HCA Tech Note 400: Connecting HCA and your own applications

There are several ways that you can connect HCA to applications that you create. These applications could be Windows programs written in a variety of programming languages, mobile applications, or browser based applications. This technical note covers these techniques:

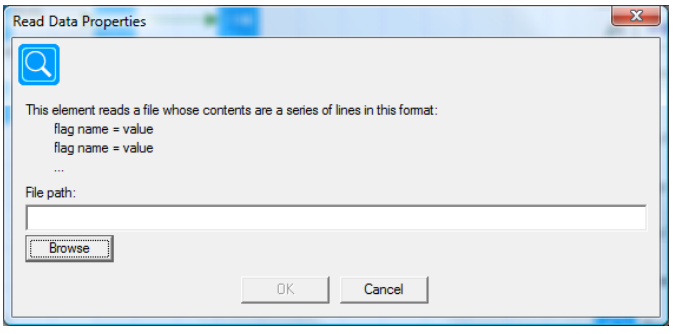
- Exchanging data by file.
- Applications that connect directly to the HCA object.
- Applications that open a TCP/IP connection to the HCA Server port. Messages are sent and received in the HCA TCP/IP Object protocol.
- Applications that use a WebSocket to open a TCP/IP connection to the HCA Server port. Messages are sent and received in the HCA TCP/IP Object protocol.
- Applications that open a TCP/IP connection to a "Generic Server" interface implemented by HCA. The content and format of the messages sent and received is up to your implementation – HCA only handles the transportation of the messages. That transportation can be a stream of characters with message delimiters or framed using the WebSocket protocol.

Exchanging data by file

The simplest method involves having your application create a file that is read by a program using the Read-Data Visual Program element. How the application creates the file and its location is up to you. The Read-Data element specifies the location of the file. The format of the file contains one or more lines with each line of the form:

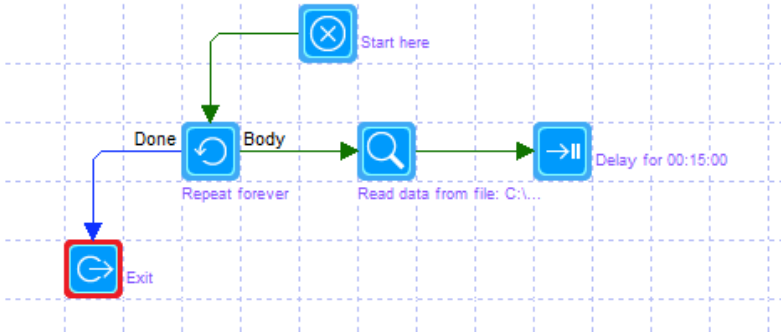
```
flagName=Value
```

When HCA executes the Read-Data element each line is read and processed. A flag matching the *flagName* in the line is found or created and the *value* assigned to it. The properties of the Read-Data element are:



All that is required is the path to the file.

It is up to your implementation to determine how often the file is read by executing the program that contains the Read-Data element. This could be once an hour, once every 15 minutes or whatever. For example, a program, started when HCA starts – there is a program trigger for that – could use a Repeat element – repeat forever – whose body of the repeat element is to read the file and then delay before starting the repeat again. For example:



Exchanging data by file works with HCA Standard or HCA Plus in stand-alone or in client-server mode.

Connecting directly to the HCA object

If the implementation language of your application supports objects, and if that application will execute on the computer that HCA is also executing on (stand-alone or HCAServer), then you can directly use the objects that HCA exposes.

The term “supports objects” means that the application can create an *instance* of an ActiveX object. Prior to calling the methods of this object and accessing and sub-objects, the application is required to create an object instance. How this is done varies with the programming language. In VBScript this is:

```
Set HCA = GetObject(, "HCA.Object")
```

Most programming languages support objects in this way – JavaScript, Java, C++, Visual Basic, etc – so they can be used with HCA. Once an instance of the object is created then all the objects and methods HCA makes available can be used. **This is all documented in the User Guide “Object Model” appendix**.

Sometimes Windows security can make this technique problematic. In general referencing the HCA object works well but in some cases, depending upon how you have configured Windows User Access Control, it may not be possible to create an instance to the HCA object.

A suggestion is to use the VBA example contained in a sample Excel workbook that ships with HCA. This is also described in the user guide appendix. If you can get that to work then your applications should also work.

Directly using the HCA Object works with HCA Plus when running in stand-alone or client-server mode. It is not available with HCA Standard.

Opening a TCP/IP connection to the HCA Server port

You can also create applications that directly connect to the same port opened by the HCA Server to support HCA clients. This is a much more complex technique because the application is responsible for correctly formatting the commands sent to and received from the HCA Server. The major advantage is that the application can be running on any computer and on any platform as long as it can support opening a TCP port (a "Windows socket"), sending data to it and receiving data from it. This is the technique used by the HCA mobile applications.

One advantage of this technique, in comparison to working directly with the HCA ActiveX objects, is that in addition to all the standard HCA Objects and methods, a second set of higher level objects exist that let you not only control objects in the design – like turning a light on or off – but also access the design itself in great detail. This gives an application the ability to configure itself to the design loaded into the HCA Server.

The data passed between the server and your application is entirely in printable text and requires that you assemble the object, method, and the arguments to that method into a string for transmission. When receiving a message from the server a similar decoding of the data must be done.

A complete description of the data formats used and these higher level objects is contained in the HCA Technical note titled "TCP/IP Object Interface".

One problem with this technique is that you must work with your firewall to make sure that your application is allowed to connect though the port you have opened. If an HCA client – or the HCA Android application – can connect then your application should also be able to connect.

Connecting to the HCA Server port works with HCA Plus when running in client-server mode. It is not available with HCA Standard.

Using WebSockets to connect to the HCA Server Port

In a similar manner to opening a port directly to the HCA server, the HCA Server also supports WebSockets. If you are not familiar with WebSockets then this may help:

http://en.wikipedia.org/wiki/Websockets

From the introduction to the linked article:

WebSocket is a protocol providing full-duplex communications channels over a single TCP connection. The WebSocket protocol was standardized by the IETF as RFC 6455 in 2011.

WebSocket is designed to be implemented in web browsers and web servers, but it can be used by any client or server application. The WebSocket Protocol is an independent TCP-based protocol. Its only relationship to HTTP is that its handshake is interpreted by HTTP servers as an Upgrade request. The WebSocket protocol makes possible more interaction between a browser and a web site, facilitating live content and the creation of real-time games. This is made possible by providing a standardized way for the server to send content to the browser without being solicited by the client, and allowing for messages to be passed back and forth while keeping the connection open. In this way a two-way (bi-directional) ongoing conversation can take place between a browser and the server. In addition, the communications are done over TCP port number 80, which is of benefit for those environments which block non-web Internet connections using a firewall. WebSocket protocol is currently supported in several browsers including Google Chrome, Internet Explorer, Firefox, Safari and Opera. WebSocket also requires web applications on the server to support it.

Once a WebSocket connection to the HCA Server is made then the data supplied is exactly the same as would be sent over a TCP/IP port connection as described in the previous section. For example this could be used as part of a web page's active content:

```
var ws = new WebSocket("ws://192.168.2.8:2000/websocket");
ws.onopen = function()
{
   // Web Socket is connected, send data using send()
   ws.send("HCA000B011002001");
   ws.send("002500340044    HCAObjectDevice.OnDen - Lamp");
   alert("Message sent");
};
```

When the browser makes the connection, all the negotiation messages necessary to implement the web socket protocol are handled by the HCA Server. All you need do is to open the socket using the IP address of the server and the HCA Server port - the same port number is used for all HCA clients. The initial web-socket protocol message tells the server that this is a web socket connection and data sent back and forth should be framed according to the web socket specification. The web socket protocol implemented by the HCA Server matches RFC-6455. HCA does not implement any of the extensions for per-message compression nor does it implement secure connections (use of "wss:").

Once the connection is open then the same text based messages that format object methods – see previous section - are sent and received. The only difference is that the data is being framed and un-framed as it passes though the WebSocket – and that's all handled by HCA so you need not concern yourself with it.

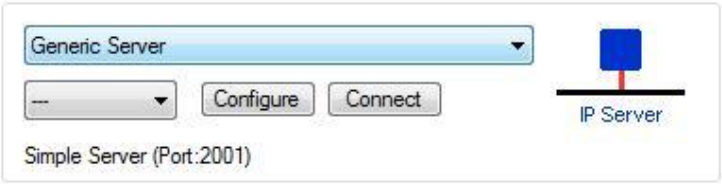
There are two major advantages to using a WebSocket connection. First it is easier to write VBScript inside a browser web page than just about any other implementation method. Second, since it runs "inside" a browser your implementation works on any platform that supports a browser that implements WebSockets. The disadvantage is that JavaScript is a fairly simple scripting language and that may make it more difficult to implement anything very complex.

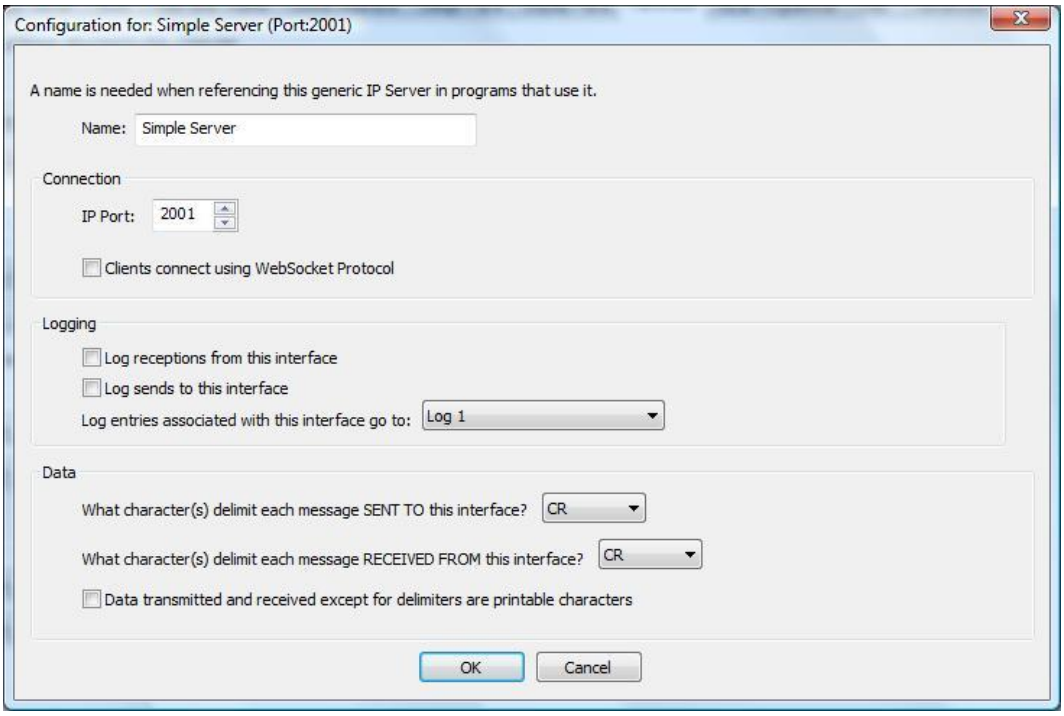
Connecting to the HCA Server port with a WebSocket works with HCA Plus when running in client-server mode. It is not available with HCA Standard.

Generic Server

In HCA 12 the concept of a "Generic Server" was implemented. You can select this as one of the interface types in the HCA Options hardware connection tab. A generic server allows for creation of applications that connect to HCA, send commands into HCA, have those commands carried out, and optionally receive responses back.

What are those commands and how are they encoded? That is up to your implementation and both sides of the communication are implemented by you. On the non-HCA side, the application opens a TCP/IP port directly or uses a WebSocket connection, and on the HCA side Visual Programs you implement handle the processing of those messages. To create a generic server, on the hardware connection tab of HCA Options, select Generic Server and then press the Configure button.





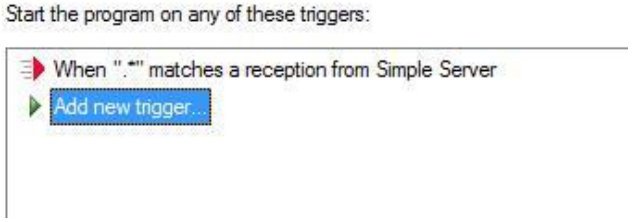
The port number must **not** be the port that the HCA Server is using. Whatever port you choose must be one that will be passed by any firewalls to the computer that HCA is executing on.

Like the Generic IP Interface and the Generic Serial interface, you can specify the characters that separate messages. 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.

As an example, suppose you had a web page with a link that when clicked upon sent the message “One” to an HCA generic server. What does that message do? Let’s assume that you want it to start a program called “P1” and have that program perform whatever action is needed. After that program is started then the server should respond back with “OK”. The web page HTML is this:

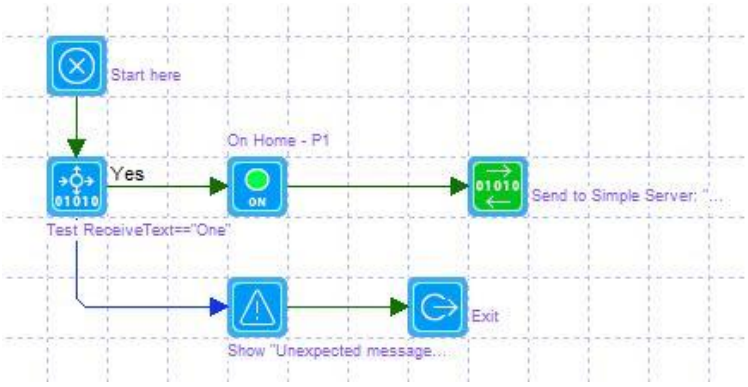
```
<!DOCTYPE HTML>
<html>
<head>
<script type="text/javascript">
function WebSocketTest()
{
  if ("WebSocket" in window)
  {
     // Open the web socket
     var ws = new WebSocket("ws://192.168.2.10:2001/websocket");

     ws.onopen = function()
     {
        // Web Socket is connected, send data using send()
        ws.send("One");
     };

     ws.onmessage = function (evt)
     {
        var received_msg = evt.data;
        alert("Message is received...");
        document.write(evt.data);

     };

     ws.onclose = function()
     {
        // websocket is closed.
        alert("Connection is closed...");
     };
  }
  else
  {
     // The browser doesn't support WebSocket
     alert("WebSocket NOT supported by your Browser!");
  }
}
</script>
</head>
<body>
<div id="sse">
   <a href="javascript:WebSocketTest()">Run WebSocket</a>
</div>
</body>
</html>
```

The HCA side of this application is implemented in a single program that is triggered on a reception from the server.

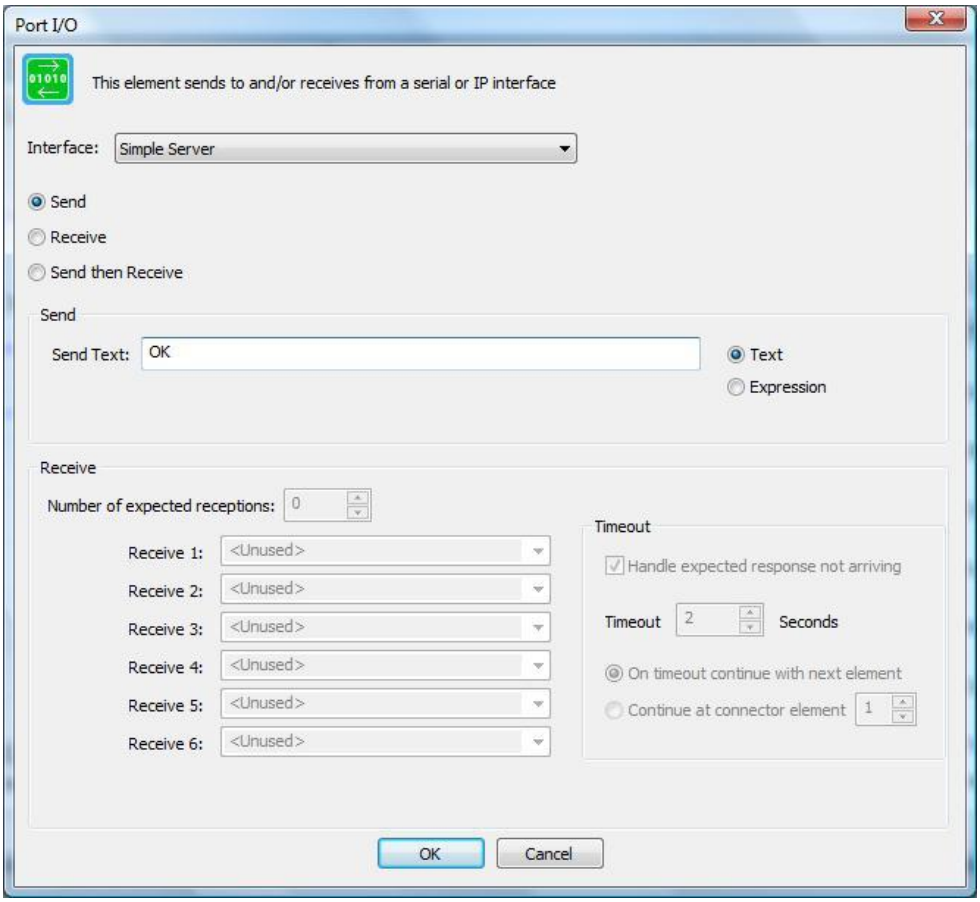


The same trigger mechanism is used as with the generic serial and generic IP interfaces – a regular expression that matches the message text received. In this example the regular expression matches any reception.

The program being triggered – called “Handle Server Message” is:



The program checks to see if the reception text is “one” and if so it starts the “P1” program and then replies back to the client – in this case the web page that opened the web socket. The Port-IO element is used for that message transmission and is the same as used by the generic serial interface and the generic IP interface.



In this example, we used a web page and a WebSocket as the “client side” of the implementation. But it could have been as easily accomplished by a client application opening a TCP port (a “Windows Socket”) and sending to and receiving from that port. On the HCA “server” side the only difference would have been to un-tick the “*Clients connect using the WebSocket protocol*” checkbox in the Generic Server configuration.

The Generic Server method works with HCA Plus in stand-alone or client-server mode. It is not supported by HCA Standard.

##end##