GEORGE HOME AUTOMATION

Users Guide V1.3

Table of Contents

Introduction

George Home Automation (or GHA) is an application platform designed to support complex home automation scenarios. Its object-oriented design gives end-users the flexibility to create objects that model the everyday items we see in our homes and map those items to underlying devices to control our home environment.

Key features of GHA include:

- Object-oriented paradigm that implements standard models (aka Capabilities) for devices.
- The complexity of the underlying device implementation is invisible to the standard Capabilities provided by GHA.
- Segregation between standard device Capabilities and underlying implementation allows for the use of multiple underlying device technologies and the ability to quickly replace underlying technologies without impacting how the end-user interacts with GHA.
- Built-in scripting capability to implement user-specific logic in response to events
- Text-to-Speech support
- Custom Modules may be developed within the application itself to implement new functionality
- Open device driver model allowing third parties to develop GHA support for their devices
- Built-in Scheduling module to implement automation flows based on date, time, sunrise or sunset events
- Custom Scenes to aggregate multiple devices with unique states (i.e. Bedtime, Movie time, etc.)
- Sophisticated media management capability including an extensible catalog of various media types (audio, video, broadcast, satellite, etc.)
- Built-in MQTT server and generalized MQTT device modeling to support compatible devices
- User presence detection
- Built-in device drivers for
    - ZWave
    - ISY
    - Venstar thermostat
    - Sonos
    - Nuvo tuner
    - Nuvo Grand Concerto matrix switcher
    - Global Cache GC100
    - BF-430 network serial port
    - Barix media streamer
    - Built-in computer audio device
    - Phillips Hue bridge and devices
    - SmartThings hub devices
    - Konnected Pro Security alarm hardware
    - Noonlight Alarm monitoring

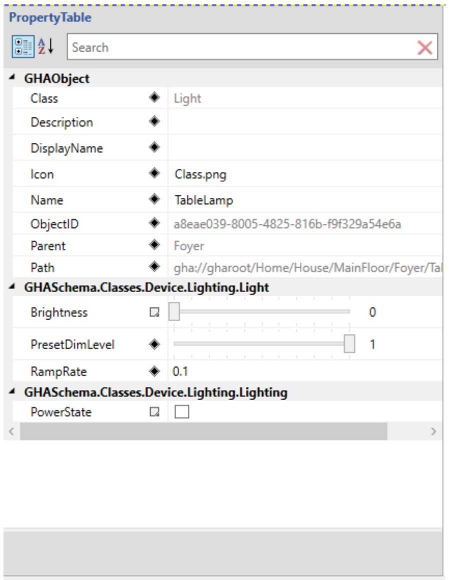
Key Concepts

There are several concepts that are necessary to understand when starting out with GHA. These are generally described in this section. Further details on these topics may be found elsewhere in the documentation.

GHA Hierarchy

The entire GHA system lives in a hierarchy that contains instances (called GHA Objects) of predefined or custom developed classes used to implement various automation scenarios. The out-of-the-box hierarchy contains the following root nodes:

- Home – Contains models for locations and devices that are part of the automation environment
- Devices – Drivers that implement standard **Capabilities** and control physical devices live here
- Media – Catalog of media managed by GHA
- Scenes – Predetermined collections of device states that can be activated and deactivated manually or through automation
- Modules – Provides for the creation of custom classes to implement new GHA capabilities
- Information Sources – Contains services that provide data from external sources (e.g. Weather)
- Monitors – Provides various monitors for GHA subsystems (i.e. Alerts, Task Scheduler)
- Server – Contains configuration nodes for various server functions



GHAObject		
Class	◆	Light
Description	◆	
DisplayName	◆	
Icon	◆	Class.png
Name	◆	TableLamp
ObjectID	◆	a8eae039-8005-4825-816b-f9f329a54e6a
Parent	◆	Foyer
Path	◆	gha://gharoot/Home/House/MainFloor/Foyer/Tal
GHASchema.Classes.Device.Lighting.Light		
Brightness	□	0
PresetDimLevel	◆	1
RampRate	◆	0.1
GHASchema.Classes.Device.Lighting.Lighting		
PowerState	□	□

Figure 1 GHA Object with Properties

GHA Objects

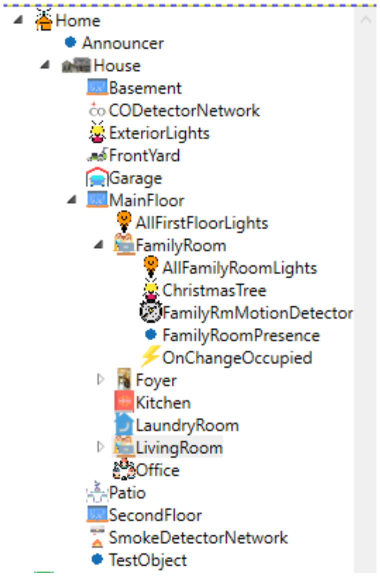
GHA Objects in the hierarchy are *instances* of classes that implement various functions. Each GHA Object has a unique Object ID that is represented as a Globally Unique Identifier (GUID). Each object can be references by its GUID or a *Path*. The Path is also unique and represents the object's location in the hierarchy. The path is a universal resource identifier (URI) starting with the prefix "gha://". Succeeding parts of the path describe the lineage of the object. For example, an object with the path: gha://gharoot/Home/House/MainFloor/Foyer/Chandelier shows where in the hierarchy the Chandelier object resides. The Chandelier's parent is an object called "Foyer". The Foyer's parent is an object called "MainFloor", and so-on. A GHA Object may have at most *one* parent and have many children.

GHA Objects contain various properties that describe the object itself and its current state. All GHA Objects contain the default properties Class, Description, DisplayName, Icon, Name, ObjectID, Parent, and Path. Object-specific properties are also defined for objects to describe its state. For example, the object shown in Figure (1) represents a Table Lamp which is an instance of the *Light* class.

In addition to the standard properties associated with every GHA object, the *Light* class shown here has properties unique to it: Brightness, PresetDimLevel, RampRate, and PowerState. It is also worth noting that this object is a good example of inheritance. As an object-oriented platform GHA makes heavy use of inheritance to simplify the creation of new capabilities. We will talk more about inheritance when we discuss custom modules.

The Home node

The Home node in GHA is used to model the space to be automated. Locations may be specified under the Home node. For example, a *House* object is normally the first object placed under the Home node. While not strictly a requirement, it is a best-practice. The House object can then contain various objects used to provide logical separation (i.e. floors, garage, basement, etc.). These objects can contain children to further subdivide the space (i.e. bedrooms, offices, etc.). Finally, device objects used to represent physical devices can be placed anywhere under the Home node hierarchy. For example, a Family Room object can contain devices model objects that represent lights, motion detectors, presence detectors, thermostats, etc. Figure (2) shows an example of a Home node and its hierarchy containing a typical three-floor home.



The Home node hierarchy in GHA is special in that it separates the physical device controllers (i.e. Insteon, SmartThings, etc.) from logical representations of devices. Logical representations "live" under the Home node in the GHA hierarchy. These logical representations are objects that contain certain **Capabilities**. When used in a Home level object, **Capabilities** define the user-level specification of a device. For example, a Light contains the **Capabilities** *PowerState* and *Dimmable*. Those **Capabilities** manifest themselves as properties in the Home level object. In Figure (1), the PowerState property represents the *PowerState* capability as do the Brightness, PresetDimLevel, and RampRate properties represent the *Dimmable* capability. The physical device controllers themselves are represented under the Devices node.

The Devices Node

GHA has a standard Application Programming Interface (API) which is used implement device drivers. The drivers delivered with GHA or those developed by third parties, leverage this API and **Capabilities** to implement a consistent control mechanism between Home level devices and physical devices. The Devices node contains the drivers configured for a GHA system. The drivers themselves are represented as GHA Objects under the Devices node. And, like all GHA Object, device driver nodes can also contain children. Figure (3) shows object hierarchy

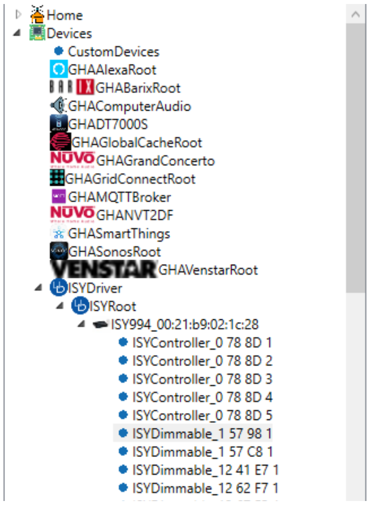
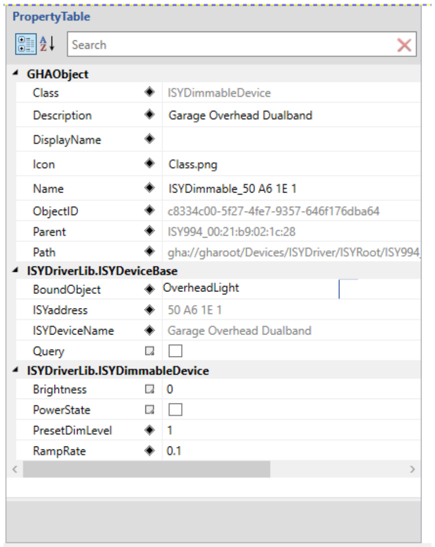


Figure 3 Devices Node

associated with the Universal Devices ISY driver including Its child nodes. Figure (4) shows the properties associated with

one of the children of the ISY device. Consistent with its name, this object implements control over a dimmer. You can see in the property list, the standard GHA Object properties as well as properties the implement the *PowerState* and *Dimmable* **Capabilities.** These are the same **Capabilities** that were used to define the properties for the Home level device. In this case, however, the properties that implement the desired **Capabilities** have code associated with them to command the ISY to set the state of the physical dimmer.



GHAObject		
Class	◆	ISYDimmableDevice
Description	◆	Garage Overhead Dualband
DisplayName	◆	
Icon	◆	Class.png
Name	◆	ISYDimmable_50 A6 1E 1
ObjectID	◆	c8334c00-5f27-4fe7-9357-646f176dba64
Parent	◆	ISY994_00:21:b9:02:1c:28
Path	◆	gha://gharoot/Devices/ISYDriver/ISYRoot/ISY994
ISYDriverLib.ISYDeviceBase		
BoundObject	◆	OverheadLight
ISYaddress	◆	50 A6 1E 1
ISYDeviceName	◆	Garage Overhead Dualband
Query	☐	☐
ISYDriverLib.ISYDimmableDevice		
Brightness	☐	0
PowerState	☐	☐
PresetDimLevel	◆	1
RampRate	◆	0.1

Figure 4 ISY Object properties

The Binding

A key concept to understand in GHA is that logical devices in the Home hierarchy are connected to device objects under the Devices hierarchy. We have this level of abstraction so that multiple physical device technologies can be consistently mapped to Home hierarchy objects without the user worrying about the differences between say Insteon and SmartThings. Another benefit of this approach is to allow for the replacing of underlying device technologies without impacting any of the objects under the Home hierarchy and associated automations. The connection between the logical representation of devices under the Home hierarchy and the device driver representation under the Devices node is called a **Binding**. During a **Binding** operation, logical devices search for physical device controllers with matching **Capabilities**. The user selects the appropriate device, and the two objects are bound together. A property changed in one, will be reflected in the other.

Scenes

A Scene in GHA is a mechanism where multiple objects can be controlled collectively. Let's say you want to define a Scene called "ChristmasDecorations" where you want to control yard lights connected to your Landscape lighting and your Christmas tree together. In GHA, this is simply done by creating a Scene object under the Scenes node, adding the objects to control, setting the desired property values for those objects and deciding on how you want the scene to behave. The key concept to understand about Scenes is the Scene behavior. In GHA, scenes can be configured to set the desired object properties when activated and leave them that way even when the scene is deactivated. This is called Set behavior. Alternatively, the scene can be configured to set the desired properties when the scene is activated and restore them to their previous state when deactivated. This is called Set-Restore behavior.

Schedules

Schedules in GHA can be used to activate a Scene or execute a script at a given date and time. The actions can be taken just once or repeated. In addition to simple date/time triggers, GHA can schedule actions for local sunrise and/or sunset. The Schedule Node in the hierarchy is where important information regarding your installation is provided such as: Street Address, Zipcode and/or Latitude/Longitude. The address is critically important if you choose to utilize the provided Noonlight

Monitoring service driver. Further, the Zipcode and/or Latitude/Longitude are required to calculate local sunrise/sunset times used elsewhere in GHA.

Customizations

GHA can be customized in several ways. First, all objects in the Home part of the hierarchy can have a custom script attached to a change of any of its property values. For example, all locations implement the *Occupancy* **Capability** which includes a property called Occupied. When Occupied is true, the location is considered occupied. When false, the location is considered empty. A property change script can be associated with the Occupied property and it will be invoked when the Occupied property changes. The script can then perform custom actions based on the occupancy of the location.

The second method for customization is the Module. There is a Modules node in the hierarchy which contains user-developed modules that can be used to implement highly sophisticated customizations. The modules themselves contain custom classes which follow the same object-oriented methodology that built-in classes follow. As such, they can contain properties and methods as well as support inheritance. Once a module has been created, it is compiled and made available to the rest of GHA. Classes within modules implicitly conform to the GHA Object specification and can be used anywhere a built-in GHA Object can be used.

The final way to do customizations is through building custom device drivers. GHA exposes an open API that can be leveraged by developers to implement their own device drivers.

Security Alarm Controller

GHA Offers a built-in security alarm controller that can bind with a device driver implementing the *AlarmControl* **Capability**. The security alarm controller offers the features you would expect: Arming/Disarming, delayed trigger, pass-code disarm, panic button, etc. The various user interfaces shipped with GHA (Windows, Android, and iOS) have a specialized interface for the alarm.   The goal is to replace classic alarm panels with one that can also control your home.

It is important that the bound device fully implement the *AlarmControl* **Capability**. Doing so will provide features like siren support and detection of various alert conditions such as sensor contact, fire, smoke, carbon monoxide, etc. The GHA security alarm controller can also work with a third-party security alarm monitoring service.

The monitoring service must have an associated driver that implements the *AlarmServiceProvider* **Capability**. We've partnered with Noonlight™ to provide 24/7 professional monitoring as a separately purchased feature. A compatible driver ships with GHA. When properly configured, GHA can automatically contact a UL certified, professionally staffed, US based monitoring center whenever a security or smoke/fire emergency is detected. Noonlight agents can dispatch first responders to your home within minutes of an alarm in all 50 US states. For more information and to purchase a subscription, go to the GHA website (https://www.george-home.com).

Speech Recognition Framework

GHA implements a framework that allows speech recognition services (like Amazon Alexa) to have a consistent method to interact with GHA.   Two things are required to utilize the framework. First, a device driver that implements the *ISpeechRecognition* **Capability** (like the built-in Alexa device driver). Second, GHASpeechPortal objects located in your hierarchy where your physical speech recognition devices exist. The GHASpeechPortal objects must be bound to devices that implement the *SpeechRecognition* **Capability**.   The idea is to have the Speech Recognition Framework default to controlling objects close to where the speech recognition device is located. For example, to control lighting using the Amazon Alexa service, a user can say "*Ask George Home to turn on lights*".   The GHA Speech Recognition Framework will recognize that as a generic lighting command and attempt to find lights at the same level in the Home hierarchy as the physical Amazon Echo device that received the command. If it finds them, it will turn them on. Alternatively, a user can say "Ask George Home to turn on lights in the living room". In this case, the GHA Speech Recognition Framework will look for a location called Living Room and turn on any lights found there. GHA also provides for a GHADefaultSpeechPortal that can be used without performing any bindings to physical devices. This is intended to be used for very simple implementations. One thing to note about searching for location names, the framework will use the DisplayName property if it is defined in its search.

The GHA Speech Recognition Framework supports the following control/query capabilities:

- Lighting control
- HVAC query
- Media control (media library and playlists)
- Scene control
- Garage query

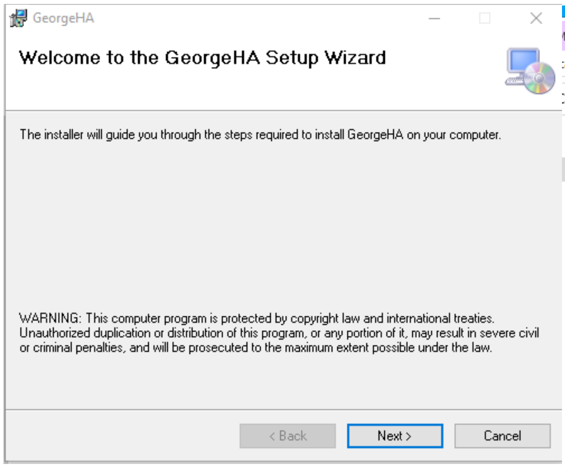
Getting Started

Installing GHA Server

GHA has two installable components. The first is the GHA Server itself. This installs as a Windows console application but can optionally be configured to run as a service. The media containing the GHA server has a program called setup.exe. Double-click on setup.exe to start the installation.

GHA requires .NET 7.0 to be running on the computer running the server. If .NET 7.0 is not available, the installation will fail. The required .NET Runtimes that are required are: Windows Hosting Bundle and Windows Desktop Runtime

The stand-alone .NET Runtime is bundled with the Desktop Runtime and should not be separately installed.

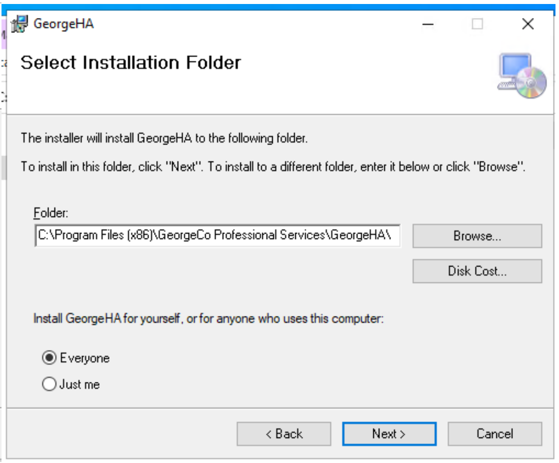


GeorgeHA

Welcome to the GeorgeHA Setup Wizard

The installer will guide you through the steps required to install GeorgeHA on your computer.

WARNING: This computer program is protected by copyright law and international treaties. Unauthorized duplication or distribution of this program, or any portion of it, may result in severe civil or criminal penalties, and will be prosecuted to the maximum extent possible under the law.



GeorgeHA

Select Installation Folder

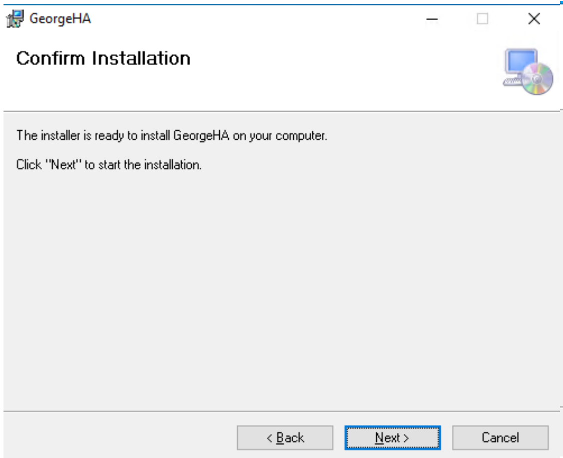
The installer will install GeorgeHA to the following folder.

To install in this folder, click "Next". To install to a different folder, enter it below or click "Browse".

Folder:
C:\Program Files (x86)\GeorgeCo Professional Services\GeorgeHA\

Install GeorgeHA for yourself, or for anyone who uses this computer:

● Everyone
○ Just me

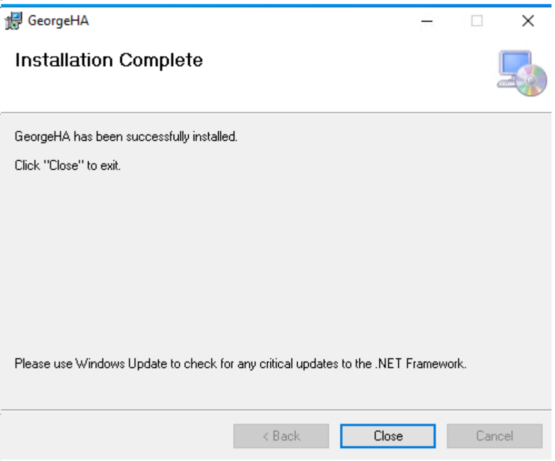


GeorgeHA

Confirm Installation

The installer is ready to install GeorgeHA on your computer.

Click "Next" to start the installation.



GeorgeHA

Installation Complete

GeorgeHA has been successfully installed.

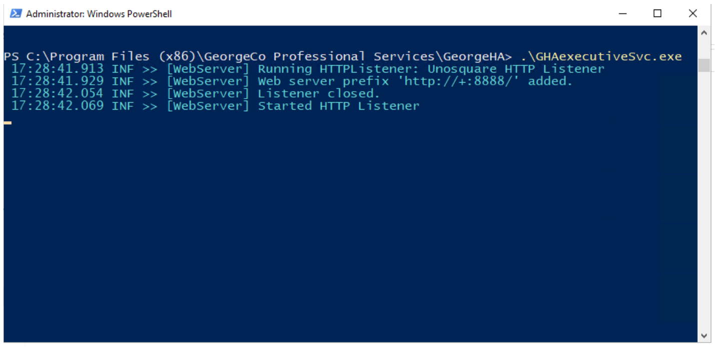
Click "Close" to exit.

Please use Windows Update to check for any critical updates to the .NET Framework.

A number of directories will be created in the installation directory. Of note is GHADevices. The GHADevices directory will contain the driver libraries that will be used with this installation of GHA Server. Driver dll's may be loaded via the GHAConfigurator under File>Devices… .

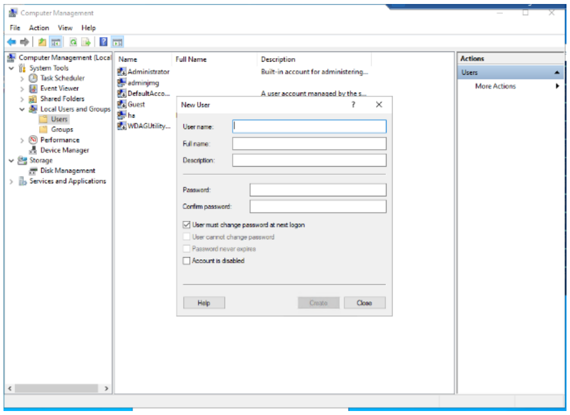
Open a PowerShell window and navigate to the installation directory. Type the following into the window: **.\GHAWorkerService.exe**

If the installation was successful, you should see a PowerShell window that looks like Figure(xx). You can gracefully terminate the GHA Server by hitting the Enter key while the PowerShell window is in focus.



```
PS C:\Program Files (x86)\GeorgeCo Professional Services\GeorgeHA> .\GHAexecutiveSvc.exe
 17:28:41.913 INF >> [WebServer] Running HTTPListener: Unosquare HTTP Listener
 17:28:41.929 INF >> [WebServer] Web server prefix 'http://+:8888/' added.
 17:28:42.054 INF >> [WebServer] Listener closed.
 17:28:42.069 INF >> [WebServer] Started HTTP Listener
```

GHA Server requires a username and password to allow access. To create the user name and password, launch the Windows Computer Manager application. Open the Local Users and Groups node and select "Users". Right-click on the Users node and select New User… You will be presented with a dialog to create a Window user account. Please provide the appropriate information and **uncheck** the block next to the words "User must change password at next logon". Click on Create. Your GHA User account is now ready for use.



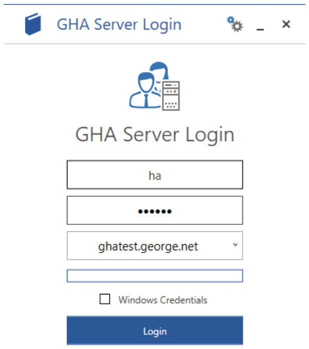
☑ User must change password at next logon
□ User cannot change password
□ Password never expires
□ Account is disabled

Optional step: Install GHA Server as a Windows service. In the GHA Server installation directory you will find two PowerShell scripts: InstallService and UninstallService. To install the GHA Sever as a Windows Service (which will automatically start the GHA Server every time the computer restarts), enter the following command in an elevated PowerShell with the default directory set to the GHA Server installation directory: **.\InstallService.ps1**

If you are upgrading from version 1.1 to version 1.2 and installed GHA as a Windows Service, you will see two services in the Computer Manager Services window called GeorgeHA Executive. You should run the PowerShell file **UninstallV1.1Service.ps1** to remove the V1.1 service.

Install GHA Configurator

GHA Configurator is the application that will be used to configure and manage the GHA Server. Double-click the setup icon from the GHA Configurator installation location. When the installer completes, GHA Configurator will launch and present a login screen. Type in the user name and password you created while installing the GHA Server. The text box below the password should contain the name of the GHA Server. If you do not see your server name there, click on the caret and see if it appears in the list. If it is still not visible, type in the name of the server manually. You can also use **localhost** if you are running GHA Configurator on the same computer as the GHA Server.



ha

••••••

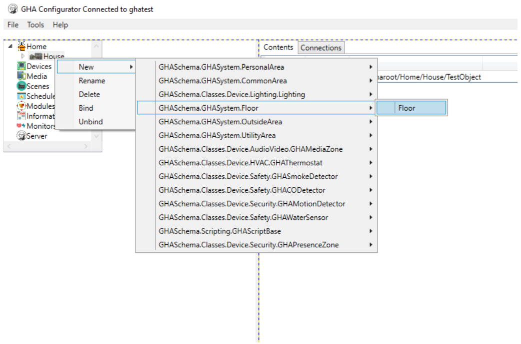
ghatest.george.net

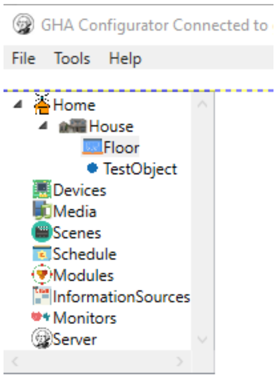
□ Windows Credentials

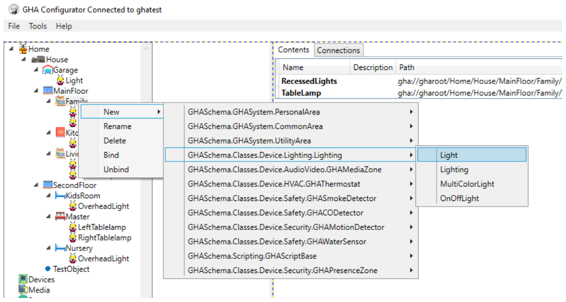
If the installation is successful, the GHA Configurator application will launch and show the default initial hierarchy in the left hand pane. At this point, GHA is up and running and ready for configuration.

Basic GHA configuration

For this basic configuration, we will be creating a two-story home with a Main floor and a Second floor. Right-clicking on the "House" node in the hierarchy will present a set of classes that can be children of the House node. Click on the "GHASchema.GHASystem.Floor" menu item to reveal the GHA class for Floor. As shown in Figure (xx), click on "Floor" to create an instance of the Floor class under House. A new object named "Floor" will appear under House in the hierarchy as shown in Figure (xx). Of course, we want this object to be named "Main Floor", so we need to right-click on the object and select "Rename". The Rename window will appear asking for the new name for the object. Type in "MainFloor" (spaces are not supported), and the object will be renamed to the desired name. Repeat the above steps to create a "SecondFloor" object under House in the hierarchy. Following the same procedure, we will add several rooms to the hierarchy to complete our two-story house. Next up is adding devices.


GHA Configurator Connected to ghatest
File
Tools
Help
Home
House
Devices
Media
Scenes
Schedule
Modules
Informat
Monitors
Server
New
Rename
Delete
Bind
Unbind
GHASchema.GHASystem.PersonalArea
GHASchema.GHASystem.CommonArea
GHASchema.Classes.Device.Lighting.Lighting
GHASchema.GHASystem.Floor
GHASchema.GHASystem.OutsideArea
GHASchema.GHASystem.UtilityArea
GHASchema.Classes.Device.AudioVideo.GHAMediaZone
GHASchema.Classes.Device.HVAC.GHAThermostat
GHASchema.Classes.Device.Safety.GHASmokeDetector
GHASchema.Classes.Device.Safety.GHACODetector
GHASchema.Classes.Device.Security.GHAMotionDetector
GHASchema.Classes.Device.Safety.GHAWaterSensor
GHASchema.Scripting.GHAScriptBase
GHASchema.Classes.Device.Security.GHAPresenceZone
Floor
Contents
Connections
aroot/Home/House/TestObject

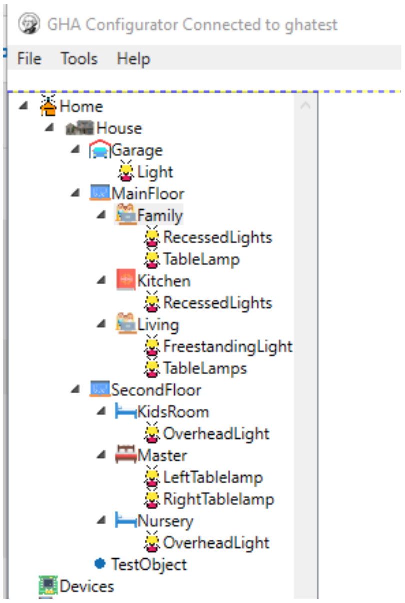



GHA Configurator Connected to ghatest
File Tools Help
Home
House
Garage
Light
MainFloor
Family
Kitc
Livi
SecondFloor
KidsRoom
OverheadLight
Master
LeftTablelamp
RightTablelamp
Nursery
OverheadLight
TestObject
Devices
Media
New
Rename
Delete
Bind
Unbind
GHASchema.GHASystem.PersonalArea
GHASchema.GHASystem.CommonArea
GHASchema.GHASystem.UtilityArea
GHASchema.Classes.Device.Lighting.Lighting
GHASchema.Classes.Device.AudioVideo.GHAMediaZone
GHASchema.Classes.Device.HVAC.GHAThermostat
GHASchema.Classes.Device.Safety.GHASmokeDetector
GHASchema.Classes.Device.Safety.GHACODetector
GHASchema.Classes.Device.Security.GHAMotionDetector
GHASchema.Classes.Device.Safety.GHAWaterSensor
GHASchema.Scripting.GHAScriptBase
GHASchema.Classes.Device.Security.GHAPresenceZone
Light
Lighting
MultiColorLight
OnOffLight
Contents
Connections

Name	Description	Path
RecessedLights		gha://gharoot/Home/House/MainFloor/Family/
TableLamp		gha://gharoot/Home/House/MainFloor/Family/

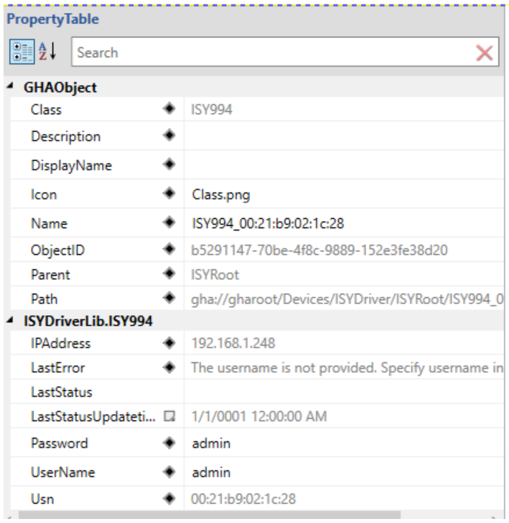
For our basic configuration, we will be focused on lighting. Using the same process as adding locations and rooms under the House hierarchy, we can add objects that represent lights. Right click on a location object, select *New > GHASChema.Classes.Device.Lighting.Lighting* to reveal the GHA classes that represent Lights. Select the "Light" class as shown in Figure (XX). The Light class in GHA contains the definitions for the *PowerState* and *Dimmable* Capabilities. If you are looking for a lighting class that simply implements on/off functionality, you can select the OnOffLight class which implements only the *PowerState* **Capability**.

Repeat the process for the lighting you wish to add to the various rooms under your House hierarchy. Once you are finished, you should see a hierarchy like Figure(xx). At this point we have added lighting objects, but we have not configured GHA to control physical devices. To do this, we need to add device drivers under the Devices node.



Our basic configuration will include a Universal Devices ISY device. There is a built-in GHA device driver that supports the ISY. To install it, right-click on the Devices node in the hierarchy, and *select New > GHASchema.IGHADevice > ISYDriver*. The ISY driver will load and discover any ISY devices on your network.

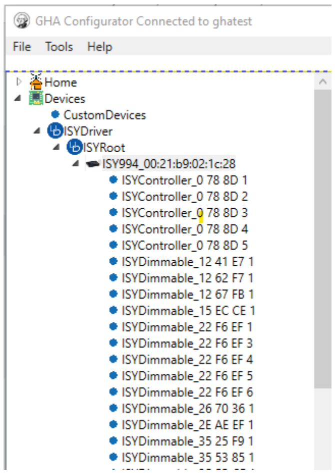
Select an ISY device under the ISYRoot node to show the properties associated with the ISY device. You'll notice an error in the *LastError* property that says, "The username is not provided. Specify a username". Type in the username and password you have configured for your ISY device. Figure(XX) shows the properties associated with the ISY.



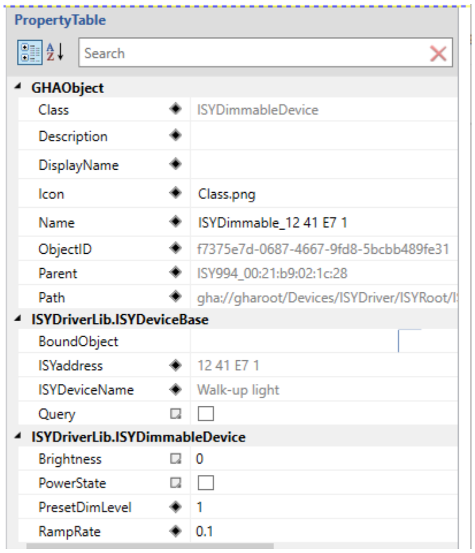
GHAObject	
Class	ISY994
Description	
DisplayName	
Icon	Class.png
Name	ISY994_00:21:b9:02:1c:28
ObjectID	b5291147-70be-4f8c-9889-152e3fe38d20
Parent	ISYRoot
Path	gha://gharoot/Devices/ISYDriver/ISYRoot/ISY994_0
ISYDriverLib.ISY994	
IPAddress	192.168.1.248
LastError	The username is not provided. Specify username in
LastStatus	
LastStatusUpdateti...	1/1/0001 12:00:00 AM
Password	admin
UserName	admin
Usn	00:21:b9:02:1c:28

The ISY driver should start automatically detecting and populating the devices and groups configured on your ISY. If you do not see devices populating within a few minutes, you may need to restart your GHA Server.

Once the GHA ISY Driver populates devices under your ISY device node, you should see something like Figure (XX). Please note that the devices are given names that relate to the type of physical ISY-compatible device that was detected. Devices that support dimming have a prefix starting with ISYDimmable. ISY groups start with ISYGroup. The combination of letters and numbers after the prefix are the unique identifiers that the ISY device uses to access the physical device. For our basic configuration, we'll focus on the devices starting with ISYDimmable. In GHA, these devices implement the *PowerState* and *Dimmable* **Capabilities**.



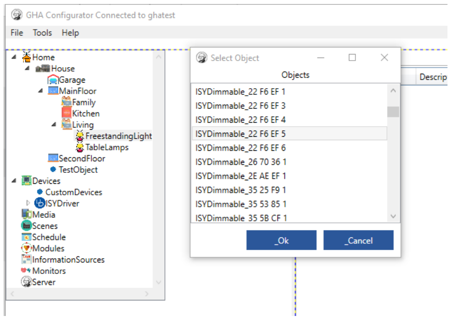
Highlight one of the ISYDimmable objects and you will see the properties associated with *PowerState* and *Dimmable* **Capabilities**. You will also see properties that unique to the ISY device object itself. Remember when we discussed how GHA is object-oriented and takes advantage of inheritance? This is inheritance in action. Within GHA, there is an ISYDeviceBase class that contains the properties that are required to manage an ISY-connected physical device. The ISYDimmableDevice class inherits from this base class to take on all of the control capabilities that are built into the ISY driver for ISY physical devices. The ISYDimmableDevice class implements the required software to control the dimming functions. As you get into building your own device drivers and modules, you can take advantage of the base capabilities offered in GHA classes to make your own custom classes. Note the ISY device has a property called ISYDeviceName. This name is set in the ISY Admin Console (see https://www.universal-devices.com for information on using the admin console). It is recommended that this name be something meaningful to allow you to easily associate the GHA ISY device object with the physical device being controlled.



GHAObject		
Class	◆	ISYDimmableDevice
Description	◆	
DisplayName	◆	
Icon	◆	Class.png
Name	◆	ISYDimmable_12 41 E7 1
ObjectID	◆	f7375e7d-0687-4667-9fd8-5bcbb489fe31
Parent	◆	ISY994_00:21:b9:02:1c:28
Path	◆	gha://gharoot/Devices/ISYDriver/ISYRoot/I
ISYDriverLib.ISYDeviceBase		
BoundObject		
ISYaddress	◆	12 41 E7 1
ISYDeviceName	◆	Walk-up light
Query	□	□
ISYDriverLib.ISYDimmableDevice		
Brightness	□	0
PowerState	□	□
PresetDimLevel	◆	1
RampRate	◆	0.1

We now have the pieces in-place to enable control from the GHA objects under our Home hierarchy. We just need to connect the objects under the Home hierarchy to their corresponding objects under the Devices hierarchy. We do this through a **Binding**.

As described under key concepts, the Binding maps the **Capabilities** associated with objects created under the Home hierarchy with the same **Capabilities** defined for device objects created under the Devices hierarchy. For example, the Light class used as the basis for a Table Lamp defined in the Home hierarchy in our basic configuration above contains properties defined in the *PowerState* and *Dimmable* **Capabilities**. The object ISYDimmable_12 41 E7 1 defined under the Devices hierarchy contains the <u>implementation</u> code for *PowerState* and *Dimmable* **Capabilities**. The two objects can be bound together. To Bind an object under the Home hierarchy with an object under the Devices hierarchy, right click the object under the Home hierarchy and select Bind. A window containing compatible objects under the Device hierarchy will be displayed. Select the object you wish to bind and click on Ok. The two objects are now bound. Changes to properties in one object will be reflected in the other.

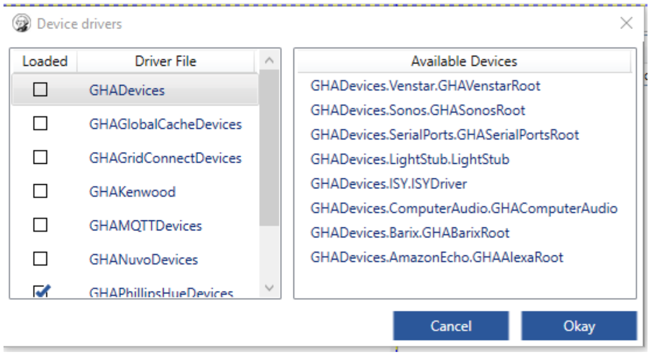


Built-in Devices

GHA comes with several built-in device drivers. This section will describe these and discuss any driver-specific configuration topics.

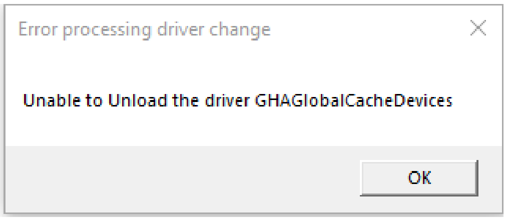
Configuring Devices for use

Device drivers may be selected for use by using the GHAConfigurator Device drivers dialog available at File>Devices… file menu. When selected, a dialog like Figure (XX) will be shown. By highlighting an item in the Driver File column, the dialog will show the GHA compatible devices available in the selected file. The checkbox in the "Loaded" column indicates that the driver file is already loaded, and its devices are available for use. To load a driver file, simply check its "Loaded" checkbox and click the "Okay" button. The driver will be loaded, and its devices will be available under the Devices>New node in GHAConfigurator. Conversely, it is possible to unload a driver by de-selecting its "Loaded" checkbox. However, if devices within the file are in-use by GHA, you will receive a prompt similar to Figure (XX) indicating that GHA will be unable to unload the driver until you delete the appropriate devices from the hierarchy. While loading device drivers is a dynamic function, unloading is not. After deselecting the "Loaded" checkbox associated with the driver you wish to unload, you will see a message like Figure (XX). You must restart the GHA Server for the driver unload to be finalized. Please wait a few minutes before restarting to allow the GHA Server to complete final reconfiguration steps.

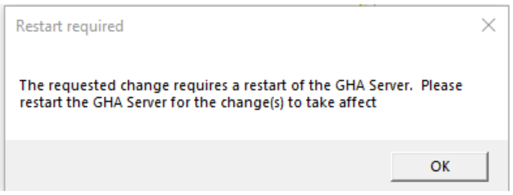

Device drivers

Loaded	Driver File
□	GHADevices
□	GHAGlobalCacheDevices
□	GHAGridConnectDevices
□	GHAKenwood
□	GHAMQTTDevices
□	GHANuvoDevices
☑	GHAPhillipsHueDevices

Available Devices
GHADevices.Venstar.GHAVenstarRoot
GHADevices.Sonos.GHASonosRoot
GHADevices.SerialPorts.GHASerialPortsRoot
GHADevices.LightStub.LightStub
GHADevices.ISY.ISYDriver
GHADevices.ComputerAudio.GHAComputerAudio
GHADevices.Barix.GHABarixRoot
GHADevices.AmazonEcho.GHAAlexaRoot


Error processing driver change

Unable to Unload the driver GHAGlobalCacheDevices


Restart required

The requested change requires a restart of the GHA Server. Please restart the GHA Server for the change(s) to take affect

ZWave Devices

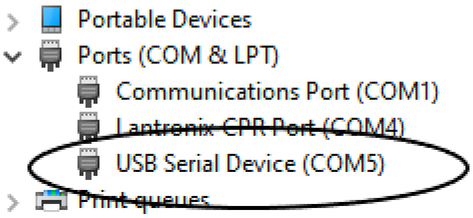
GHA supports Zwave devices through a USB Controller. While most ZWave USB controllers are based on the same underlying chip-set provided by Silicon Labs, we recommend that the AEOTEC Z-Stick Gen5+ be used. Theoretically, any ZWave USB device built with Silicon Labs 500 or 700 series chips should work.

GHA Implements ZWave via the GHAZWaveDevices driver. Once the driver is enabled, created a GHAZWaveDeviceRoot under the Devices node. Under the GHAZWaveDeviceRoot, create a GHAZWaveController. The driver does not currently automatically detect ZWave controllers that may be attached to your GHA Server. You'll need to determine the COM port that your USB ZWave controller is assigned by looking at Windows Device Manager.   An example Device Manager window showing the USB Controller and associated COM port is shown in the figure. Note the COM port as that will need to be provided to the ZwaveDevice property of the GHAZwaveController object in your Configurator as shown in the figure.

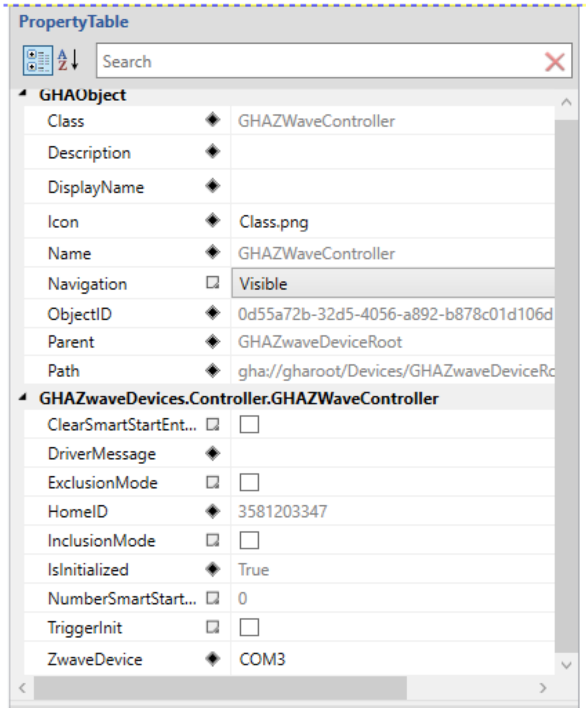
Once the ZwaveDevice property is set to the COM port associated with your USB ZWave controller, the GHAZwaveController IsInitialized property should be "True". If the property does not show as "True" after a few minutes, please restart your GHA Server.

If your controller had previously associated ZWave devices, they will appear under the GHAZWaveController node. The GHA Zwave driver supports the following device types:

- On/Off Switch
- Dimmable device
- Wall Controller
- Scene Switch
- Notification Sensor



Portable Devices
Ports (COM & LPT)
Communications Port (COM1)
Lantronix CPR Port (COM4)
USB Serial Device (COM5)
Print queues



PropertyTable

Search

GHAObject

Class	GHAZWaveController
Description	
DisplayName	
Icon	Class.png
Name	GHAZWaveController
Navigation	Visible
ObjectID	0d55a72b-32d5-4056-a892-b878c01d106d
Parent	GHAZwaveDeviceRoot
Path	gha://gharoot/Devices/GHAZwaveDeviceRc

GHAZwaveDevices.Controller.GHAZWaveController

ClearSmartStartEnt...	☐
DriverMessage	
ExclusionMode	☐
HomeID	3581203347
InclusionMode	☐
IsInitialized	True
NumberSmartStart...	0
TriggerInit	☐
ZwaveDevice	COM3

Adding new devices to your ZWave can be done in multiple ways. First, many USB ZWave controllers allow you to include devices simply by pushing a button on the controller itself. This approach allows you to move around your home with the ZWave controller and perform the inclusion steps as outline in your controller's documentation. Once you've completed this process, you may plug your ZWave USB controller back into your GHA Server, restart GHA, and your new, supported devices will appear under the GHAZWaveController node.

The second way to include devices into your ZWave network is to use the "Inclusion mode" from within GHA. Most ZWave devices need to be placed into Inclusion mode before the ZWave controller. Follow the instructions provided by your ZWave device manufacturer. In most cases the process is as follows:

1. Place your device in Inclusion mode. Usually, by pressing and holding a button on the device itself until it flashes.
2. In the Configurator, check the GHAZWaveController InclusionMode property.
3. Back on the device, press the button again. Most devices will flash a green LED indicating that the inclusion to the ZWave network was successful. A red LED will flash if inclusion was not successful.

The GHA ZWave driver will interrogate the device looking for supported capabilities. If the device is supported, it will be added under the GHAZWaveController node in the Configurator.

Removing devices from your ZWave network follows a similar process to inclusion. Again, the manual method using the button on a controller can be used (follow the instructions on your controller and device), or you can use the built-in capability provided by the GHA ZWave driver.

The GHA ZWave driver method involves checking the GHAZWaveController Exclusion mode property after placing the device into exclusion mode. Follow the directions provided by your device manufacturer. Once the device is excluded, it will be removed from the GHA Hierarchy.

Please make sure that the GHA ZWave device driver is not in Inclusion or Exclusion mode for long periods of time.

Additionally, the GHA Zwave driver can "cast" certain devices into other types. For example, there are ZWave motion sensors (like the Zooz ZSE40) that also include a variety of other sensors (e.g. Temperature sensor) that wouldn't normally be surfaced as a separate device. In these cases, you can manually add the "Casted" devices, set the Node ID of the casted device to the device that contains other sensors, and the driver will automatically replace the existing device and attempt to detect any additional sensors that the device supports. Currently, the casted motion sensor is supported. Other devices may be added in the future.

The GHA Zwave driver also supports grouping devices so that they may be controlled as a single device. Device groups are defined by setting the MemberNodeIDs property of a group with a comma separated list of node IDs for the devices that are to be grouped. Two group types are currently supported:

- Lighting Group (contains dimmable devices)
- Relay Group (contains On/Off switch devices)

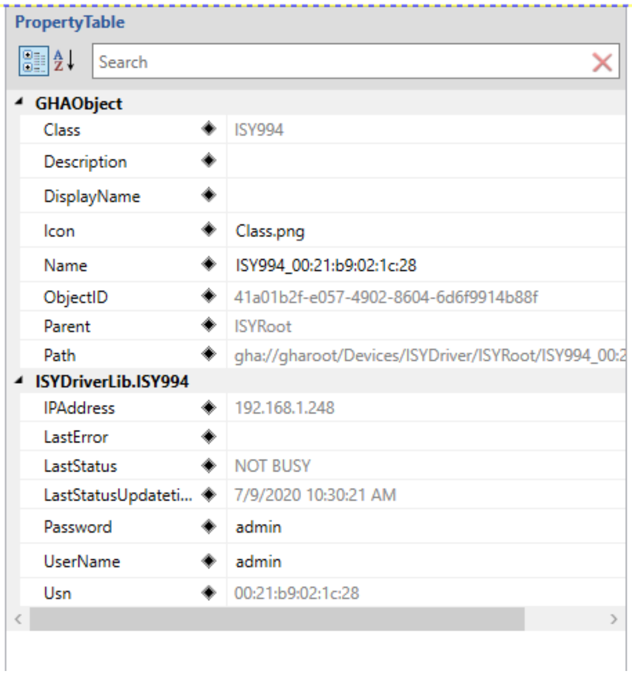
Universal Devices ISY

The Universal Devices ISY-994 device provides control over a variety of devices including those that support Insteon. The GHA ISY driver exposes the various ISY Insteon device models in a way that can be used by the GHA System. The current ISY supported version is 4.7.5. Other versions may work but have not been tested. The first step in using the device driver is to add it to your GHA Devices hierarchy. Right click on Devices and select the ISYDriver. Once the driver is installed, it will automatically detect ISY devices on the network attached to the GHA server. Once detected, the ISY device will appear under the Devices>ISYDriver>ISYRoot. If ISY Devices are not detected within five minutes, you may need to restart the GHA Executive Service.

The properties exposed by the ISY driver to GHA are showing in Figure(XX). Only two properties are required to be configured for the ISY driver: UserName and Password. Enter them in the properties field to allow GHA to access the ISY Device. Once authenticated, the ISY driver will populate GHA with the Insteon devices known to the ISY device. These devices will appear under the ISY Device in the hierarchy.

The GHA ISY driver supports the following Insteon device types:

- ControlLinc
- Health and Safety
- Dimmer
- Relay switch
- Groups
- Sensor/Actuators

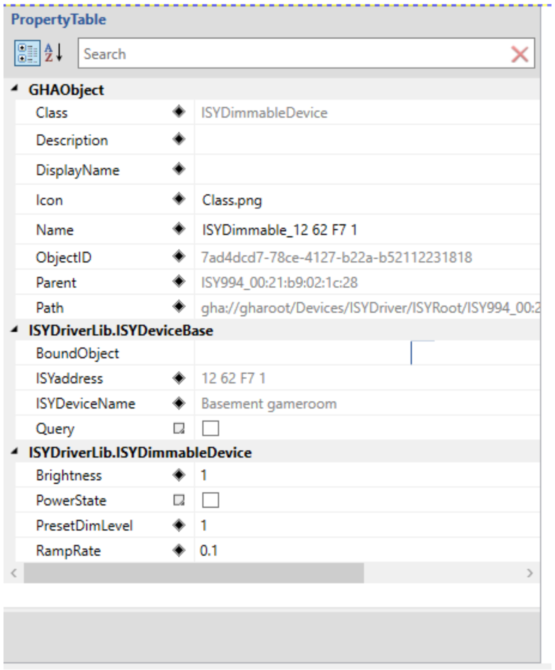


PropertyTable

Search

GHAObject		
Class	◆	ISY994
Description	◆	
DisplayName	◆	
Icon	◆	Class.png
Name	◆	ISY994_00:21:b9:02:1c:28
ObjectID	◆	41a01b2f-e057-4902-8604-6d6f9914b88f
Parent	◆	ISYRoot
Path	◆	gha://gharoot/Devices/ISYDriver/ISYRoot/ISY994_00:2
ISYDriverLib.ISY994		
IPAddress	◆	192.168.1.248
LastError	◆	
LastStatus	◆	NOT BUSY
LastStatusUpdateti...	◆	7/9/2020 10:30:21 AM
Password	◆	admin
UserName	◆	admin
Usn	◆	00:21:b9:02:1c:28

Each device type exposes base Insteon properties and device-unique properties. For example, the ISYDimmable device (Insteon Dimmer) exposes the base Insteon properties of ISYAddress and ISYDeviceName. Additionally, it supports the dimmer-specific properties of Brightness, PowerState, PresetDimLevel, and RampRate. Please see Figure (XX). If the dimmer-specific properties sound familiar, they should. The ISYDimmableDevice class implements the *IPowerState* and *IDimmable* **Capabilities**.



PropertyTable

Search

GHAObject		
Class	◆	ISYDimmableDevice
Description	◆	
DisplayName	◆	
Icon	◆	Class.png
Name	◆	ISYDimmable_12 62 F7 1
ObjectID	◆	7ad4dcd7-78ce-4127-b22a-b52112231818
Parent	◆	ISY994_00:21:b9:02:1c:28
Path	◆	gha://gharoot/Devices/ISYDriver/ISYRoot/ISY994_00:2
ISYDriverLib.ISYDeviceBase		
BoundObject		
ISYaddress	◆	12 62 F7 1
ISYDeviceName	◆	Basement gameroom
Query		□
ISYDriverLib.ISYDimmableDevice		
Brightness	◆	1
PowerState		□
PresetDimLevel	◆	1
RampRate	◆	0.1

Sonos

The Sonos driver allows GHA to control Sonos media players. The driver supports playing of MP3, WAV, and Sonos favorites with control initiated from within GHA's media handling services. Adding the Sonos driver can be accomplished by right-clicking on Devices and selecting New>IGHADevices>GHASonos Root. Once the driver is loaded, it searches for Sonos devices present on your home network. Discovered devices appear under the GHASonosRoot. Each GHA Sonos device implements the *IMediaPlayer* **Capability** allowing it to be bound to a GHA MediaZone under the Home hierarchy. The driver also creates and populates the SonosFavorites root under Media>Content in the GHA hierarchy.

Computer Audio

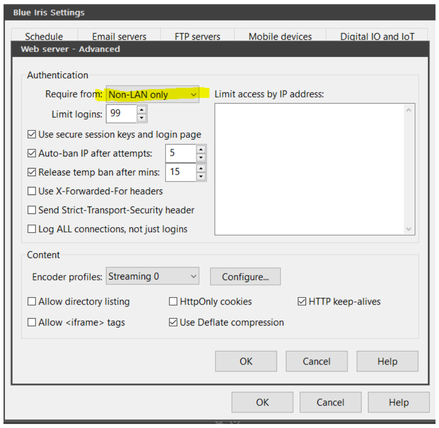
The Computer Audio driver detects and makes available to GHA the sound devices that exist on the computer running the GHA server. Once created, the GHA driver searches for sound cards and makes them available under the Devices>GHAComputerAudio root. The driver also can manually initiate detection of computer audio devices. The GHA driver implements the *IMediaPlayer* **Capability** for detected devices allowing them to be bound to GHA MediaZones under the Home hierarchy.

Barix Exstreamer

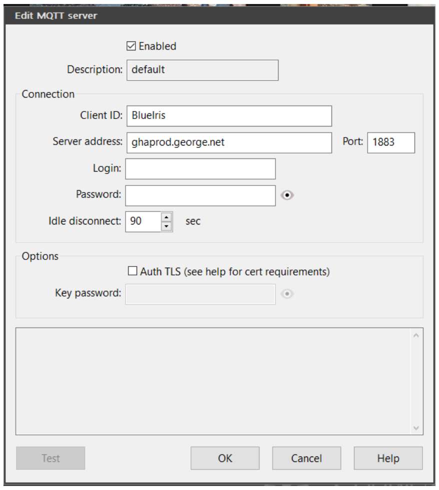
The Barix Exstreamer is purpose-built MP3 media streaming device. It attaches to your home network and through the GHA Barix Exstreamer driver, can play GHA-managed media content. Once installed, the GHA Barix driver will automatically detect Exstreamers on your network and create Extreamer devices under the Devices>GHABarixRoot. The Exstreamer GHA devices implement the *IMediaPlayer* **Capability** for detected devices allowing them to be bound to GHA MediaZones under the Home hierarchy.

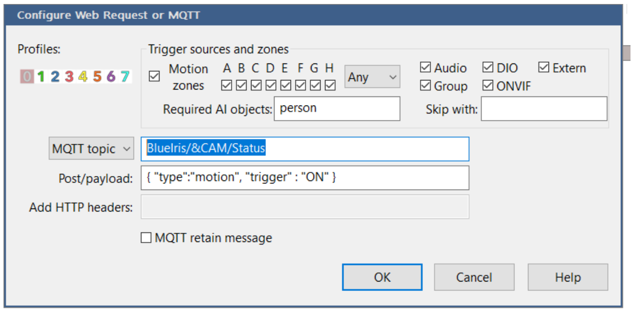
Blue Iris Webcam and Security Software

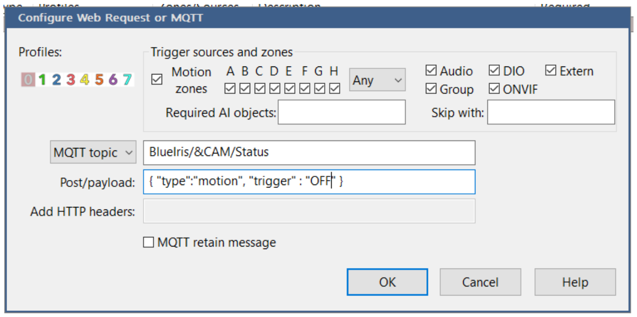
Blue Iris is a webcam and security software designed to monitor, record and manage IP-based and Analog cameras. Additionally, the GHA Blue Iris software provides a device that implements the *IMotionSensor* **Capability** allowing Blue Iris to report motion detected events to GHA. Configuring Blue Iris to work with GHA requires two steps. First, the Blue Iris software must be configured. Please note, this document assumes that you have a working knowledge of Blue Iris. Step by step instructions for Blue Iris will not be presented.


Blue Iris Settings
Web server - Advanced
Authentication
Require from: Non-LAN only
Limit access by IP address:
Limit logins: 99
☑ Use secure session keys and login page
☑ Auto-ban IP after attempts: 5
☑ Release temp ban after mins: 15
□ Use X-Forwarded-For headers
□ Send Strict-Transport-Security header
□ Log ALL connections, not just logins
Content
Encoder profiles: Streaming 0
□ Allow directory listing
□ HttpOnly cookies
☑ HTTP keep-alives
□ Allow <iframe> tags
☑ Use Deflate compression

These instructions assume that Blue Iris release 5.5.4.4 is being configured. There may be differences in other versions. Configure the Blue Iris Web server to only require login's from Non-LAN sources. Optionally, you can limit access to specific IP addresses. These settings are required to access the camera streams only. Access to configure Blue Iris will continue to require a Username and Password. To configure Blue Iris to report motion events to GHA, you need to enable the MQTT in the Digital IO and IoT settings tab. Set the MQTT server address to point to your GHA server and port 1833. GHA does not require authentication, so leave the Login and Password fields blank. Don't forget to check the Enabled checkbox. Next, each camera must be configured to send an MQTT alert and reset message to GHA. The alert and reset messages are shown here. Please make sure the MQTT topic and post/payload are copied EXACTLY as shown in the figure.

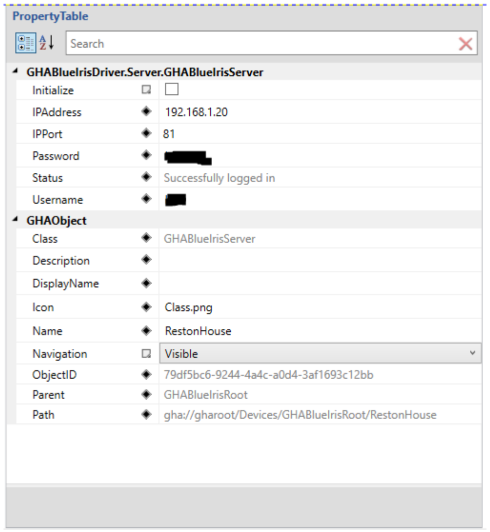

Edit MQTT server
☑ Enabled
Description: default
Connection
Client ID: BlueIris
Server address: ghaprod.george.net
Port: 1883
Login:
Password:
Idle disconnect: 90 sec
Options
□ Auth TLS (see help for cert requirements)
Key password:


Configure Web Request or MQTT
Profiles: 0 1 2 3 4 5 6 7
Trigger sources and zones
☑ Motion zones A B C D E F G H ☑ ☑ ☑ ☑ ☑ ☑ ☑ ☑ Any
☑ Audio ☑ DIO ☑ Extern
☑ Group ☑ ONVIF
Required AI objects: person
Skip with:
MQTT topic: BlueIris/&CAM/Status
Post/payload: { "type":"motion", "trigger" : "ON" }
Add HTTP headers:
□ MQTT retain message


Configure Web Request or MQTT
Profiles: 0 1 2 3 4 5 6 7
Trigger sources and zones
☑ Motion zones A B C D E F G H ☑ ☑ ☑ ☑ ☑ ☑ ☑ ☑ Any
☑ Audio ☑ DIO ☑ Extern
☑ Group ☑ ONVIF
Required AI objects:
Skip with:
MQTT topic: BlueIris/&CAM/Status
Post/payload: { "type":"motion", "trigger" : "OFF" }
Add HTTP headers:
□ MQTT retain message

Configuring the GHA server requires adding the GHABlueIrisRoot driver under the Devices Node. Blue Iris servers cannot be auto-detected, so you must manually add a Blue Iris server under the GHA BlueIrisRoot. Of course, you may change the default name to suit your needs. Once created, highlight the new device and examine the Property Browser window. Enter the IP address and port number for your Blue Iris server along with a Username and Password. Once configured, the GHA Blue Iris server Status property should show "Successfully logged in". The Blue Iris driver will then create camera and motion sensor objects under the Blue Iris server node you previously created. Highlight one of the cameras under the Blue Iris Server node. You will see a number of properties, most are read-only. However, there are two that control the Blue Iris' driver's behavior while creating GHA Stream objects under Media/Content. Yes, the Blue Iris driver will create a BlueIrisStreams node under Media/Content. These streams are standard GHA Streams and can be played on compatible devices (i.e. devices that support the MIME type of the stream). Google Chrome Cast devices are able to play the MJPG Streams, so configuring a Media Zone to include a television that supports Chrome Cast will give you the ability to play security camera streams right on your television!

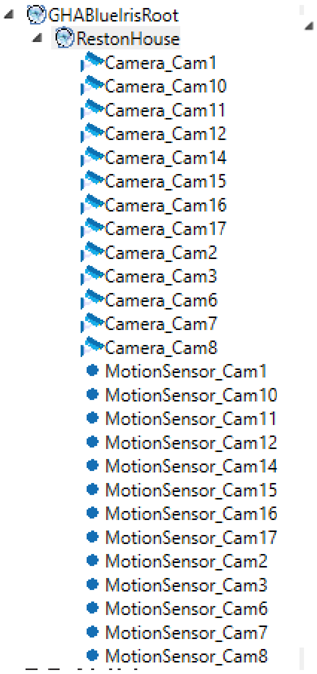
By default, both H.264 and MJPEG streams are created. The GHA Blue Iris camera object allows you to specify if you want either or both. The figure shows the result of both stream types being selected.



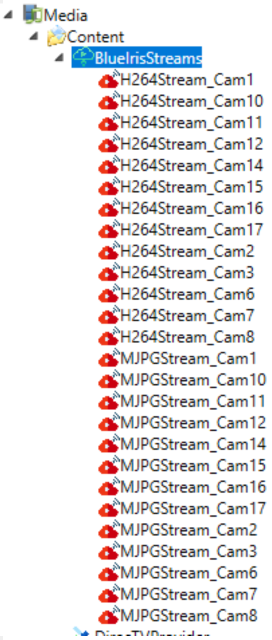
PropertyTable

Search

GHABlueIrisDriver.Server.GHABlueIrisServer	
Initialize	□
IPAddress	192.168.1.20
IPPort	81
Password	[redacted]
Status	Successfully logged in
Username	[redacted]
GHAObject	
Class	GHABlueIrisServer
Description	
DisplayName	
Icon	Class.png
Name	RestonHouse
Navigation	Visible
ObjectID	79df5bc6-9244-4a4c-a0d4-3af1693c12bb
Parent	GHABlueIrisRoot
Path	gha://gharoot/Devices/GHABlueIrisRoot/RestonHouse



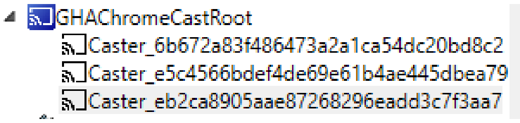
- GHABlueIrisRoot
  - RestonHouse
    - Camera_Cam1
    - Camera_Cam10
    - Camera_Cam11
    - Camera_Cam12
    - Camera_Cam14
    - Camera_Cam15
    - Camera_Cam16
    - Camera_Cam17
    - Camera_Cam2
    - Camera_Cam3
    - Camera_Cam6
    - Camera_Cam7
    - Camera_Cam8
    - MotionSensor_Cam1
    - MotionSensor_Cam10
    - MotionSensor_Cam11
    - MotionSensor_Cam12
    - MotionSensor_Cam14
    - MotionSensor_Cam15
    - MotionSensor_Cam16
    - MotionSensor_Cam17
    - MotionSensor_Cam2
    - MotionSensor_Cam3
    - MotionSensor_Cam6
    - MotionSensor_Cam7
    - MotionSensor_Cam8



- Media
  - Content
    - BlueIrisStreams
      - H264Stream_Cam1
      - H264Stream_Cam10
      - H264Stream_Cam11
      - H264Stream_Cam12
      - H264Stream_Cam14
      - H264Stream_Cam15
      - H264Stream_Cam16
      - H264Stream_Cam17
      - H264Stream_Cam2
      - H264Stream_Cam3
      - H264Stream_Cam6
      - H264Stream_Cam7
      - H264Stream_Cam8
      - MJPGStream_Cam1
      - MJPGStream_Cam10
      - MJPGStream_Cam11
      - MJPGStream_Cam12
      - MJPGStream_Cam14
      - MJPGStream_Cam15
      - MJPGStream_Cam16
      - MJPGStream_Cam17
      - MJPGStream_Cam2
      - MJPGStream_Cam3
      - MJPGStream_Cam6
      - MJPGStream_Cam7
      - MJPGStream_Cam8

Google Chrome Cast

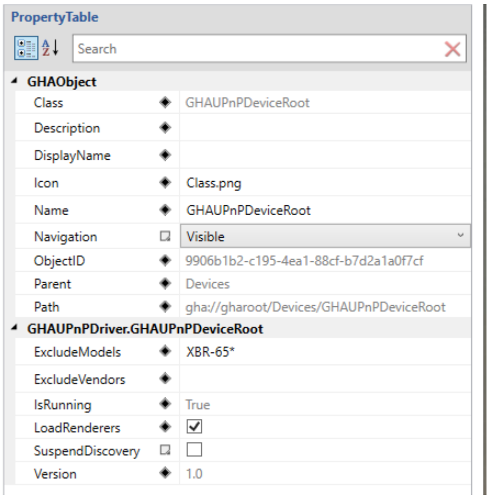
Chrome Cast is a media playing protocol implemented by many television manufacturers. The GHA Chrome Cast driver models Chrome Cast capable devices as a Media Player by implementing the *IMediaPlayer* **Capability**. Install the GHAChromeCastRoot driver under the Devices node. Once installed, the GHAChromeCast driver will search your network for Chrome Cast compatible devices and create a GHAChromeCaster device under the GHAChromeCastRoot node. You are free to rename the Caster nodes to whatever you like if the names are unique.



GHAChromeCastRoot
- Caster_6b672a83f486473a2a1ca54dc20bd8c2
- Caster_e5c4566bdef4de69e61b4ae445dbea79
- Caster_eb2ca8905aae87268296eadd3c7f3aa7

Universal Plug and Play

Universal Plug and Play (UPnP) defines a standard set of protocols and device specifications to allow compatible devices to interact. UPnP defines a specific set of devices and protocols designed to support Audio-Visual applications called UPnPAV. GHA implements the UPnPAV Control Point standard and supports control of UPnPAV Media Renderer devices. To install UPnPAV functionality, install the GHAUPnPDeviceRoot driver under the Devices node. Once installed, the driver will automatically begin searching for UPnPAV Media Renderer devices on your network. Discovered UPnPAV Media Renderer devices will appear as GHA Objects under the GHAUPnPDeviceRoot node. Unfortunately, many vendors do not fully implement the UPnPAV Media Renderer specification. The GHA UPnP driver will detect many incomplete implementations and ignore them. However, some truly incompatible devices will get created. To account for this, the GHA UPnP driver allows for user-defined exclusions of Models and Vendors. The fields are comma separated lists of Models/Vendors that should be excluded. Each item can leverage wildcards. Using a '?' character in the exclusion will look for matches containing all of the supplied characters and any single character in the position where the '?' is located. An '*' character will look for matches containing all of the supplied characters and any number of characters in the position where the '*' is located. Any discovered Media Renderer devices will implement the *IMediaPlayer* **Capability**.

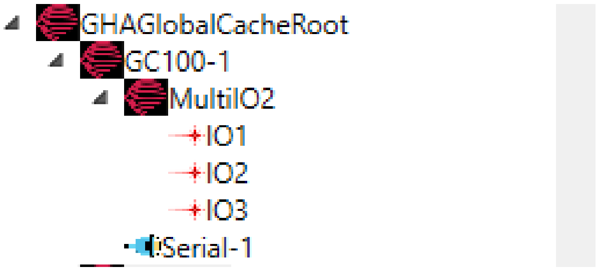


PropertyTable

GHAObject		
Class	◆	GHAUPnPDeviceRoot
Description	◆	
DisplayName	◆	
Icon	◆	Class.png
Name	◆	GHAUPnPDeviceRoot
Navigation	□	Visible
ObjectID	◆	9906b1b2-c195-4ea1-88cf-b7d2a1a0f7cf
Parent	◆	Devices
Path	◆	gha://gharoot/Devices/GHAUPnPDeviceRoot
GHAUPnPDriver.GHAUPnPDeviceRoot		
ExcludeModels	◆	XBR-65*
ExcludeVendors	◆	
IsRunning	◆	True
LoadRenderers	◆	☑
SuspendDiscovery	□	□
Version	◆	1.0

Global Cache GC-100

The Global Cache GC-100 is a network attached device that can present serial ports and Infrared transmitters on your home network.   While modern home automation devices are network-based, there are a number that still rely on RS-232 serial ports and infrared to control. The GC-100 allows GHA to communicate with these devices. The GC100 driver will automatically detect and provision newer GC-100's into the Devices>GHAGlobalCacheRoot portion of the hierarchy. Older devices can be added manually by right clicking on GHAGlobalCacheRoot and selecting New>GHAGC100. Once the device is added its IP address would need to be manually configured. That is not necessary for automatically detected by GC-100. Once the GC-100 is added to the hierarchy, the driver will attempt to determine the sub-devices contained within the discovered GC-100.   These sub-devices will appear under the GC-100 in the hierarchy. If no devices are shown, check the GetDevices property to initiate the device discovery manually. The GHA GC-100 driver currently supports serial ports and Infrared transmitters. An example of the GC-100 and how its sub-devices are show in Figure (xx).



```
◢ GHAGlobalCacheRoot
    ◢ GC100-1
        ◢ MultiIO2
              IO1
              IO2
              IO3
          Serial-1
```

CHIYU BF-430 Network Serial Port

USE THIS DEVICE AT YOUR OWN RISK. IT OFTEN GETS INTO A STATE WHERE IT REFUSES CONNECTIONS REQUIRING A POWER-CYCLE. WHEN IN THIS STATE, GHA MIGHT BECOME NON-RESPONSIVE

The CHIYU BF-430 is a low cost SMALL (2.6" x 3.6") industrial Single port RS232 or RS485 serial device server that can make your industrial serial devices to be IP / Ethernet network enabled. BF-430 can be added by adding the GHAGridConnect device under the Devices root, and manually adding the BF430SerialPort device under the GHAGridConnectRoot. Unfortunately, automatic detection of the BF-430 is not possible. You will note that the BF430 device properties do not include typical serial port properties (e.g. Baud rate). This is because the BF-430 does not support remote configuration of the serial port via API calls. Configuration of the BF-430 is done via browser.

Like other devices that implement the *ISerialPort* **Capability**, there are some properties that can be used either through the Configurator or through custom modules and scripting.   For example, the DataReceived property is set to true when the serial port has detected that the connected device has sent data to the serial port. BytesAvailable shows the number of bytes sent from the connected device to the serial port. Like all ISerialPort devices like the BF-430 can operate in a "stream" mode or in a "line-oriented" mode. The default mode is stream, where the DataReceived flag is set as soon as data is received on the serial port. In Line-oriented mode, the driver waits until the EndOfMessageMarker is presented to the serial port before setting the DataReceived flag.

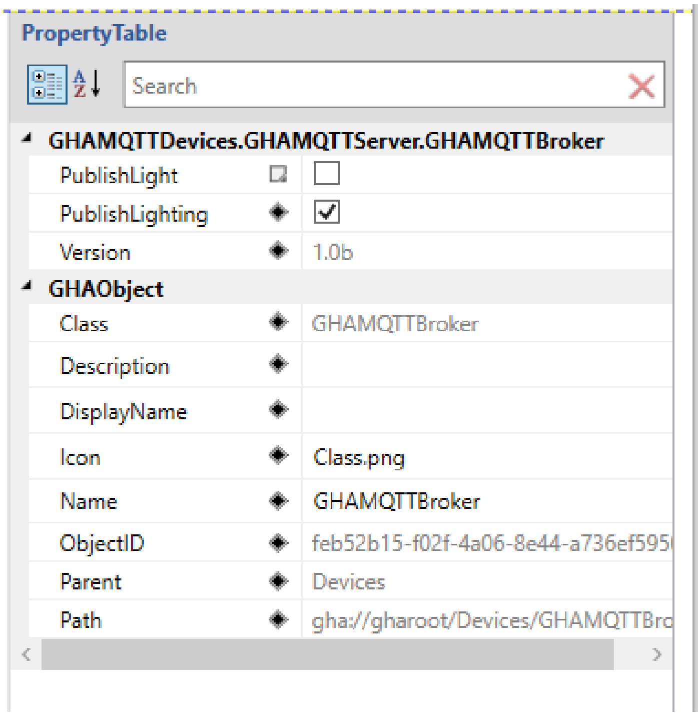
Legrand Nuvo Concerto Whole Home Audio System

The Concerto is an eight zone, six input matrix switcher with a built in amplifier. It is controlled via serial port commands. The GHA Grand Concerto driver models the device with two types of GHA Objects: Inputs and Zones. Once the GHAGrandConcerto device is created under the Devices root, the driver automatically creates the Input and Zone objects. The only configuration required by the driver is to assign the serial port connected to the Grand Concerto itself.

Once configured, the inputs can be bound to other audio devices defined in the GHA Devices hierarchy by using the standard GHA Bind functionality. GHA Media Zones located under the Home hierarchy can be mapped to Grand Concerto Zones. Once inputs and zones are properly bound, GHA can control how and where audio content is played. And here is where the magic of GHA's media handling can really be seen. Let's say your configuration has a Sonos device connected to one of the Concerto. When a user wants to play a Sonos favorite, they first navigate to the GHA Media Zone associated with the room where the content is to be played. They select the item and hit "Play". GHA takes over from there. First, the inputs connected to the Concerto are searched to determine if any are compatible with the required media. In this case, the Sonos device is found and allocated to play the Sonos favorite. GHA tells the Concerto to power-on the proper media zone and connect the selected input to that zone. Finally, GHA tells the Sonos device to select and play the Sonos favorite. That's it! There is no manual picking of inputs and outputs. No fumbling with your Sonos app to select and play the right media. Just simple, intelligent control of how audio is played.

MQTT

In addition to having a built-in MQTT broker, GHA can also support presenting of internal objects to an MQTT network or controlling external MQTT devices. To enable MQTT functionality, a GHQMQTTBroker device must be created. Do this by right-clicking on the Devices node and select New>IGHADevice>GHAMQTTBroker. The GHAMQTTBroker root node should appear under the Devices node in the hierarchy.



PropertyTable

Search

GHAMQTTDevices.GHAMQTTServer.GHAMQTTBroker	
PublishLight	□
PublishLighting	☑
Version	1.0b
GHAObject	
Class	GHAMQTTBroker
Description	
DisplayName	
Icon	Class.png
Name	GHAMQTTBroker
ObjectID	feb52b15-f02f-4a06-8e44-a736ef595
Parent	Devices
Path	gha://gharoot/Devices/GHAMQTTBro

Internal GHA devices are presented through a Client Proxy. For your convenience, the GHAMQTTBroker root can directly publish objects with the *Lighting* and *Light* **Capabilities** by simply selecting the PublishLight and PublishLighting properties as shown in Figure (XX). After PublishLight or PublishLighting is checked, a proxy object will be created under the GHAMQTTBroker. The presence of that object indicates that any object under the Home hierarchy that is of the Light or Lighting class and their properties will be automatically published to the MQTT network using the MQTT topic

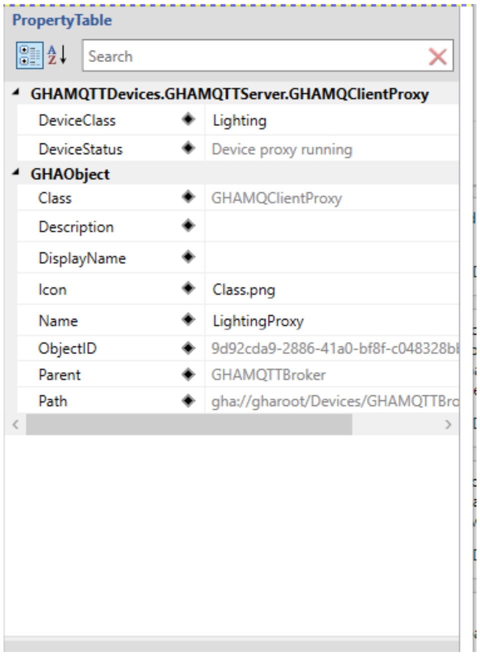
GHA/<ghaservername>/Objects/<ghaPathToObject>/<propertyname>

When a property for these proxied objects change, a MQTT message is published with a payload object containing the value of the property.

It is also possible to publish other object types by creating a GHAMQTTClientProxy and specifying the class of objects to be published. A Client Proxy is created by right-clicking on the GHAMQTTBroker node and select the GHAMQClientProxy. Once created, the Client Proxy requires that the *DeviceClass* property be specified. If the device class is valid, all objects under the Home hierarchy of that class will be published to the MQTT network. The *DeviceStatus* property will show if the device class is being successfully published.

In addition to publishing internal objects to an MQTT network, it can control external MQTT devices. Built-in support is provided for dimmable lighting devices through the GHAMQTTIDimmableDevice device class (right-click the GHAMQTTBroker node and select GHAMQTTIDimmableDevice to create one). New devices will be added in future releases as well as the ability to create custom devices.

Once the device is created, it needs to be configured to control a device on the MQTT network. This is done by creating a Mapping between GHA known properties and the topics controlling the remote device.

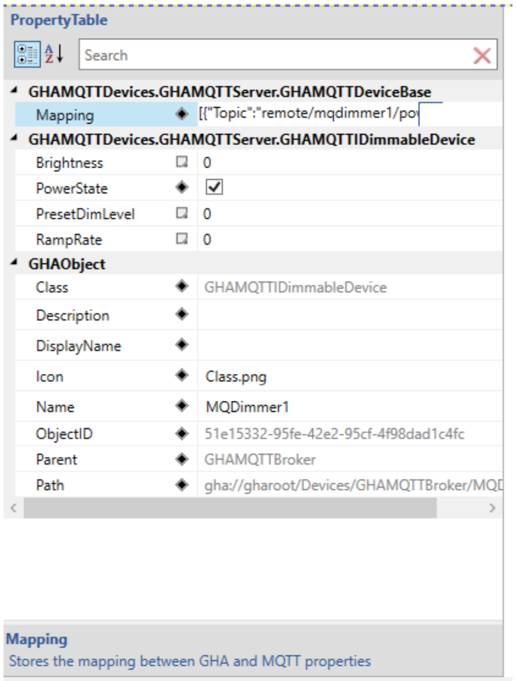


PropertyTable

Search

GHAMQTTDevices.GHAMQTTServer.GHAMQClientProxy	
DeviceClass	Lighting
DeviceStatus	Device proxy running
GHAObject	
Class	GHAMQClientProxy
Description	
DisplayName	
Icon	Class.png
Name	LightingProxy
ObjectID	9d92cda9-2886-41a0-bf8f-c048328bl
Parent	GHAMQTTBroker
Path	gha://gharoot/Devices/GHAMQTTBro

The GHAMQTTIDimmableDevice contains the properties associated *with the IDimmable* **Capability***.* Those properties must be mapped to the topics controlling the equivalent properties on the remote MQTT device. This is done by clicking on the browse button next to the Mappings property of the GHAMQTTIDimmableDevice. The MQTT GHA Map Builder window appears as shown in Figure (XX). The map builder displays the properties available with the GHA object. When one of these properties is selected, you will see the mapping between the internal GHA value and the MQTT value. For example, the *PowerState* property shows the internal values of *true* and *false* mapping to the MQTT values of *on* and *off*. Note that the MQTT topic for the *PowerState* property must be set to the topic used by the remote device. In Figure (xx) the topic is set to **remote/mqdimmer1/powerstate.** The value must be set to a topic the remote MQTT device will respond do. The MQTT payload will be set to the mappings specified for the property.



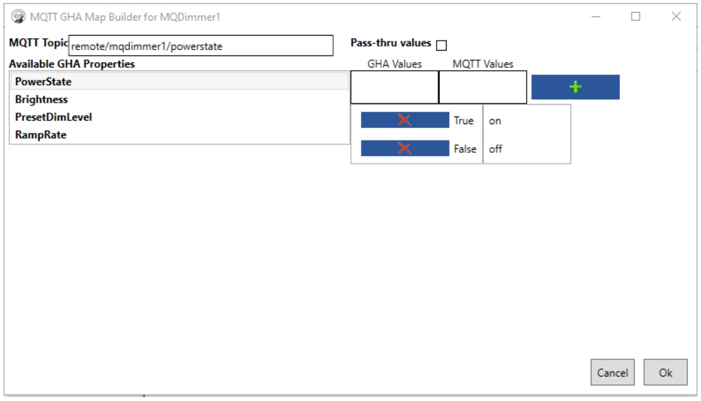
PropertyTable

Search

GHAMQTTDevices.GHAMQTTServer.GHAMQTTDeviceBase	
Mapping	[{"Topic":"remote/mqdimmer1/po
GHAMQTTDevices.GHAMQTTServer.GHAMQTTIDimmableDevice	
Brightness	0
PowerState	☑
PresetDimLevel	0
RampRate	0
GHAObject	
Class	GHAMQTTIDimmableDevice
Description	
DisplayName	
Icon	Class.png
Name	MQDimmer1
ObjectID	51e15332-95fe-42e2-95cf-4f98dad1c4fc
Parent	GHAMQTTBroker
Path	gha://gharoot/Devices/GHAMQTTBroker/MQD

Mapping

Stores the mapping between GHA and MQTT properties



MQTT GHA Map Builder for MQDimmer1

MQTT Topic: remote/mqdimmer1/powerstate

Pass-thru values □

Available GHA Properties

PowerState
Brightness
PresetDimLevel
RampRate

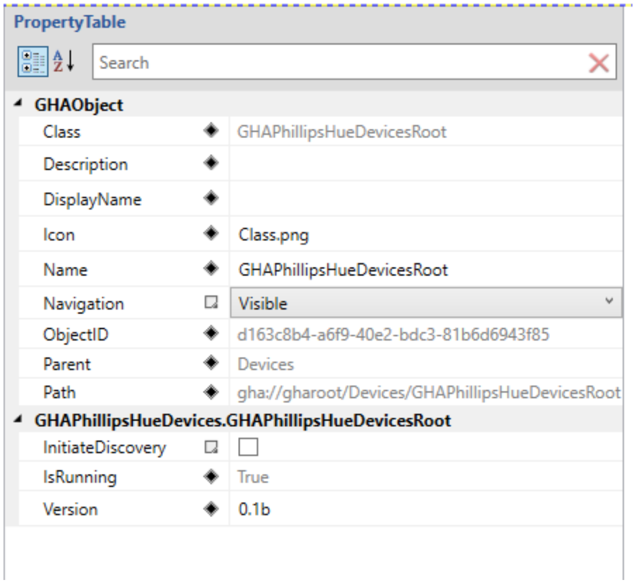
GHA Values	MQTT Values
True	on
False	off

Certain value types are automatically configured to be "pass-thru". For example, numeric values (integer and floating point) will be tagged as passthrough, and their values will be provided via MQTT Payload associated with the topic. In the case of the

GHAMQTTIDimmableDevice, Brightness, PresetDim Level and RampRate are floating point numbers that are passed via the MQTT topic without change.

Phillips Hue Lighting

GHA provides support for Phillips Hue Lighting by right-clicking the Devices node and selecting GHAPhillipsHueDevicesRoot. Once the driver is installed, a device discovery operation will be initiated that will populate the GHAPhillipsHueDevicesRoot with all Hue bridges discovered on your network. If necessary, it is possible to manually trigger a discovery operation by clicking on the InitiateDiscovery property of the GHAPhillipsHueDeviceRoot object.

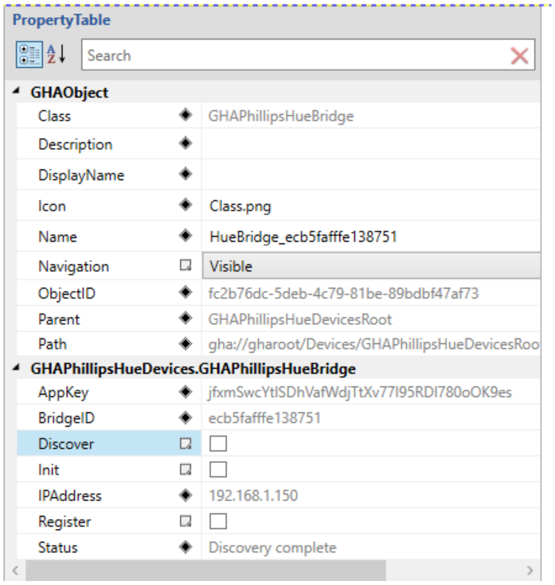


PropertyTable

Search

GHAObject		
Class	◆	GHAPhillipsHueDevicesRoot
Description	◆	
DisplayName	◆	
Icon	◆	Class.png
Name	◆	GHAPhillipsHueDevicesRoot
Navigation		Visible
ObjectID	◆	d163c8b4-a6f9-40e2-bdc3-81b6d6943f85
Parent	◆	Devices
Path	◆	gha://gharoot/Devices/GHAPhillipsHueDevicesRoot
GHAPhillipsHueDevices.GHAPhillipsHueDevicesRoot		
InitiateDiscovery		☐
IsRunning	◆	True
Version	◆	0.1b

Once a Hue bridge is discovered, it must be registered with GHA. This is done by selecting the discovered Hue Bridge under the GHAPhillipsHueDevicesRoot, PUSHING THE BUTTON ON THE HUE BRIDGE, and clicking on the Register property. It is incredibly important that the button on the Hue bridge is pushed prior to clicking on Register. Once registered, the Hue bridge object will acquire and store an AppKey that will subsequently be used to control devices associated with the Hue bridge. Once an AppKey is visible, click on the Init property to discover and create GHAObjects to control Hue Lighting devices. New devices may be added by clicking on the Discover property or by restarting GHA.



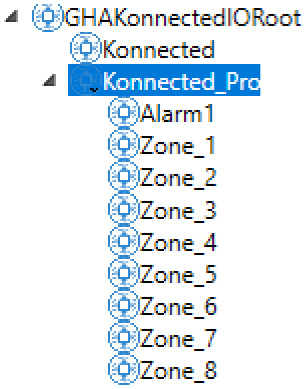
PropertyTable

Search

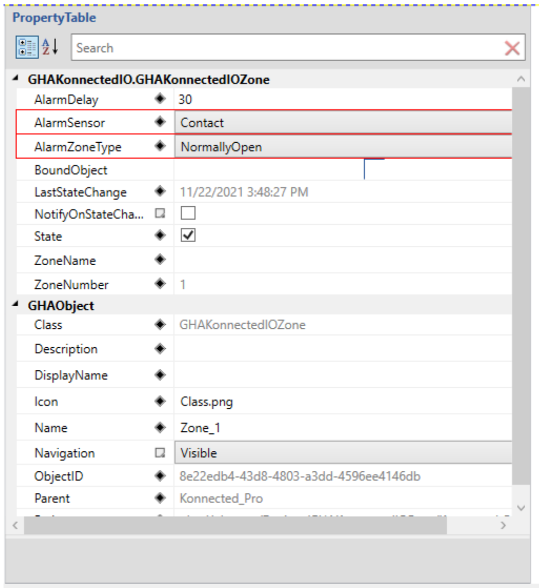
GHAObject		
Class	◆	GHAPhillipsHueBridge
Description	◆	
DisplayName	◆	
Icon	◆	Class.png
Name	◆	HueBridge_ecb5fafffe138751
Navigation		Visible
ObjectID	◆	fc2b76dc-5deb-4c79-81be-89bdbf47af73
Parent	◆	GHAPhillipsHueDevicesRoot
Path	◆	gha://gharoot/Devices/GHAPhillipsHueDevicesRoo
GHAPhillipsHueDevices.GHAPhillipsHueBridge		
AppKey	◆	jfxmSwcYtlSDhVafWdjTtXv77I95RDI780oOK9es
BridgeID	◆	ecb5fafffe138751
Discover		☐
Init		☐
IPAddress	◆	192.168.1.150
Register		☐
Status	◆	Discovery complete

Konnected Alarm Panel

GHA has built-in support for the Konnected Alarm Panel Pro. The supplied driver implements the *IAlarmControl* **Capability**. Once the driver has been enabled, it can be installed by right-clicking on the Devices node and selecting the GHAKonnectedIORoot device. The driver will then detect any Konnected Alarm Panel Pro hardware installed on your network. The driver will then attempt to provision the device and configure it to report alarm events to your GHA server. If you have pre-configured the Konnected hardware with zones and alarms, the driver will import that configuration into GHA. The figure shows a portion of the GHA hierarchy containing Konnected Pro device with zones and an alarm configured. In addition to importing an existing configuration, it is possible to add zones and alarms within GHA by right clicking on the Konnected Pro device node and selecting the desired sensor or alarm to add.



```
GHAKonnectedIORoot
    Konnected
    Konnected_Pro
        Alarm1
        Zone_1
        Zone_2
        Zone_3
        Zone_4
        Zone_5
        Zone_6
        Zone_7
        Zone_8
```



PropertyTable

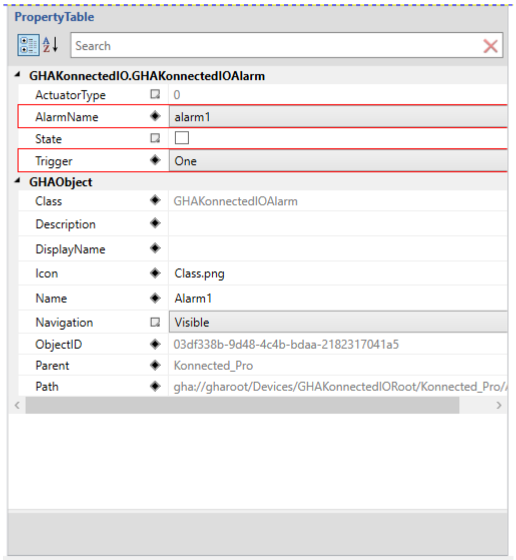
Search

GHAKonnectedIO.GHAKonnectedIOZone		
AlarmDelay	◆	30
AlarmSensor	◆	Contact
AlarmZoneType	◆	NormallyOpen
BoundObject		
LastStateChange	◆	11/22/2021 3:48:27 PM
NotifyOnStateCha...	□	□
State	◆	☑
ZoneName	◆	
ZoneNumber	◆	1
GHAObject		
Class	◆	GHAKonnectedIOZone
Description	◆	
DisplayName	◆	
Icon	◆	Class.png
Name	◆	Zone_1
Navigation	□	Visible
ObjectID	◆	8e22edb4-43d8-4803-a3dd-4596ee4146db
Parent	◆	Konnected_Pro

The figure shows the properties associated with a Konnected Zone. The AlarmDelay property defines the number of seconds to wait before triggering an alarm. This configuration is useful for entryways into a home where you wish to allow some time to disarm an alarm after home entry.

The AlarmSensor property allows you to select the type of sensor being monitored by this zone. These include Contact, Smoke, Fire, Carbon Monoxide (CO), and Motion. It is imporant to correctly configure this property in order to inform the Noonlight alarm monitoring service (if used) of the emergency service that is required.

The AlarmZoneType property allows you to configure if the sensor is Normally Open or Normally Closed. For example, most door sensors are normally closed until the door is opened. This will allow the GHA Alarm Controller to correctly interpret the alarm status associated with the zone.



PropertyTable

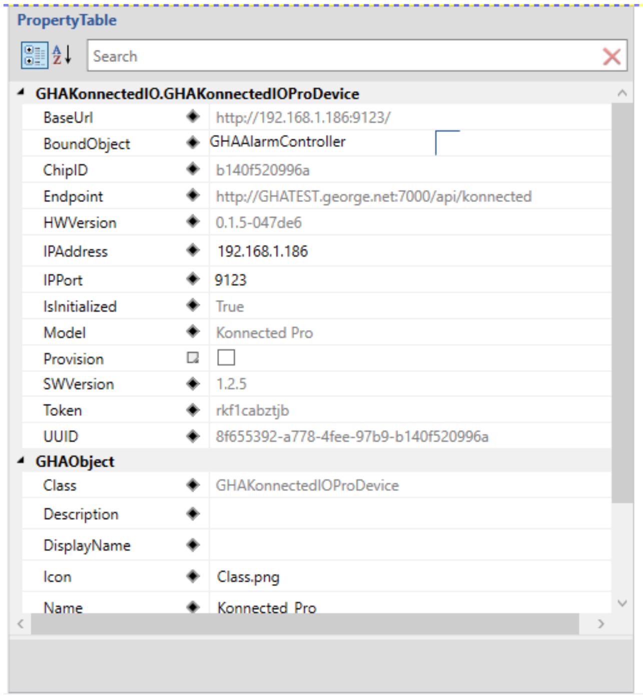
Search

GHAKonnectedIO.GHAKonnectedIOAlarm		
ActuatorType	□	0
AlarmName	◆	alarm1
State	□	□
Trigger	◆	One
GHAObject		
Class	◆	GHAKonnectedIOAlarm
Description	◆	
DisplayName	◆	
Icon	◆	Class.png
Name	◆	Alarm1
Navigation	□	Visible
ObjectID	◆	03df338b-9d48-4c4b-bdaa-2182317041a5
Parent	◆	Konnected_Pro
Path	◆	gha://gharoot/Devices/GHAKonnectedIORoot/Konnected_Pro/

The figure shows the properties associated with the Konnected Alarm. The AlarmName property is the internal name used by the Konnected Pro device for the alarm. Please refer to the Konnected Pro documentation for more information. The driver default is “alarm1”. The Trigger property defines how the Konnected Pro will trigger the alarm. Please refer to the Konnected Pro documentation for more information. The driver default is “One”.

The figure shows the Konnected Pro device and it's associated properties. Note the "Provision" property. When clicked, the Konnected driver will provision the Konnected Pro hardware with the zones and alarms currently defined in GHA. It will also configure the Konnected Pro hardware to report alarm status to your GHA Server. Note that after a provisioning, the Konnected device will reboot.

Noonlight Monitoring

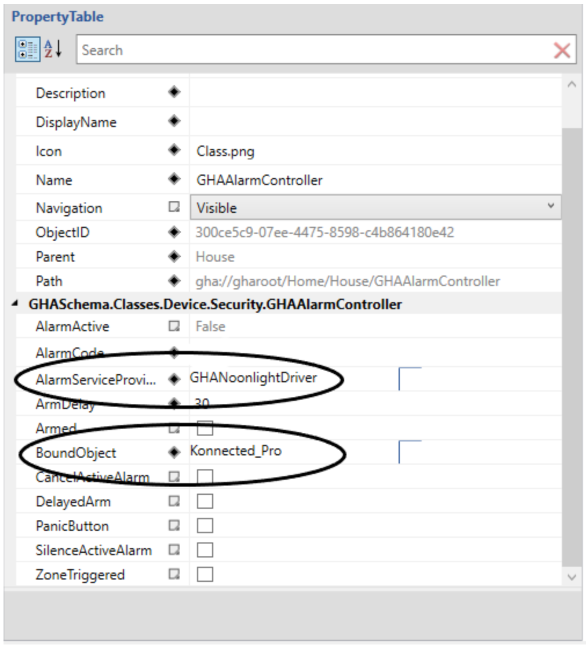


PropertyTable

GHAKonnectedIO.GHAKonnectedIOProDevice		
BaseUrl	◆	http://192.168.1.186:9123/
BoundObject	◆	GHAAlarmController
ChipID	◆	b140f520996a
Endpoint	◆	http://GHATEST.george.net:7000/api/konnected
HWVersion	◆	0.1.5-047de6
IPAddress	◆	192.168.1.186
IPPort	◆	9123
IsInitialized	◆	True
Model	◆	Konnected Pro
Provision	□	□
SWVersion	◆	1.2.5
Token	◆	rkf1cabztjb
UUID	◆	8f655392-a778-4fee-97b9-b140f520996a
GHAObject		
Class	◆	GHAKonnectedIOProDevice
Description	◆	
DisplayName	◆	
Icon	◆	Class.png
Name	◆	Konnected_Pro

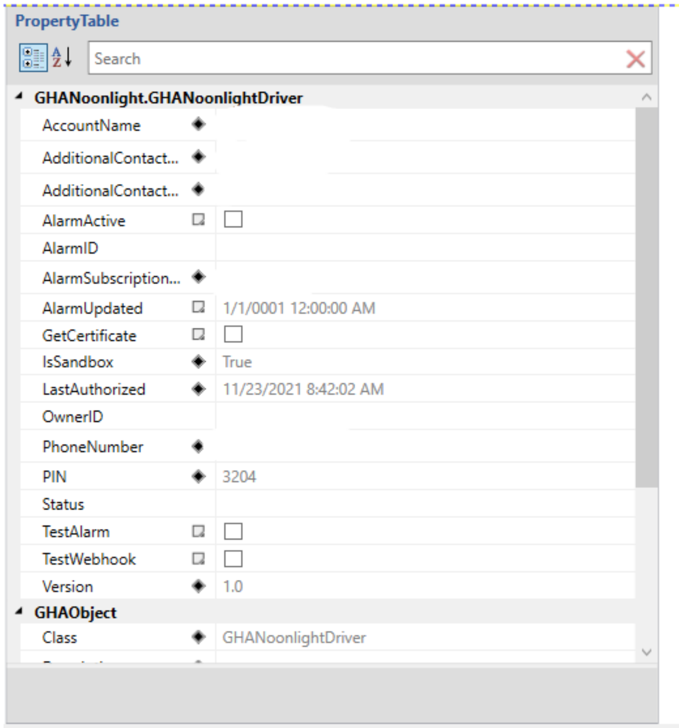
GHA has partnered with Noonlight to provide a professional 24x7 monitoring for US customers. The service itself is a separately purchased item. Please see the George HA website (https://www.george-home.com) for additional details. The Noonlight monitoring interface is implemented as a standard GHA Driver. Once enabled, it is installed by right-clicking on the Devices node in the GHA Hierarchy and selecting the GHANoonlightDriver.

The Noonlight driver needs to be associated with Alarm Controller configured under the Home Node. The figure shows the properties of a GHA Alarm Controller. Note the AlarmServiceProvider property is set to the GHANoonlightDriver. Also, note that we've bound the Alarm Controller to a Konnected Pro device. It is important to understand that the GHA Alarm Controller MUST be bound with a compatible alarm panel device such as the Konnected Pro. Further, the GHA Alarm Controller MUST be associated with an instance of the GHANoonlightDriver for the alarm monitoring to be enabled.



PropertyTable

Description	◆	
DisplayName	◆	
Icon	◆	Class.png
Name	◆	GHAAlarmController
Navigation	□	Visible
ObjectID	◆	300ce5c9-07ee-4475-8598-c4b864180e42
Parent	◆	House
Path	◆	gha://gharoot/Home/House/GHAAlarmController
GHASchema.Classes.Device.Security.GHAAlarmController		
AlarmActive	□	False
AlarmCode	◆	
AlarmServiceProvi...	◆	GHANoonlightDriver
ArmDelay	◆	30
Armed	□	□
BoundObject	◆	Konnected_Pro
CancelActiveAlarm	□	□
DelayedArm	□	□
PanicButton	□	□
SilenceActiveAlarm	□	□
ZoneTriggered	□	□



PropertyTable

Search

GHANoonlight.GHANoonlightDriver

AccountName	
AdditionalContact...	
AdditionalContact...	
AlarmActive	□
AlarmID	
AlarmSubscription...	
AlarmUpdated	1/1/0001 12:00:00 AM
GetCertificate	□
IsSandbox	True
LastAuthorized	11/23/2021 8:42:02 AM
OwnerID	
PhoneNumber	
PIN	3204
Status	
TestAlarm	□
TestWebhook	□
Version	1.0

GHAObject

Class	GHANoonlightDriver

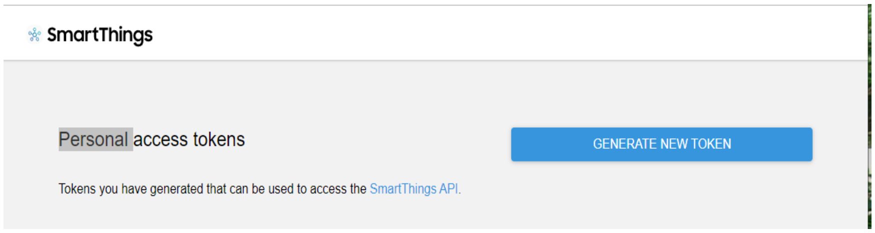
The figure shows the properties associated with the GHA Noonlight driver. Several settings must be provided to enable the Noonlight service. First, purchase a product key from the GHA website. The product key must be typed into the AlarmSubscriptionKey field and hit return. The key will be validated. You'll be notified if the key is valid or invalid. Next, the AccountName and PhoneNumber properties must be populated with accurate information. The phone number MUST be provided in the 1XXXYYYZZZZ format. No spaces or special characters are allowed.

In addition to the Noonlight driver configuration, the Schedule node in the GHA Hierarchy MUST have address information defined. This address is provided to Noonlight during an alarm event which will in-turn be provided to first responders. It is critically important that this information is provided and accurate. IF THE REQUIRED INFORMATION OR A VALID PRODUCT KEY IS NOT PROVIDED, THE NOONLIGHT MONITORING SERVICE WILL NOT RESPOND TO ALARM EVENTS.

In addition to the primary contact information, the GHA Noonlight driver allows for an additional contact name and phone number. If the primary contact does not respond to the Noonlight monitoring center, the additional contact will be notified. If nobody responds to the Noonlight monitoring center, Noonlight will dispatch the appropriate first responder for the address provided in the Schedule node.

SmartThings

GHA Support for SmartThings is via the GHA SmartThings driver controlling a SmartThings Hub. You must first configure the SmartThings Hub and associate devices before enabling the GHA SmartThings driver. Once your hub is configured, install the GHA SmartThings driver like you would any other GHA driver. However, to make the driver work, you will need a Personal Access Token (PAT) from the SmartThings website. First, ensure you have a SmartThings account. You should have created one when you configured your SmartThings Hub. Navigate to the SmartThings PAT web page and sign-in with your SmartThings account. Click on the GENERATE NEW TOKEN button. You will be presented with a page that allows you to name your new access token and grant it privileges. Give your token a name you will recognize (like GHASmartThingsToken) Provide your token with ALL PRIVILEGES by checking the major categories (i.e. Devices, Installed Applications, Applications, etc.).



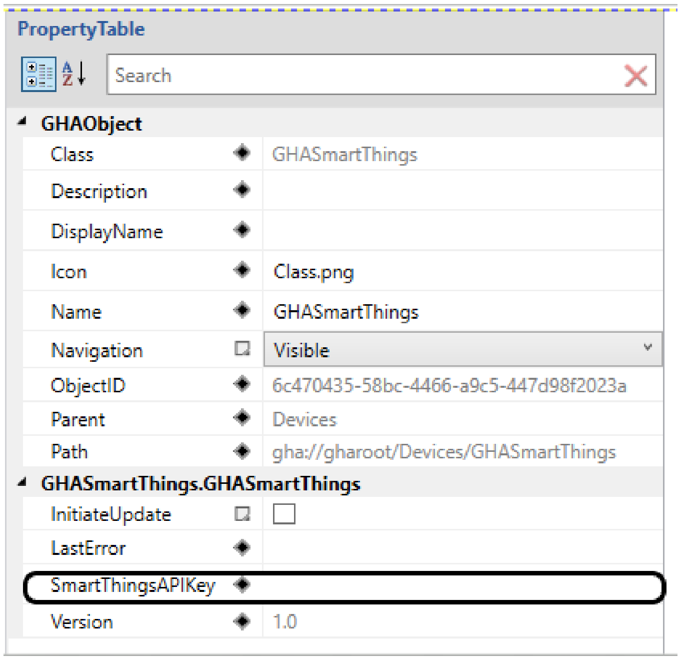
SmartThings

Personal access tokens

GENERATE NEW TOKEN

Tokens you have generated that can be used to access the SmartThings API.

Then click the GENERATE TOKEN button at the button of the page. You will then be shown your token. COPY IT IMMEDIATELY AND KEEP IT IN A SAFE PLACE. YOU WILL NOT BE ABLE TO SEE IT AGAIN. Paste or type your newly acquired API key into the SmartThingsAPIKey property as shown. Click InitiateUpdate and your SmartThings devices should start appearing under the GHASmartThings node.



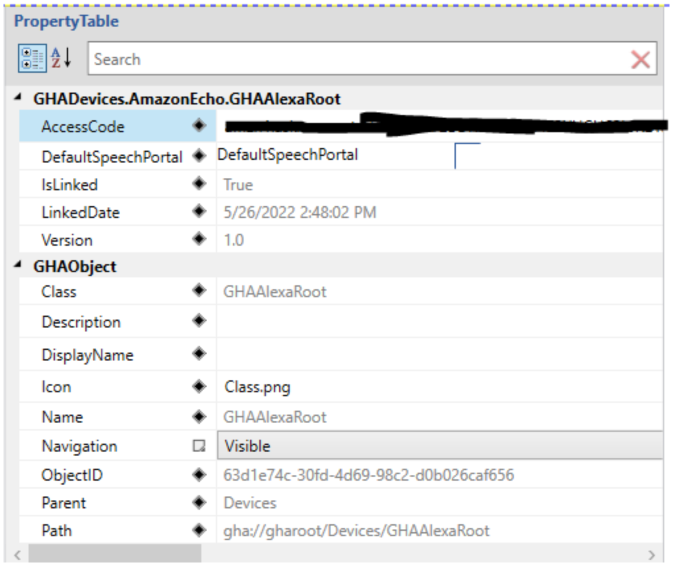
GHAObject		
Class		GHASmartThings
Description		
DisplayName		
Icon		Class.png
Name		GHASmartThings
Navigation		Visible
ObjectID		6c470435-58bc-4466-a9c5-447d98f2023a
Parent		Devices
Path		gha://gharoot/Devices/GHASmartThings
GHASmartThings.GHASmartThings		
InitiateUpdate		□
LastError		
SmartThingsAPIKey		
Version		1.0

Amazon Echo and Alexa speech recognition

GHA supports the Amazon Echo in two ways. First, as a method to provide notifications. Second, as one of GHA's supported speech recognition technologies.

Amazon Echo notifications

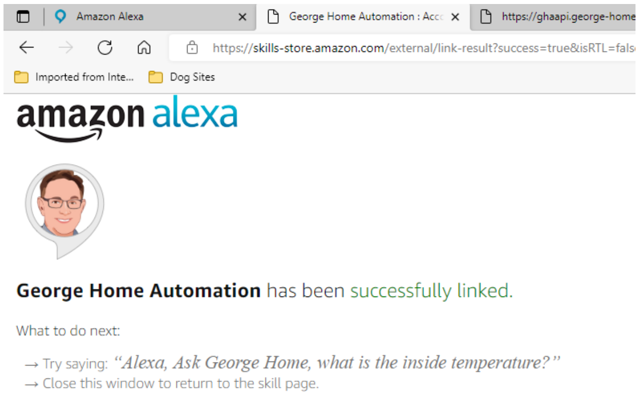
Notifications rely on a skill called Notify Me by Thomptronics. To enable notifications through your Amazon Echo devices, you must enable the skill. Follow the instructions to link the skill and obtain the AccessCode associated with your Echo (Notify Me will send you an email with the key). Cut and paste that key into the AccessCode property under the GHAAlexaRoot node. Your Amazon Echo devices will be enabled GHA text to speech devices implementing the *ITTSText* **Capability**. Your custom drivers, modules, and scripts can now send notifications through your Amazon Echo devices. Please see the example in the GHA Developer Guide, **The IGHAObject (a deeper dive)** section.



GHADevices.AmazonEcho.GHAAlexaRoot		
AccessCode		[redacted]
DefaultSpeechPortal		DefaultSpeechPortal
IsLinked		True
LinkedDate		5/26/2022 2:48:02 PM
Version		1.0
GHAObject		
Class		GHAAlexaRoot
Description		
DisplayName		
Icon		Class.png
Name		GHAAlexaRoot
Navigation		Visible
ObjectID		63d1e74c-30fd-4d69-98c2-d0b026caf656
Parent		Devices
Path		gha://gharoot/Devices/GHAAlexaRoot

Amazon Alexa Speech Recognition

The GHA Alexa driver works with the GHA Speech Recognition Framework. To utilize your Amazon Alexa service, you must link your GHA Server to your Alexa account. This is done either in the Alexa app or the Alexa website. While linking, you will be asked for your Machine ID and your System ID. Both are found as properties under the Server node in the Configurator. For your convenience, you can right click on the MachineID and SystemID nodes to copy the values into your clipboard which can be subsequently used in the linking process. Once successfully linked, you will see a page like the figure. Once linked, you can start using your Amazon Echo devices for speech recognition.


Amazon Alexa
George Home Automation : Acc
https://ghaapi.george-home
https://skills-store.amazon.com/external/link-result?success=true&isRTL=fals
Imported from Inte...
Dog Sites
amazon alexa

George Home Automation has been successfully linked.

What to do next:

→ Try saying: *"Alexa, Ask George Home, what is the inside temperature?"*
→ Close this window to return to the skill page.

GHAEcho devices will be created under the Devices>GHAAlexaRoot node when a command from the device is connected. It is recommended that you invoke a simple voice command on each Echo device one at a time, and rename the device to something meaningful for you (e.g. KitchenEcho). While not strictly required, it is recommended that you create a GHASpeechPortal under the Home hierarchy where your physical Echo devices are located. This will allow the GHA Speech Framework to search for objects to control/query closest to the Echo device. Alternatively, you could simply let the DefaultSpeechPortal handle speech requests. It is important that you do not bind the DefaultSpeechPortal to an Echo device.

All GHA Alexa commands must start with “Open George Home”, and then follow with a supported command. GHA Alexa speech recognition supports the following commands:

HVAC

“What is the inside temperature”

“What is the inside temperature in the [room name]

Lighting

“Turn [on/off] lights”

“Turn [on/off] [room name] lights”

Garage

“What is the status of the garage?”

Scenes

“stop scene [scene name]”

“deactivate scene [scene name]”

“play scene [scene name]”

“activate scene [scene name]”

Media

“play playlist [playlist name]”

“play playlist on [media zone name]”

“play album [album name]”

“play album [album name] in “[media zone name]”

“play anything by [artist name]”

“play something by [artist name]”

“play something by [artist name]” in the [media zone]

“play anything by {artist name]” in the {media zone]”

“stop playing”

“stop music”

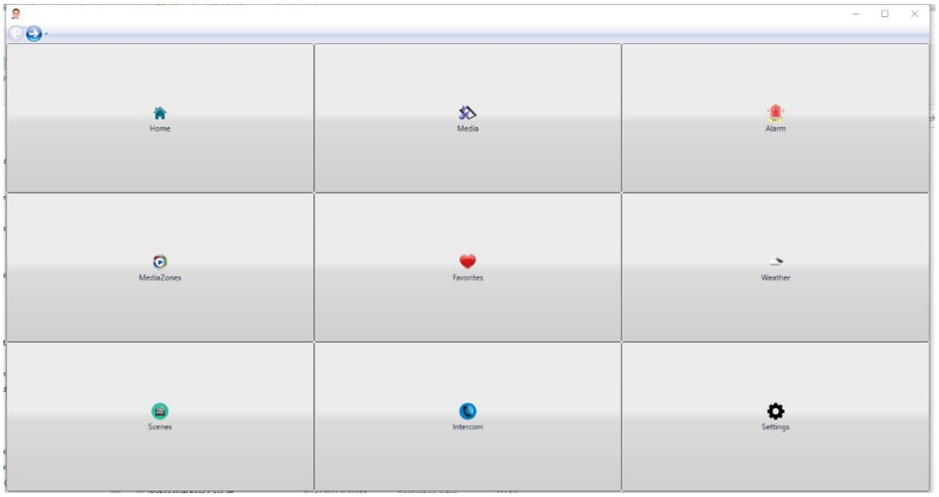
“stop [media zone]”

“stop the [media zone]”

“stop what is playing on [media zone]”

User Interfaces

GHA ships with user interfaces (UI) for Windows (GHAWinUI) and Android devices (iOS version is in development). The Windows UI is available from the George Home Automation website, Downloads section. Once installed, the GHAWinUI will prompt you for your GHA Server name and GHA Server credentials. Once authenticated, you will see a window that looks similar to the figure. Pressing the "Home" button will render your GHA Objects specified in your GHA Hierarchy (what you defined in Configurator). For your quick access, hierarchy items such as Media, Media Zones, Scenes, and Intercom are available on this first page. You also have the ability to define Favorites by right-clicking an object and selecting the "Add to Favorites" menu item.



The Android version of the user interface is available on the Google Play Store. The Android version is designed for both mobile phones and tablets. The tablet version is targeted for wall-mounted controllers and optimized for devices in landscape mode. Like the Windows version, you must provide credentials and your GHA Server name. This is by tapping LOGIN SETTINGS under the Settings window.

GHA Built-in Classes

GHA defines a library of built-in classes that can be used to model virtually any home environment. This section describes the various classes available, their inheritance tree, and the properties that can be used to control the behavior of objects created from the classes.

Base class

The GHAObject is the base class for all objects in the GHA System. The class defines the following properties.

Name	Description
Children	Collection containing the children of the object
Class	Type from which the object is derived
Description	Free-form description of the object
DisplayName	Name to be used by end-user displays to refer to the object
Icon	Icon used to represent the object
Name	Name of the object
ObjectID	Unique ID of the object expressed as a .NET GUID
Parent	The object owning the current object in the hierarchy
Path	URI pointing to the location of the object in the hierarchy

Root classes

Root classes appear at the top of the hierarchy and are designed to contain related classes.

Home

The Home hierarchy contains objects that model the physical home environment. For example, locations and device abstractions.

Devices

The Devices hierarchy contains all physical device objects defined for the GHA installation.

Information Sources

The Information sources hierarchy contains objects that can provide external information to the rest of the GHA system. For example, weather information is provided as an object under Information Sources.

Media

The Media root contains media objects managed by GHA

Modules

The Modules root contains the objects that can be used to create custom classes for GHA.

Monitors

The Monitors root contains objects that can provide status of various GHA components. For example, scheduled tasks.

Scenes

The Scenes root contains objects that can control multiple objects collectively.

Server

The Server root contains objects that can define users and presence information for this installation of GHA.

Home root classes

There are several classes that can be used to create objects under the home hierarchy. These typically include models for locations (i.e. rooms, hallways, etc.) and device abstractions (i.e. lighting, thermostats, media zones, etc.).

Location

Classes that inherit from the Location class implement the *Occupancy* **Capability**. They include the following properties.

Name	**Description**
LastOccupied	The last time the location was occupied
OccupancyCount	The number of times the location has been occupied
OccupancyDuration	Occupancy timer duration. When the area is occupied, an occupancy timer will start and run for the specified duration before declaring the area unoccupied. If an occupancy trigger occurs while the timer is running, the timer will reset to the value specified by this property.
Occupied	Determines if location is occupied (true) or not (false)

The classes below inherit from the Location class.

Building

Models structures that may be present on the property (e.g. external garage)

Floor

Models levels within a structure (e.g. First floor)

House

Models the enclosing living space (i.e. apartment, condo, house). Should be the highest level node under Home.

Property

Models the grounds on which structures are built.

Room

Base class to model locations within a structure.

CommonArea

Inherits from Room and models locations within a structure that can be used by multiple people

Dining

Models a dining room

Family

Models a family room

Foyer

Models a foyer

Hallway

Models a hallway

Kitchen

Models a kitchen

Living

Models a living room

Staircase

Models a stairway

OutsideArea

Inherits from Location and models areas outside of a structure

Patio

Models an exterior patio.

Yard

Models the various portions of the land on which structures reside (i.e. Front yard, back yard, etc.)

PersonalArea

Inherits from Room and models locations within a structure that are normally single use

Bedroom

Models a bedroom

Master

Models a master suite

UtilityArea

Inherits from Room and models locations within a structure that don’t cleanly fit in other categories

Garage

Models an enclosed garage

Theatre

Models a home theatre

Device classes

Device classes under the home root are designed to be abstractions of actual devices. They model device controls based on the **Capabilities** assigned to the device class. Until they are bound to a device under the Devices root, they cannot control anything in the home environment.

Appliance

Models appliances

GHAGarageDoorOpener

Models garage door opener and defines the following properties.

Name	Description
BoundObject	Object bound to this object
State	Sets or gets the state of the output

AudioVideo

Models various audio/video devices

GHAMediaZone

The Media zone is an object the abstracts media devices. The concept is that devices are bound to a media zone and the underlying drivers implement control of those bound devices. The MediaZone defines the following properties.

Name	Description
Balance	Left-right levels between 0.0(full-left) and 1.0(full-right). Centered at 0.5
Bass	Low-end Bass level between 0.0 and 1.0
BoundObject	Object bound to this object
CanQueue	Bound object supports built-in queuing mechanism
Content	Content currently selected for playing by the media zone
CurrentTrack	Currently playing track
Duration	Duration of the media currently loaded on the transport
Loudness	Sets Loudness response between 0.0 and 1.0

MediaFinishedPlaying	Momentary. Raised when currently selected media is finished playing and the play state of the device is true
Mute	Immediately set volume to zero when set to true and return to previous level when set to false
Pause	Temporarily suspends playback when set to true, resumes playback from current media position when set to false
Play	Initiates playback when set to true, stops transport when set to false
Progress	Percentage of current media processed by the transport
SupportedMedia	List of supported media types separated by commas. MIME types
TransportCommand	Sets desired state for the underlying transport
Treble	High-end Treble level between 0.0 and 1.0
Volume	Sound pressure level between 0.0 and 1.0

GHAMatrixSwitcher

Models a matrix switcher device. A matrix switcher establishes connections between inputs and outputs. The GHA Matrix switcher model implements a unique capability that can automatically select an input based on the type of media being played. The model implements the following properties.

Name	**Description**
NumberOfInputs	Number of matrix switcher inputs
NumberOfOutputs	Number of matrix switcher outputs

GHAMatrixSwitcherInput

Models the input of the matrix switcher.

GHAMatrixSwitcherZone

Models the output of a matrix switcher and defines the following properties.

Name	**Description**
Balance	Left-right levels between 0.0(full-left) and 1.0(full-right). Centered at 0.5
Bass	Low-end Bass level between 0.0 and 1.0
BoundObject	Object bound to this object
CurrentInput	Input currently associated with this output
CurrentTrack	Media currently loaded into the transport
Duration	Duration of the media currently loaded on the transport
IsQueueEmpty	Determines if que is empty (true) or populated (false)
Loudness	Sets Loudness response between 0.0 and 1.0

MediaFinishedPlaying	Momentary. Raised when currently selected media is finished playing and the play state of the device is true
Mute	Immediately set volume to zero when set to true and return to previous level when set to false
Pause	Temporarily suspends playback when set to true, resumes playback from current media position when set to false
Play	Initiates playback when set to true, stops transport when set to false
PowerState	Power state, On if true, Off if false
Progress	Percentage of current media processed by the transport
SupportedMedia	List of supported media types separated by commas. MIME types
TransportCommand	Sets desired state for the underlying transport
Treble	High-end Treble level between 0.0 and 1.0
Volume	Sound pressure level between 0.0 and 1.0
ZoneNumber	Number identifying the zone

HVAC

Models various environmental control capabilities

GHAThermostat

Models a thermostat device with the following properties.

Name	**Description**
BoundObject	Object bound to this object
CoolingSetPoint	Gets or sets the requested temperature for the cooling subsystem
CurrentSetPoint	Gets or sets the currently reqeusted temperature of the HVAC system
FanControl	Gets or sets the fan state (true=on, false=off)
FanStatus	Gets the current fan running status (true=on, false = off)
HeatingSetPoint	Gets or sets the requested temperature for the heating subsystem
HeatingStatus	Gets the status of the heating component
Mode	Get schedule status for the HVAC
OutdoorTemperature	Gets the outdoor temperature
SetbackMode	Gets or sets the setback mode
SetbackOffset	Gets or sets the offset for the setback
State	Gets or sets the system state
Temperature	Gets the measured temperature
TemperatureMode	Gets or sets the temperature mode
UserHold	Gets or sets the command to freeze temperature at the current set point

InputOutput

Models various input/output devices

GHASerialPortBase

For more information on this class, please see the GHA Developers guide

Lighting

This is the base class for all GHA Light device models and implements the *PowerState* **Capability**.

Name	**Description**
PowerState	Power state, On if true, Off if false

Light

Inherits from lighting and implements the *Dimmable* **Capability**.

Name	**Description**
BoundObject	Object bound to this object
Brightness	Level of lighting between 0.0 and 1.0
PowerState	Power state, On if true, Off if false (Inherited from Lighting.)
PresetDimLevel	Level to set lighting when the PowerState is true between 0.0 and 1.0
RampRate	Rate in seconds at which to get desired lighting level

MultiColorLight

Inherits from Lighting and implements the ColoredLighting **Capability**.

Name	**Description**
BlueLevel	Blue component of the desired color
BoundObject	Object bound to this object (Inherited from Light.)
Brightness	Level of lighting between 0.0 and 1.0 (Inherited from Light.)
GreenLevel	Green component of the desired color
PowerState	Power state, On if true, Off if false (Inherited from Lighting.)
PresetDimLevel	Level to set lighting when the PowerState is true between 0.0 and 1.0 (Inherited from Light.)
RampRate	Rate in seconds at which to get desired lighting level (Inherited from Light.)
RedLevel	Red component of the desired color

OnOffLight

Inherits from Lighting and implements a basic on/off light.

Name	Description
BoundObject	Object bound to this object
PowerState	Power state, On if true, Off if false (Inherited from Lighting.)

Safety

Models safety-related devices

GHACODetector

Implements the carbon dioxide detector device model.

Name	Description
BoundObject	Object bound to this object
CODetected	Determines if Carbon monoxide is present (true) or not (false)

GHASmokeDetector

Implements the smoke detector device model.

Name	Description
BoundObject	Object bound to this object
SmokeDetected	Determines if smoke is present (true) or not (false)

GHAWaterSensor

Implements the water sensor device model.

Name	Description
BoundObject	Object bound to this object
WaterDetected	Determines if water is present (true) or not (false)

Security

Implements various security-related device models

GHAGarageDoorSensor

Implements the garage door sensor device model.

Name	Description
BoundObject	Object bound to this object
GarageDoorOpened	Opened if true, closed if false

GHAMotionDetector

Implements the motion detector device model

Name	Description
BoundObject	Object bound to this object
LastTimeTriggered	The date/time the last time motion was detected
MotionDetected	Determines if motion has been detected (true) within the sampling period or not (false)

GHAPresenceZone

Implements the Presence zone device model

Name	Description
BoundObject	Object bound to this object
LastPresenceChanged	The date/time the last presence changed occurred
PresenceChanged	Set when the detected entities has changed
UsersPresent	Users detected in this presence Zone

GHASecurityCamera

Implements the security camera device model

Name	Description
BoundObject	Object bound to this object
Uri	Uri to access the camera video H.264 stream
ThumbnailUri	Optional Uri to access the thumbnail picture of the camera video
FPS	Stream frame rate in frames per second
IsAudioCapable	True if the camera stream includes an audio sub-stream
Width	Width of the camera image in pixels
Height	Height of the camera image in pixels
NewAlerts	Number of new alerts detected by the camera. "New" is defined by the underlying driver
LastAlert	Date/Time of the last alert
IsOnline	State of the camera as reported by the underlying driver
Error	Last error reported by the camera
MpegUri	Uri to access the camera MPEG 2 stream
Preferred Stream	Selects the stream preferred by UI devices. GHA Mobile apps for Android and iOS require the MPEG stream

GHA Capability Definitions

IBindable

The **IBindable** type exposes the following members.

Properties

	Name	Description
	BoundObject	Object bound to this object

ILocationInfo

Defines the location info capability

The **ILocationInfo** type exposes the following members.

Properties

	Name	Description
	City	City for the location
	Latitude	Latitude of the location
	Longitude	Longitude of the location
	State	State of the location
	ZipCode	Zip code for the location

IObjectReference

Defines the object reference capability

The **IObjectReference** type exposes the following members.

Properties

	Name	Description
	ObjectReference	Object reference

IQuartzScheduler

Defines the Quartz Scheduler capability

The **IQuartzScheduler** type exposes the following members.

Properties

	Name	Description
	Error	Last error encountered by the scheduler
	Running	Determines running state of the schedule (true if running)
	RunningSince	Date time the scheduler was started

	Version	Version of the Quartz scheduler

Methods

	Name	Description
	GetJobs	Returns jobs currently managed by the scheduler

IShutdown Interface

Defines the Shutdown capability

The **IShutdown** type exposes the following members.

Methods

	Name	Description
	Shutdown	Implementation specificc shutdown method. Implementors should deal with any class specific shutdown processing here

IClimateControl

Defines the Climate Control capability

The **IClimateControl** type exposes the following members.

Properties

	Name	Description
	FanControl	Gets or sets the fan state (true=on, false=off)
	FanStatus	Gets the current fan running status (true=on, false = off)
	HeatingStatus	Gets the status of the heating component
	Mode	Get schedule status for the HVAC
	State	Gets or sets the system state
	TemperatureMode	Gets or sets the temperature mode

ITemperatureController

Defines the Temperature Controller capability

The **ITemperatureController** type exposes the following members.

Properties

	Name	Description
	CoolingSetPoint	Gets or sets the requested temperature for the cooling subsystem
	CurrentSetPoint	Gets or sets the currently requested temperature of the HVAC system
	HeatingSetPoint	Gets or sets the requested temperature for the heating subsystem
	OutdoorTemperature	Gets the outdoor temperature

	UserHold	Gets or sets the command to freeze temperature at the current set point

ITemperatureSensor

Defines the Temperature sensor capability

The **ITemperatureSensor** type exposes the following members.

Properties

	Name	Description
	Temperature	Gets the measured temperature

IThermostat

Defines the Thermostat capability

The **IThermostat** type exposes the following members.

Properties

	Name	Description
	CoolingSetPoint	Gets or sets the requested temperature for the cooling subsystem (Inherited from ITemperatureController.)
	CurrentSetPoint	Gets or sets the currently reqeusted temperature of the HVAC system (Inherited from ITemperatureController.)
	FanControl	Gets or sets the fan state (true=on, false=off) (Inherited from IClimateControl.)
	FanStatus	Gets the current fan running status (true=on, false = off) (Inherited from IClimateControl.)
	HeatingSetPoint	Gets or sets the requested temperature for the heating subsystem (Inherited from ITemperatureController.)
	HeatingStatus	Gets the status of the heating component (Inherited from IClimateControl.)
	Mode	Get schedule status for the HVAC (Inherited from IClimateControl.)
	OutdoorTemperature	Gets the outdoor temperature (Inherited from ITemperatureController.)
	SetbackMode	Gets or sets the setback mode (Inherited from ITemperatureSetback.)
	SetbackOffset	Gets or sets the offset for the setback (Inherited from ITemperatureSetback.)
	State	Gets or sets the system state (Inherited from IClimateControl.)
	Temperature	Gets the measured temperature (Inherited from ITemperatureSensor.)
	TemperatureMode	Gets or sets the temperature mode (Inherited from IClimateControl.)
	UserHold	Gets or sets the command to freeze temperature at the current set point (Inherited from ITemperatureController.)

IDigitalOutput

Defines the single digital input/output capability

The **IDigitalOutput** type exposes the following members.

Properties

	Name	Description
	State	Sets or gets the state of the output

IInfraredOutput

When implemented, provides infrared output capability

The **IInfraredOutput** type exposes the following members.

Properties

	Name	Description
	ContinuousSend	Continuously send TXData until set to false
	Send	Momentary, initiates send when set to true
	TxData	IR data in Pronto (CCF) remote control format to be sent
	TxRepeat	Number of times to repeat the transmitted codes

Methods

	Name	Description
	Transmit	Sends the encoded CCF string to an IR transmitter device

ISerialPort

Defines the Serial Port capability

The **ISerialPort** type exposes the following members.

Properties

	Name	Description
	BaudRate	Defines the speed (baud rate) for the serial port
	BytesAvailable	Number of bytes available in the read stream
	CTS	Sets the Clear to send bit
	DataBits	Defines the number of data bits to use for the serial port
	DataReceived	Momentary, set when new data is available from the serial port.
	DSR	Sets the Data Set Ready bit
	DTR	Sets the Data Terminal Ready bit
	EndOfMessageMarker	Defines the character that the driver will use to determine if a received message is complete

	ErrorReceived	Momentary, set when the serial port detects an error
	FlowControl	Defines the desired flow-control for the serial port
	IsOpen	Determines if serial port is in-use (true) or available (false)
	IsTextOriented	Determines if the serial port exclusively sends standard printable characters (true) versus binary (false)
	Parity	Defines the parity setting for the serial port
	Reset	Momentary, initiates a reset of the port
	RTS	Sets the request to send bit
	StatusCTS	Reads the status of the Clear to send bit
	StatusDSR	Reads the status of the Data set ready bit
	StatusRing	Reads the status of the Ring bit
	StopBits	Defines the number of stop bits for the serial port

Methods

	Name	Description
	Close	Closes the serial port
	Open	Opens the serial port
	Read	Reads data from the serial port
	ReadLine	Reads a line of text from the serial port
	Write(String)	Writes a line of text without a terminator character
	Write(Byte[], Int32, Int32)	Writes a byte buffer to the serial port
	WriteLine	Writes a line of text to the serial port with a termination character

IColoredLighting

Defines lighting capable of displaying multiple colors

The **IColoredLighting** type exposes the following members.

Properties

	Name	Description
	BlueLevel	Blue component of the desired color
	GreenLevel	Green component of the desired color
	RedLevel	Red component of the desired color

IDimmable

Defines the dimmer capability

The **IDimmable** type exposes the following members.

Properties

	Name	Description
	Brightness	Level of lighting between 0.0 and 1.0
	PresetDimLevel	Level to set lighting when the PowerState is true between 0.0 and 1.0
	RampRate	Rate in seconds at which to get desired lighting level

IAudioControl

Defines the audio control capability

The **IAudioControl** type exposes the following members.

Properties

	Name	Description
	Balance	Left-right levels between 0.0(full-left) and 1.0(full-right). Centered at 0.5
	Bass	Low-end Bass level between 0.0 and 1.0
	Loudness	Sets Loudness response between 0.0 and 1.0
	Mute	Immediately set volume to zero when set to true and return to previous level when set to false
	Treble	High-end Treble level between 0.0 and 1.0
	Volume	Sound pressure level between 0.0 and 1.0

IAV_Component

Defines the AV component capability

IAvailability

Defines the Availability capability

The **IAvailability** type exposes the following members.

Properties

	Name	Description
	CurrentUse	Resource currently allocating this device

IAVInput

Defines the AV input capability

The **IAVInput** type exposes the following members.

Properties

	Name	Description
	Balance	Left-right levels between 0.0(full-left) and 1.0(full-right). Centered at 0.5 (Inherited from IAudioControl.)
	Bass	Low-end Bass level between 0.0 and 1.0 (Inherited from IAudioControl.)
	BoundObject	Object bound to this object (Inherited from IBindable.)
	CurrentTrack	Media currently loaded into the transport (Inherited from ITransportControl.)
	CurrentUse	Resource currently allocating this device (Inherited from IAvailability.)
	Duration	Duration of the media currently loaded on the transport (Inherited from ITransportControl.)
	Loudness	Sets Loudness response between 0.0 and 1.0 (Inherited from IAudioControl.)
	MediaFinishedPlaying	Momentary. Raised when currently selected media is finished playing and the play state of the device is true (Inherited from IMediaPlayer.)
	Mute	Immediately set volume to zero when set to true and return to previous level when set to false (Inherited from IAudioControl.)
	Pause	Temporarily suspends playback when set to true, resumes playback from current media position when set to false (Inherited from ITransportControl.)
	Play	Initiates playback when set to true, stops transport when set to false (Inherited from ITransportControl.)
	Progress	Percentage of current media processed by the transport (Inherited from ITransportControl.)
	SupportedMedia	List of supported media types separated by commas. MIME types (Inherited from ISupportedMedia.)
	TransportCommand	Sets desired state for the underlying transport (Inherited from ITransportControl.)
	Treble	High-end Treble level between 0.0 and 1.0 (Inherited from IAudioControl.)
	Volume	Sound pressure level between 0.0 and 1.0 (Inherited from IAudioControl.)

Methods

	Name	Description
	Next	Commands the transport to play the next media item (Inherited from ITransportControl.)
	Previous	Commands the transport to play the previous media item (Inherited from ITransportControl.)

IFader

Defines the Fader capability

The **IFader** type exposes the following members.

Properties

		Name	Description
▤		Fade	Sets front to back ratio for the playback (0.0 full front, 1.0 full back)

IMatrixSwitcher

Defines the matrix switcher capability

The **IMatrixSwitcher** type exposes the following members.

Properties

	Name	Description
▤	NumberOfInputs	Number of matrix switcher inputs
▤	NumberOfOutputs	Number of matrix switcher outputs

Methods

	Name	Description
◆	GetAvailableInputs	Returns the unallocated inputs that are capable of playing the specified media item
◆	GetAvailableInputsForContent	Returns the unallocated inputs that are capable of playing the specified Content Note that content is a collection of media items.
◆	GetCurrentlyMappedInput	Returns the input currently mapped to the specified output
◆	GetCurrentlyMappedOutput	Returns the currently mapped output to the specified input
◆	Map	Maps an input to an output
◆	RegisterInput	Registers an input to the matrix switcher
◆	RegisterOutput	Register an output to the matrix switcher
◆	SendMessageToZone	For hardware that supports it, sends a message to a switcher zone that can be ultimately displayed to the end user

IMatrixSwitcherZone

Defines the matrix switcher zone capability

The **IMatrixSwitcherZone** type exposes the following members.

Properties

	Name	Description
▤	Balance	Left-right levels between 0.0(full-left) and 1.0(full-right). Centered at 0.5 (Inherited from IAudioControl.)
▤	Bass	Low-end Bass level between 0.0 and 1.0 (Inherited from IAudioControl.)
▤	BoundObject	Object bound to this object (Inherited from IBindable.)

	CurrentTrack	Media currently loaded into the transport (Inherited from ITransportControl.)
	Duration	Duration of the media currently loaded on the transport (Inherited from ITransportControl.)
	Loudness	Sets Loudness response between 0.0 and 1.0 (Inherited from IAudioControl.)
	MediaFinishedPlaying	Momentary. Raised when currently selected media is finished playing and the play state of the device is true (Inherited from IMediaPlayer.)
	Mute	Immediately set volume to zero when set to true and return to previous level when set to false (Inherited from IAudioControl.)
	Pause	Temporarily suspends playback when set to true, resumes playback from current media position when set to false (Inherited from ITransportControl.)
	Play	Initiates playback when set to true, stops transport when set to false (Inherited from ITransportControl.)
	PowerState	Power state, On if true, Off if false (Inherited from IPowerState.)
	Progress	Percentage of current media processed by the transport (Inherited from ITransportControl.)
	SupportedMedia	List of supported media types separated by commas. MIME types (Inherited from ISupportedMedia.)
	TransportCommand	Sets desired state for the underlying transport (Inherited from ITransportControl.)
	Treble	High-end Treble level between 0.0 and 1.0 (Inherited from IAudioControl.)
	Volume	Sound pressure level between 0.0 and 1.0 (Inherited from IAudioControl.)

Methods

	Name	Description
	Disconnect	Disconnect any input associated with this zone
	FindAndConnect	Finds an input compatible with the current media and connects it to the output
	Next	Commands the transport to play the next media item (Inherited from ITransportControl.)
	Previous	Commands the transport to play the previous media item (Inherited from ITransportControl.)

IMediaPlayer

Defines the Media player capability

The **IMediaPlayer** type exposes the following members.

Properties

	Name	Description
	Balance	Left-right levels between 0.0(full-left) and 1.0(full-right). Centered at 0.5 (Inherited from IAudioControl.)
	Bass	Low-end Bass level between 0.0 and 1.0 (Inherited from IAudioControl.)
	CurrentTrack	Media currently loaded into the transport (Inherited from ITransportControl.)
	Duration	Duration of the media currently loaded on the transport (Inherited from ITransportControl.)
	Loudness	Sets Loudness response between 0.0 and 1.0 (Inherited from IAudioControl.)
	MediaFinishedPlaying	Momentary. Raised when currently selected media is finished playing and the play state of the device is true
	Mute	Immediately set volume to zero when set to true and return to previous level when set to false (Inherited from IAudioControl.)
	Pause	Temporarily suspends playback when set to true, resumes playback from current media position when set to false (Inherited from ITransportControl.)
	Play	Initiates playback when set to true, stops transport when set to false (Inherited from ITransportControl.)
	Progress	Percentage of current media processed by the transport (Inherited from ITransportControl.)
	SupportedMedia	List of supported media types separated by commas. MIME types (Inherited from ISupportedMedia.)
	TransportCommand	Sets desired state for the underlying transport (Inherited from ITransportControl.)
	Treble	High-end Treble level between 0.0 and 1.0 (Inherited from IAudioControl.)
	Volume	Sound pressure level between 0.0 and 1.0 (Inherited from IAudioControl.)

Methods

	Name	Description
	Next	Commands the transport to play the next media item (Inherited from ITransportControl.)
	Previous	Commands the transport to play the previous media item (Inherited from ITransportControl.)

IMediaQueue

Implements the media queuing capability

The **IMediaQueue** type exposes the following members.

Properties

	Name	Description
	IsQueueEmpty	Determines if que is empty (true) or populated (false)

Methods

	Name	Description
	AddItem	Adds an item to the queue
	ClearQueue	Clears the queue
	GetQueue()	Returns the current queue
	GetQueue(Int32)	Return the specified item in the queue
	IsQueueable	Determines if the implemented device can queue the specified media
	RemoveItem	Remove item from queue

ISupportedMedia

Defines the supported media capability

The **ISupportedMedia** type exposes the following members.

Properties

	Name	Description
	SupportedMedia	List of supported media types separated by commas. MIME types

ITransportControl

Implements the transport control capability

The **ITransportControl** type exposes the following members.

Properties

	Name	Description
	CurrentTrack	Media currently loaded into the transport
	Duration	Duration of the media currently loaded on the transport
	Pause	Temporarily suspends playback when set to true, resumes playback from current media position when set to false
	Play	Initiates playback when set to true, stops transport when set to false
	Progress	Percentage of current media processed by the transport
	TransportCommand	Sets desired state for the underlying transport

Methods

	Name	Description
	Next	Commands the transport to play the next media item
	Previous	Commands the transport to play the previous media item

ITuner

Defines the tuner capability

The **ITuner** type exposes the following members.

Properties

	Name	Description
	Frequency	Sets the frequency to which to set the tuner
	RDSMessage	Returns the currently available Radio Data Service (RDS) message
	TunerBand	Sets the tuner band
	TunerCommands	Sets the tuner command

Methods

	Name	Description
	GetPreset	Returns a preset
	GetPresetFrequency	Returns the requency of a preset
	GetPresets	Returns all current presets
	SetPreset(TunerPreset)	Creates a preset using the preset class
	SetPreset(Int32, Double)	Creates a preset

IPowerState

Defines PowerState capability

The **IPowerState** type exposes the following members.

Properties

	Name	Description
	PowerState	Power state, On if true, Off if false

ICODetector

Defines Carbon Monoxide detector capability

The **ICODetector** type exposes the following members.

Properties

	Name	Description
	CODetected	Determines if Carbon monoxide is present (true) or not (false)

ISmokeDetector

Defines the Smoke Detector capability

The **ISmokeDetector** type exposes the following members.

Properties

	Name	Description
	SmokeDetected	Determines if smoke is present (true) or not (false)

IWaterSensor

Defines the Water sensor capability

The **IWaterSensor** type exposes the following members.

Properties

	Name	Description
	WaterDetected	Determines if water is present (true) or not (false)

IGarageDoorSensor

Defines the garage door sensor

The **IGarageDoorSensor** type exposes the following members.

Properties

		Name	Description
		GarageDoorOpened	Opened if true, closed if false

IMotionDetector

Implements the motion detector capability

The **IMotionDetector** type exposes the following members.

Properties

	Name	Description
	LastTimeTriggered	The date/time the last time motion was detected

	MotionDetected	Determines if motion has been detected (true) within the sampling period or not (false)

IPresenceDetector

Defines the presence detector capability

The **IPresenceDetector** type exposes the following members.

Properties

	Name	Description
	LastPresenceChanged	The date/time the last presence changed occurred
	PresenceChanged	Set when the detected entities has changed

Methods

	Name	Description
	GetDetections	List of presence detections

ITTSStream

Defines the Text to speech stream capability

The **ITTSStream** type exposes the following members.

Methods

	Name	Description
	Play	Plays the specified IO stream

ITTSText

Defines the Text to Speech text capability

The **ITTSText** type exposes the following members.

Methods

	Name	Description
	Speak	Converts the specified text to an audio stream and plays

IAlarmControl

Defines the Alarm control capability

The IAlarmControl type exposes the following members.

Properties

	Name	Description
	AlarmCode	Code used to disarm the alarm system
	Armed	Determines if the controller will trigger an alarm if it detects a zone event
	PanicButton	Triggers an alarm regardless of Armed setting

Methods

	Name	Description
	SetSiren	Tells a bound object to trigger an audible alarm (if able)
	Signal	Called by a bound driver to indicate a security event has occurred

Disclaimers

General

THE SOFTWARE IS PROVIDED "AS IS" AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, GEORGECO PROFESSIONAL SERVICES, LLC (GEORGECO) AND ITS SUPPLIERS DISCLAIM ALL WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES AND CONDITIONS OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR A PARTICULAR PURPOSE, AND ANY WARRANTIES AND CONDITIONS ARISING OUT OF COURSE OF DEALING OR USAGE OF TRADE. NO ADVICE OR INFORMATION, WHETHER ORAL OR WRITTEN, OBTAINED FROM OR ELSEWHERE WILL CREATE ANY WARRANTY OR CONDITION NOT EXPRESSLY STATED IN THIS AGREEMENT.

GeorgeCo does not warrant that the Products and Services are defect or error free, or that the operation of the Products and Services will be uninterrupted or 100% available. The Products and Services rely in part on factors outside of GeorgeCo's control, including the transmission of data through your local area network (LAN), third-party cloud services, mobile devices, and internet access. These factors may result in the Products and Services being unreliable or unavailable. GeorgeCo does not guarantee that you will receive notifications when expected or at all. GeorgeCo cannot provide specific information related to a situation in your home or elsewhere. It is your responsibility to educate yourself on how to respond to an emergency according to the specifics of your situation.

Alarm controller, Konnected hardware and Noonlight monitoring

A valid Noonlight subscription must be purchased and product key provided for the GHA Noonlight driver to function. This driver is provided as-is without guarantees or warranties of any kind. Using Noonlight with George Home Automation involves multiple service providers and potential points of failure, including (but not limited to) your home network, your internet service provider, 3rd party cloud services such as Microsoft Azure, Amazon Web Services, services operated by GeorgeCo Professional Services, LLC and Noonlight.

You must read and understand the Noonlight terms of use (https://www.noonlight.com/terms) which includes important limitations of liability and use.

Use of this driver constitutes your understanding and acceptance of these and other terms that are found in your George Home Automation License Agreement.

If you choose to not utilize a third-party monitoring service, your use of this product constitutes your agreement to the following statement: THE PROUDCTS AND SERVICES ARE PROVIDED FOR INFORMATIONAL PURPOSES ONLY, AND ARE NOT A SUBSTITUTE FOR A THIRD-PARTY MONITORED EMERGENCY NOTIFICATION SYSTE.