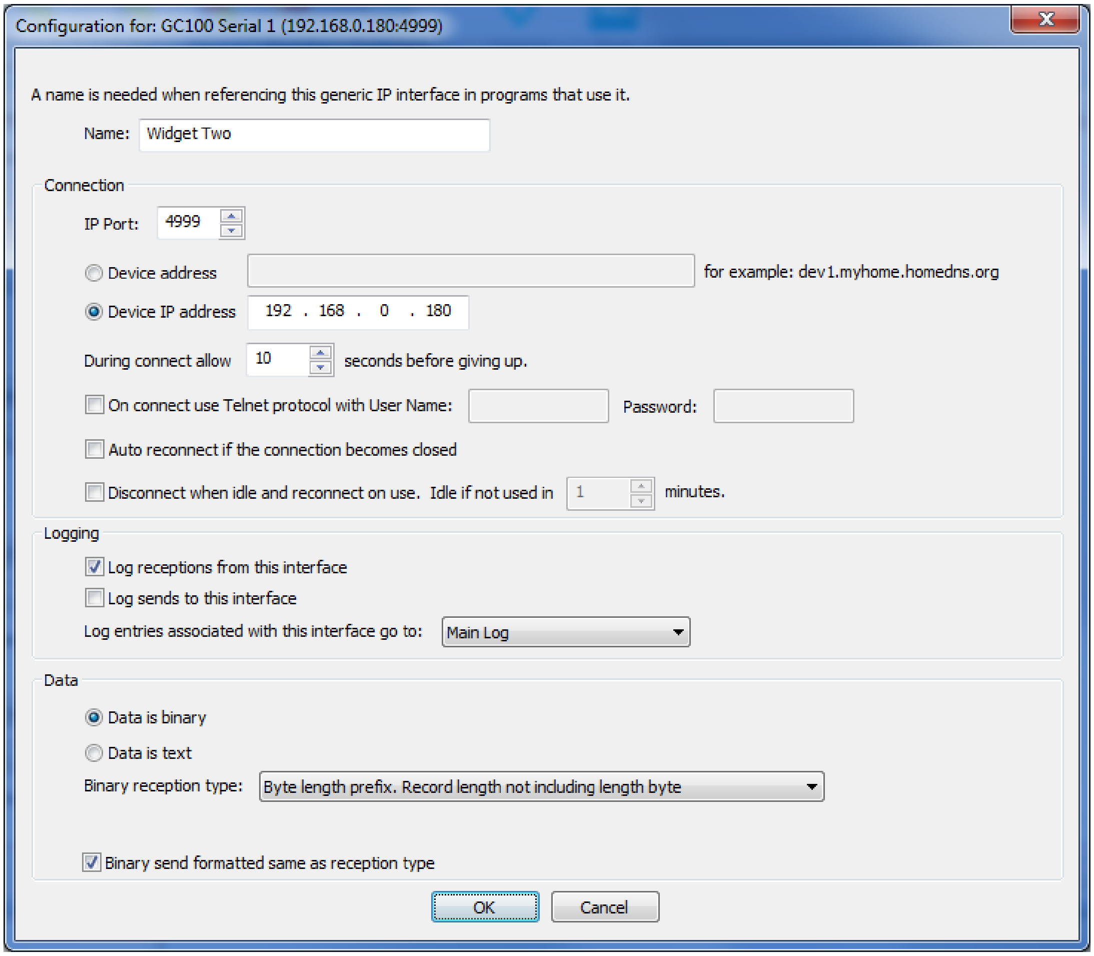
Set the *Send Text* to the command you want to pass to the other end of the serial cable. If the message is binary, then the data is entered in hex as text. It is converted to bytes before being sent. For binary messages there is an option in the interface configuration to “frame” the messages being sent using the same format as specified for message parsing.

With this in place, this flow has been implemented:

Alexa -> HCA Skill -> HCA Server -> HCA program -> Port I/O element -> serial link -> what is being controlled

If you can control your device by a IP connection

If what you want to control can be communicated with using an IP connection, the first step is to configure one of the HCA interfaces as a “Generic IP Interface” in the hardware setup dialog. Once that type is selected, press the *Configure* button to set the parameters.



Configuration for: GC100 Serial 1 (192.168.0.180:4999)

A name is needed when referencing this generic IP interface in programs that use it.

Name: Widget Two

Connection

IP Port: 4999

○ Device address [ ] for example: dev1.myhome.homedns.org

● Device IP address 192 . 168 . 0 . 180

During connect allow 10 seconds before giving up.

□ On connect use Telnet protocol with User Name: [ ] Password: [ ]

□ Auto reconnect if the connection becomes closed

□ Disconnect when idle and reconnect on use. Idle if not used in 1 minutes.

Logging

☑ Log receptions from this interface

□ Log sends to this interface

Log entries associated with this interface go to: Main Log

Data

● Data is binary

○ Data is text

Binary reception type: Byte length prefix. Record length not including length byte

☑ Binary send formatted same as reception type

The IP address and port is specified on this dialog. The IP address can be entered as the usual 4-part set of numbers or as text that is resolved using DNS.

HCA can both receive from and send messages to the network port but it has to know how to delimit messages being sent and how to parse a single message out from the data stream when receiving. There are selections for this and the choices depend upon if the message is text or binary. How messages are formatted should be part of whatever information you have on the commands you will be sending to the device.

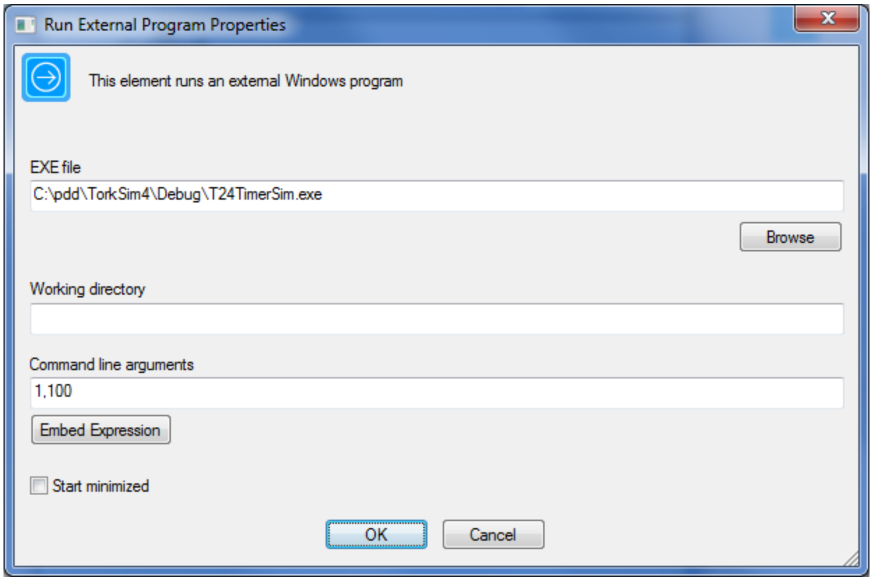
Once that is done then the same Port/IO element described in the serial case above is used. The Port/IO element works exactly the same for serial and IP interfaces.

With this in place, this flow has been implemented:

Alexa -> HCA Skill -> HCA Server -> HCA program -> Port I/O element -> network link -> what is being controlled

If you can control your device from a Windows program

If neither the serial or network connection doesn't do the job, you can create a Windows program and have HCA invoke that program in response to an Alexa command. There is a Visual Programmer element called *Run*. Its properties are:



Run External Program Properties

This element runs an external Windows program

EXE file: C:\pdd\TorkSim4\Debug\T24TimerSim.exe

Working directory:

Command line arguments: 1,100

□ Start minimized

When executed this element starts the named Windows program passing parameters on "the command line" that the program can process.

With this in place, this flow has been implemented:

Alexa -> HCA Skill -> HCA Server -> HCA program -> Run element -> Windows program -> what is being controlled

More information on the how to configure a generic serial interface and the Port I/O element is in this HCA User Guide Generic Interfaces Appendix

In addition to this appendix there are two technical notes that show how Generic Interfaces are used. These are:

Tech Note 401: Integrating HCA with a Lutron controller using a generic IP interface

Tech Note 402: Integrating HCA with an energy monitor using a generic IP interface

##end##