HCA Tech Note 502

HCA Cloud Developer Access (as of 13-June-2020)

Using the same facilities used to support partner services, HCA provides a way for individual users to access to their own HCA Server using the same cloud interface that HCA provides for partners. This makes it possible for individual users to create their own hardware and software applications that connect to and control their own automation design.

***Note**: **This isn't beginner stuff**. To make much sense of what follows you should first review several of the HCA Technical notes and user guide chapters. You should also know what a HTTP POST is and how to make one in whatever implementation language you plan to use. You should also know what JSON is and how it is formatted.*

Please have these available:

1. Technical Note 500: HCA Cloud
2. Technical Note 501: HCA Cloud partner services
3. User Guide Appendix 7: Object Model
4. Technical note 450: HCA Server protocol
5. Technical note 113: JSON support

There are two parts to Developer Access.

- In the same way that HCA integrates with partner services, like the Ring Doorbell, your application can also use this method. Your application can POST to a URL and supply JSON encoded data. In your HCA design a program is designated that is started when it receives the message from your application. For example, suppose you created your own hardware sensor out of the popular Raspberry pi. When that sensor detected some condition, it can send a message to the HCA Cloud and the cloud in turn sends the message to the HCA server where the program you designated starts.

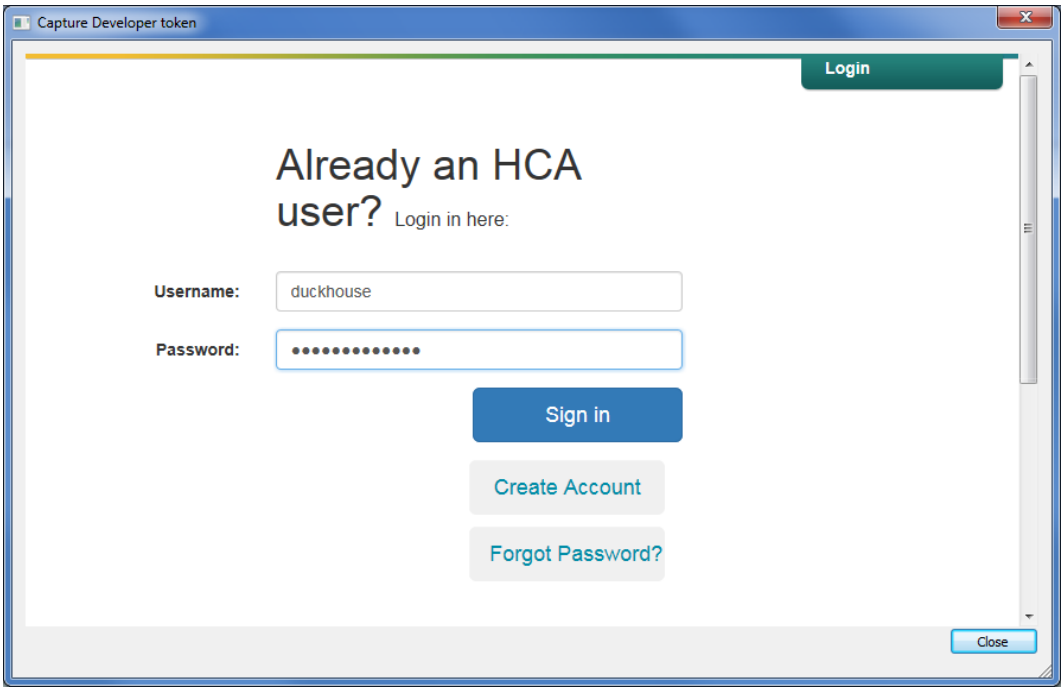
- Developer access also offers a REST interface to the HCA Server. Using this you can send messages to the HCA Cloud via HTTP POST and the cloud sends that message into your HCA Server for execution. Using this you can invoke any of the object methods documented in the HCA Object user guide appendix. For example, you may have a software application where you want to be able to control the lighting in your home. Using this method your application can easily communicate with the server using a very simple interface.

HCA Tech Note 502

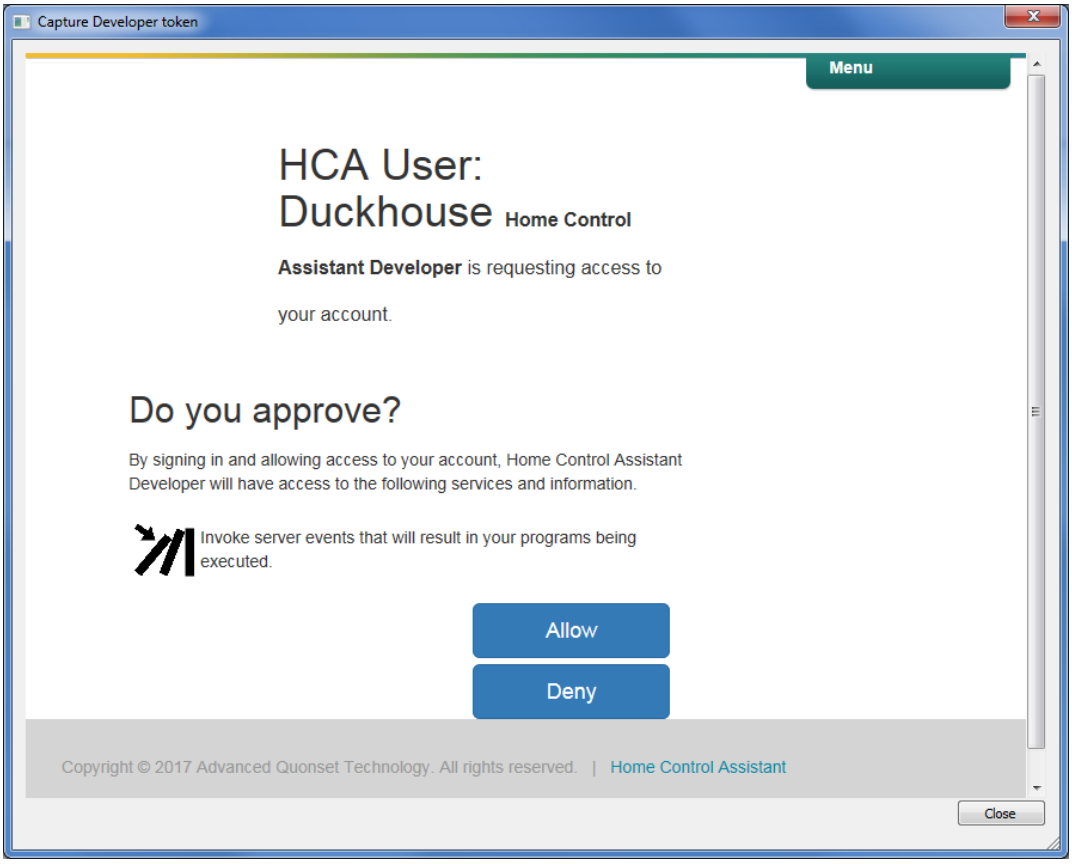
Authorizing the “HCA Developer Access” into your cloud account

Before you can use either of these methods to access your HCA server you must first grant access and receive an “access token”. The access token is just a string of 256 random characters. All you need do is to capture it during the authorization process – described below – and use it in each transaction with the HCA Cloud.

Assuming you already have a HCA Cloud account, the first step to using HCA Developer access is to press the “HCA Server Access for Developers” button in the “HCA Cloud” ribbon category. Read the, hopefully informative, popup message and close it. A new window opens where you can log into your HCA Cloud account.

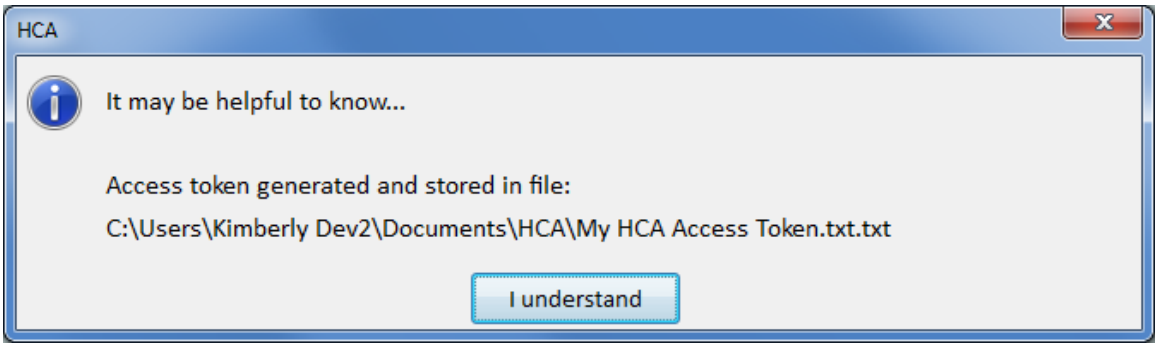


Enter your user name and password and then press “Sign in”. The next page is the usual one for authorizing access:



Press “Allow” to continue.

After the access token is generated and it is saved in a file for your use.



Don’t lose this file and so lose the token! If you do, then you will have to revoke developer access and then reauthorize. This creates a new token and all your applications that use the old token will have to be updated.

Now that you have a token, you can use it in the two ways described in the next sections.

Developer Access as a Service

There are two sides in the conversation between your application and HCA. There is the application side that sends a message to the HCA Cloud, and your HCA server that receives the message from the HCA Cloud.

Your application: Sending

To pass an event message to your HCA Server through the HCA Cloud, make a HTTP POST from your application. The portions of that HTTP interaction are:

- POST to URL: **trigger.hcatech.com/api**
- Must use HTTPS on port 443

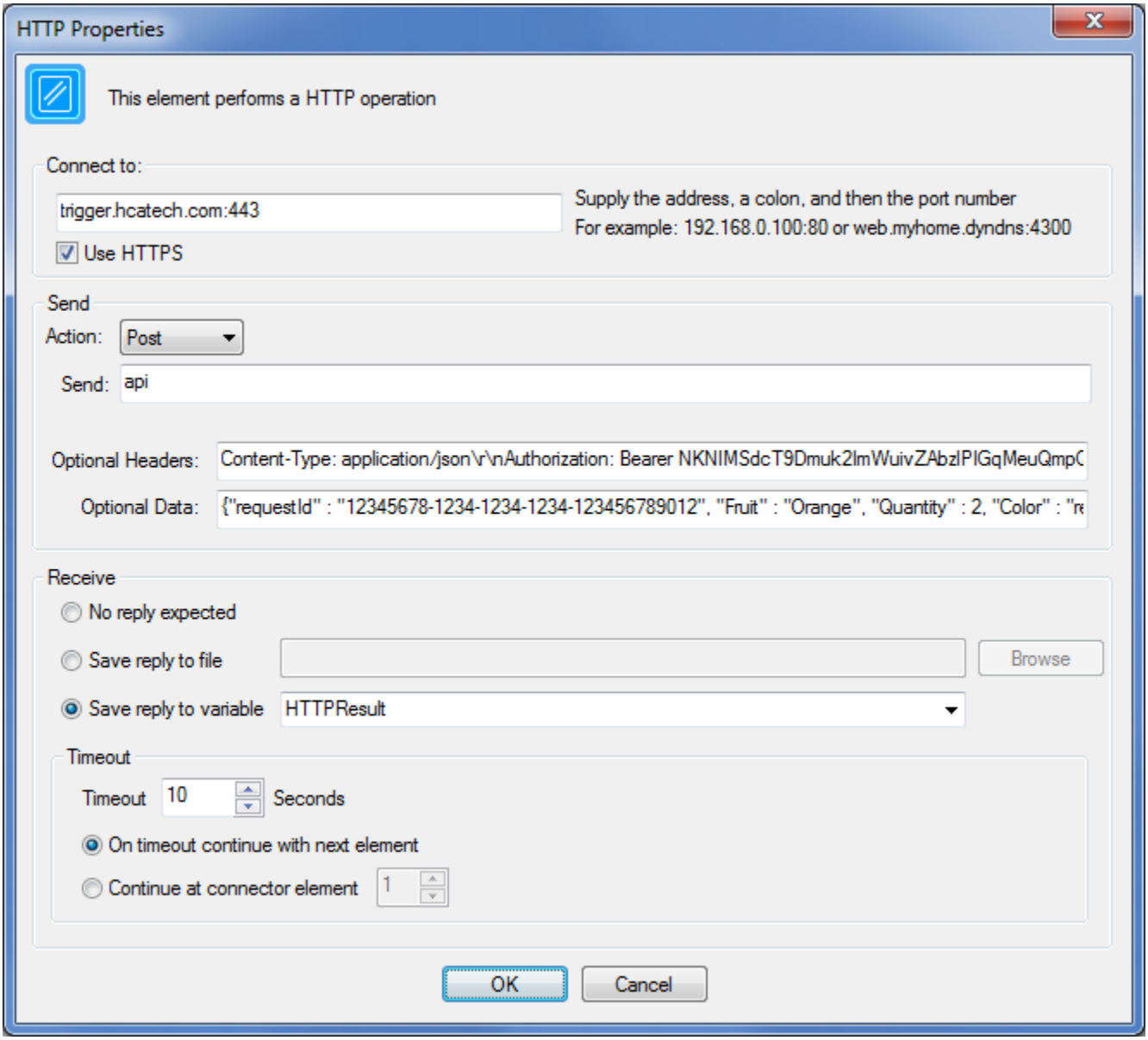
Required message headers:

- Content-Type: application/json
- Authorization: Bearer <the token>

The data that comes along with the POST must be encoded in JSON format. The first key/value pair must be named “RequestId” whose value is unique but can be provided in any format desired. The remainder of the JSON can be anything you want. Here is an example:

```
{
  "requestId":"12345678-1234-1234-1234-123456789012",
  "portal":"Mailbox",
  "Action":"Open"
}
```

Here is a HTTP element configured as needed:



Note these important details:

- The connect to and the “Use HTTPS” checkbox must be configured exactly as in the image
- The Send Action is “Post”
- The headers are separated by each other using “\r\n” and are formatted as show in the image. Your token will of course be different

- The “Optional Data” is what you send in and is specific to your application

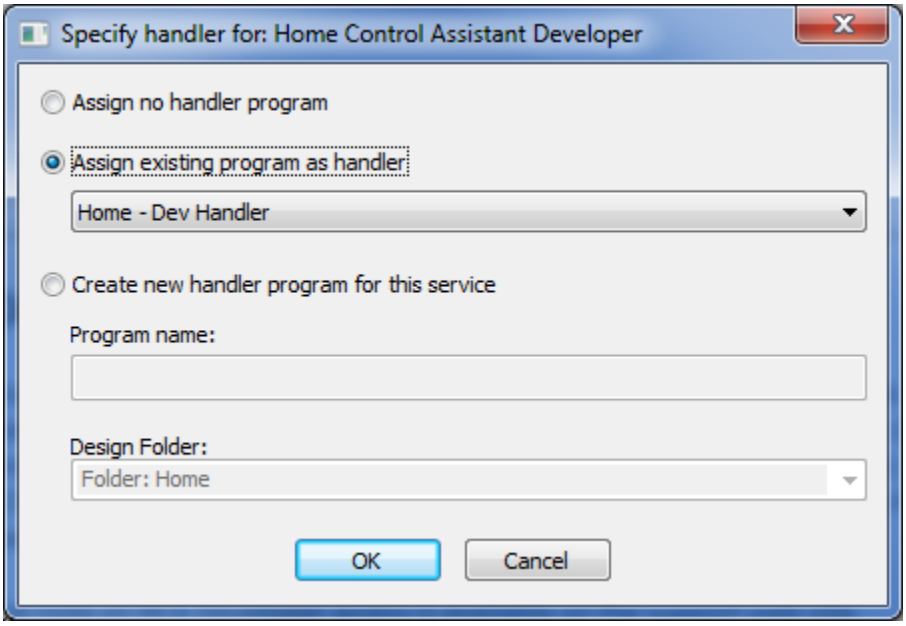
If you enable HTTP logging for this program – “Tools” ribbon category “Aux Log Setup” button) you will see a log like this:

```
05:25:00 InternetHttpOpen
05:25:00 InternetHttpSetTimeouts
05:25:00 Connect trigger.hcatech.com:443
05:25:01 Initialize ok
05:25:01 Send "api"
05:25:01 Headers "Content-Type: application/json
Authorization: Bearer
NKNIMSdcT9Dmuk2ImWuivZAbzlPIGqMeuQmpQiLJghZym27xQpFBVTINj5MlROgprs72UDqnoBbTh02hmb69jly4FuzGm2
dQoUO9gOv3bpzFhfQtaG02tToq8qcA4rR9WMZzzr7KAe7G74803JSrZAEhlSL5XcH"
05:25:01 Data "{"requestId" : "12345678-1234-1234-1234-123456789012", "Fruit" : "Orange", "Quantity" : 2, "Color" :
"red"}"
05:25:01 Send:InternetHttpOpenRequest
05:25:01 Send:InternetHttpSendRequest
05:25:09 Send:InternetHttpReceiveResponse
05:25:09 Send:InternetHttpQueryDataAvailable
05:25:09 Response cbData: 78 "{"requestId":"12345678-1234-1234-1234-123456789012","payload":{"status":"ok"}}"
05:25:09 Send:InternetHttpQueryDataAvailable
05:25:09 Op Complete 0
05:25:09 Terminate
```

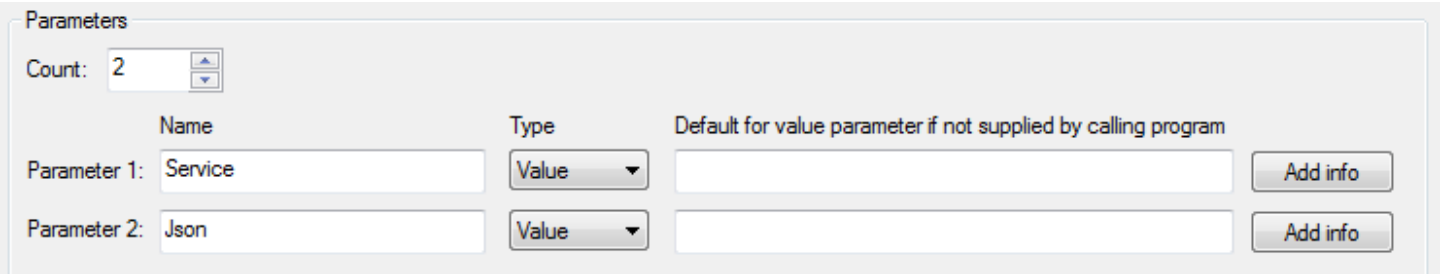
Your HCA server: Receiving

To configure your HCA design to start a program when your message is received, press the "Inbound Connection Manager" button in the "HCA Cloud" ribbon category. Locate the "Home Control Assistant Developer" service and press the *Assign Handler* button.

Select the "Home Control Assistant Developer" service and press the "Assign Service Handler" button.



In this dialog you can either assign an existing program to start when messages are received or create a new program that is the handler for the message. The program selected must have "Supports parameters" enabled in the Advanced tab of the program's properties. The two parameters should be like this:



The "Json" parameter receives whatever you put in the data of the HTTP post to trigger.hcatech.com/api. You can use the JSON facilities in the Compute element to open the JSON and extract out the various values.

Here is a timeline of what happens:

1. Your application does a POST to trigger.hcatech.com/api passing along the access token and any JSON you want.

2. The HCA cloud receives that message, uses the token to locate your HCA server, connects to the server and sends a message to it with your JSON then disconnects.

3. The HCA Server accepts the message and finds the handler program for HCA Developer access. It starts that program passing in the JSON your application sent. You can then decode that JSON and do whatever you want.

Developer access to the HCA Server directly

In addition to sending messages to the HCA Cloud to cause handler programs to start, HCA Developer access has another use entirely. By sending messages to the HCA Cloud – using a different URL and data – you can cause any method of any of the HCA Object to be invoked.

In the HCA Objects appendix of the User Guide you will see that HCA makes available several objects – device, program group, etc. – and methods – On, Off, Start, etc. - that are used with those objects to do things. For example, to control a device to 50%, to start a program, to assign a value to a variable, read the log, etc. The list is comprehensive and is documented in the object appendix of the user guide.

Using this you can create your own applications that talk to the HCA Cloud and in turn pass object/method requests into your HCA Server.

***Note**: This is just another way to do what you can already do now albeit much simpler in many cases. For example, you could have written an application that runs on the same computer as the HCA Server and directly access the HCA Objects using facilities in your application implementation language. Or you could have written an application that makes a direct TCP/IP socket connection to the server computer and sent requests that way. Those ways still work but they are not easy to implement. This new method is simpler.*

To communicate to the HCA Cloud to invoke a HCA Object method:

- POST to URL: **trigger.hcatech.com/server**
- Must use HTTPS on port 443

Required message headers:

- Content-Type: application/json
- Authorization: Bearer <authorization token>

The data that comes along with the POST is encoded in JSON format. The first key/value pair must be named "RequestId" whose value is unique but can be provided in any format desired.

The remainder of the JSON must use these key/value pairs:

- “Group”: This is either “HCAApp” or “HCAObject”
- “Command”: This is the HCA Object and method to invoke
- “Params:” This is an array containing the argument values for the command in the order expected by the method.

For example:

```
{
   "requestId":"ff36a3cc-ec34-11e6-b1a0-64510650abcf",
   "group":"HCAObject",
   "command":"Device.On",
   "params":["Dining Room-Lights"]
}
```

If the token in the HTTP POST is valid, and you have HCA Developer access authorized into your account, then a HTTP POST to “trigger.hcatech.com/server” with the correct HTTP headers and the above JSON, causes the “Dining Room-Lights” go on.

What is the group?

There are two groups: HCAObject and HCAApp. When working with HCAObject you should look in the User Guide Objects appendix as that gives the methods of the HCA, Device, Program, Group, Controller, Flag, Schedule, Log, and Display objects.

The HCAApp methods are defined in the Server Protocol technical note.

What is the command?

For the HCAObject group, the command is the object name followed by a dot followed by the method name. For example, “Device.On”, or “Program.Start”. It is important to have the text *exactly* as in the documentation and upper/lower case is important.

For the HCAApp group, the command is one of those given in the Server Protocol technical note. For example, “ThermostatState” or “ThermostatChange”.

While all the HCAObjects and methods are available with developer access, not all HCAApp methods are usable with Developer Access. The ones that are:

- GetHomeModeNames
- GetHomeMode
- SetHomeMode
- ThermostatState
- ThermostatChange
- FormatText
- GetScheduleNames
- SetCurrentSchedule
- GetServerStatus
- GetDesign
- GetDisplays

- GetLog

What parameters go with what command?

For the HCA Objects, each method and its arguments is documented in the User Guide Objects appendix. For example, Device.On:

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

It says that the ON is a method of the Device, Program, and Controller objects. It takes one parameter and that is the name of the object to be controlled on.

For the HCAApp methods, the Server Protocol technical note provides all the deyails of the method parameters and results.

How are results obtained?


```
{
  "response": {
    "returncode":"0",
    "items": [
      "0"
    ],
    "group":"HCAObject",
    "command":"Device.On",
    "params": [
      "Dining Room-Lights"
    ]
  }
}
```


You can retrieve the result from the HTTP POST and it contains the results of the command. For example, the Device.On example given above, the result is show at the left.

The “returncode” is the return code of the object method. These are documented in the Objects appendix of the user guide. The elements of the command are also supplied to create a “stateless” interface.

```
{
   "response": {
      "returncode":"0",
      "items": [
         "Home & Awake",
         "Home & Asleep",
         "Away"
      ],
      "group":"HCAApp",
      "command":"GetHomeModeNames",
      "params": [
      ]
   }
}
```


In this example, HCAApp.GetHomeModeNames was invoked and it results in three items in the response.

How are errors reported?

There are several errors that can be reported. For example, in a device control operation you could name a device that doesn't exist. In the Object user guide appendix, there is a table that documents what all the return codes are and their meaning. In general, all result codes that report an error are negative numbers. Consult the documentation for specifics. Here are two examples:


```
{
   "response": {
      "returncode":"-1",
      "items":"",
      "group":"HCAObject",
      "command":"Device.On",
      "params": [
         "Living Room-Lights"
      ]
   }
}
```


In this example, all the parts of the command are correctly supplied, but the device name "Living Room-Lights" didn't exist in the design. An error of -1 is returned.

```
{
   "response":{
      "returncode":"-104",
      "items":"",
      "group":"HCAObject",
      "command":"Device.on",
      "params":[
         "Living Room-Lights"
      ]
   }
}
```


In this example, an error is made in the command. It should be Device.On not device.on. In this case it returns an -104 error. These types of errors are documented at the start of the HCAObject appendix.

And finally, you will get errors if the JSON supplied isn't correctly formatted.

How is security handled?

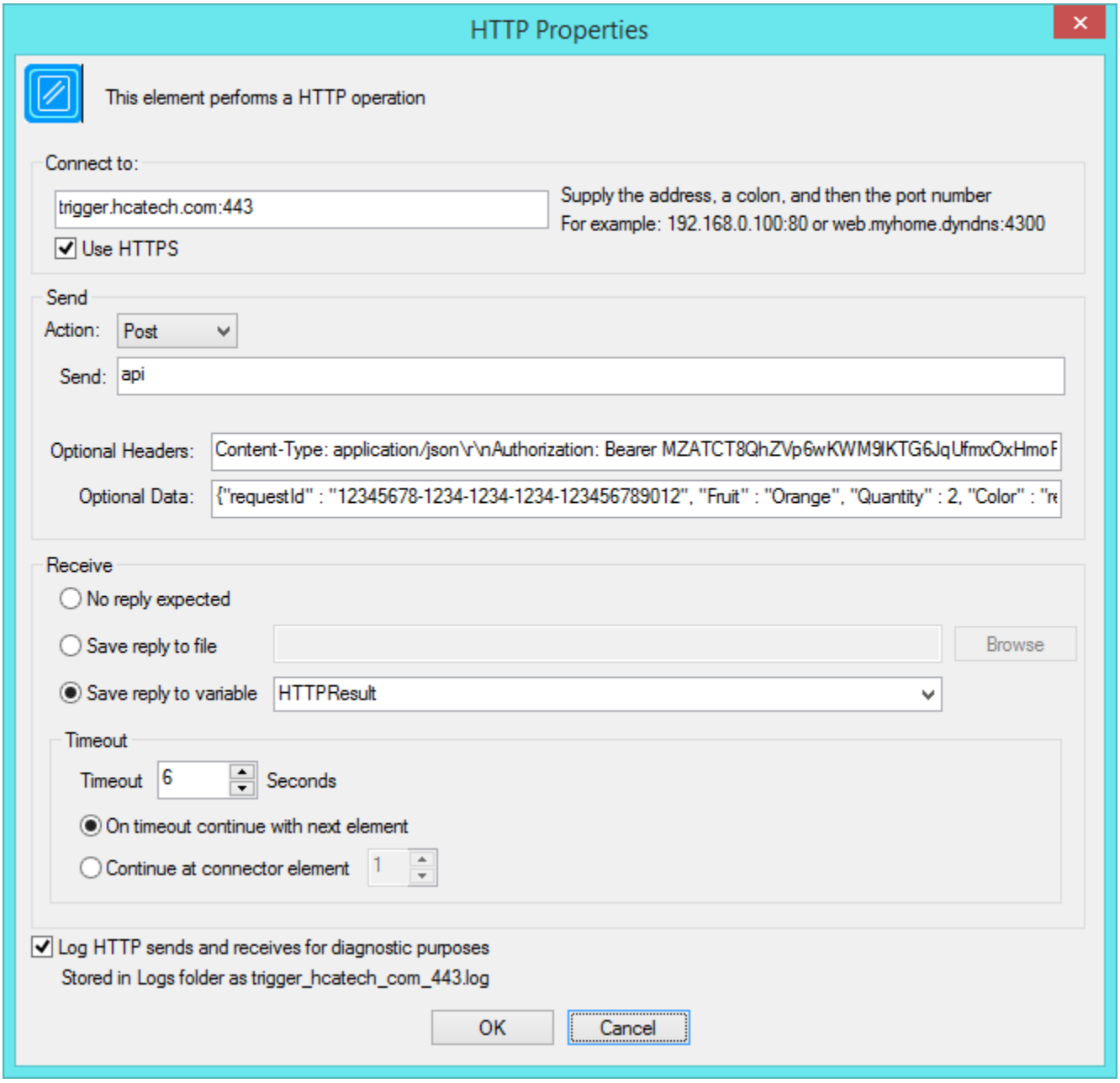
Because you must supply the access token as part of the HTTP POST, you don't have to be concerned about the remote access password. The HCA Cloud takes care of all of that for you. In the Objects User Guide appendix are documented functions named SetPassword and RemovePassword but your application doesn't need to use them.

Getting started – Trigger.HCATech.com/api

The "thing" sending to the HTTP cloud – and then on to your server – is probably going to be some piece of hardware or software you created. An application running on a Raspberry Pi or something like that. Here is a good way to have a working example so that you can make sure that the HCA Server side works, and you understand all the details of how to connect to the HCA Cloud.

What we are going to do is rather silly, but we are going to use HCA to control the HCA Server. All we are going to do is to use the program HTTP element to make the Post to trigger.hcatech.com. We could do the same thing using CURL or something similar, but this is what we know so let's stick with it.

Here is the element:

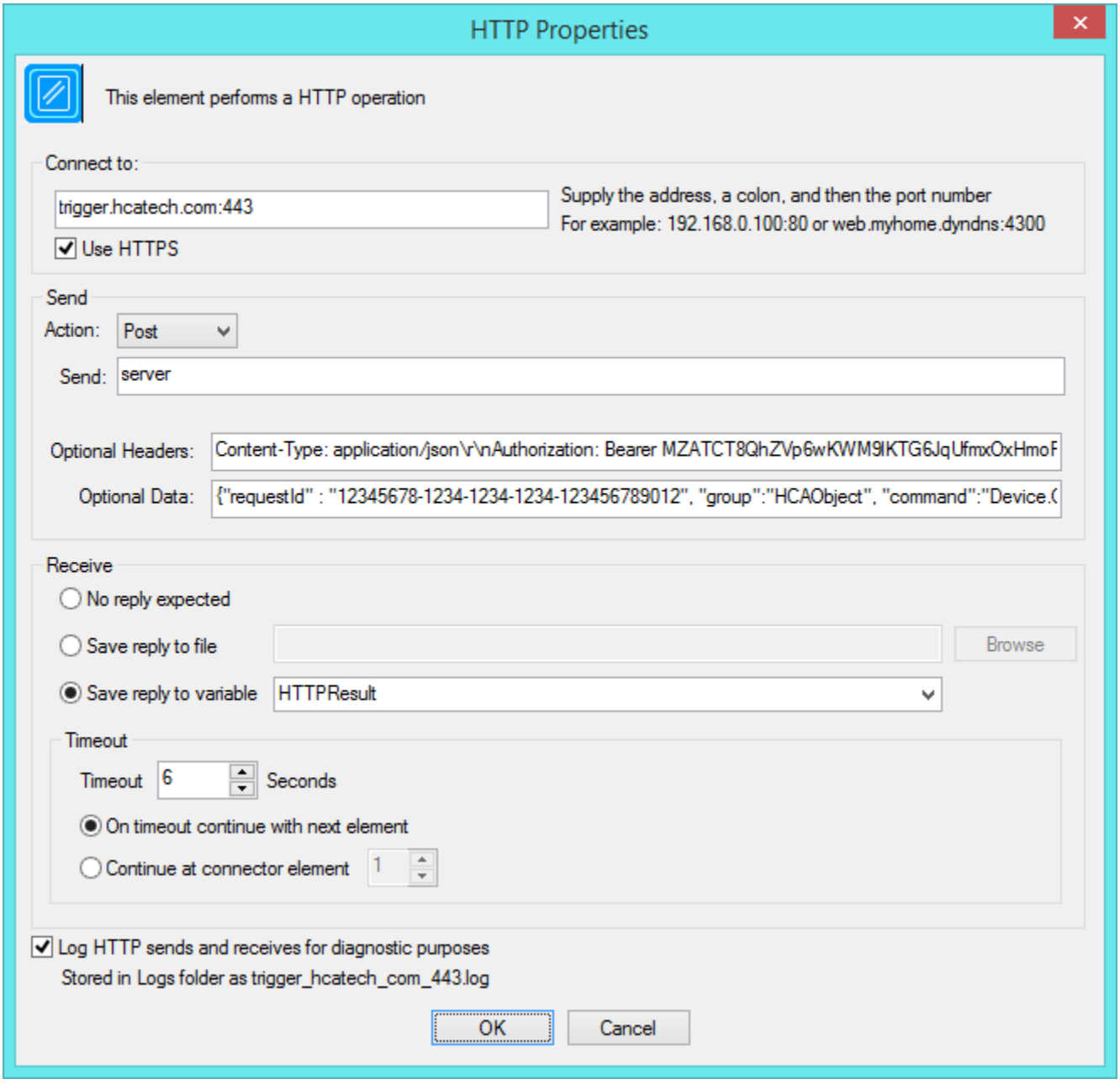


Note: Review the Technical note on the HTTP element for any details on the element options.

What this element does is to connect to trigger.hcatech.com on port 443 (the HTTPS port) and do a POST to “api”. The other element settings are as described on page 4 above. The advantage of this test is that you can do it yourself and make sure that you understand the settings of the HTTP action. You can also test the server side of the conversation where you implement a handler for the message.

Getting started – Trigger.HCATech.com/server

To communicate with the server to affect a device, here is a HTTP element that does that:



The difference than the previous example is that the Send shows “server” and not “api” and the “Data” is the JSON for the Server request. In this example it is the same JSON shown at the top of page 8.

As was said at the start of this technical note, this isn't beginner stuff, but it provides a way for your applications to connect to your server in a clearly defined manner using very simple tools – HTTP and JSON.

##end##