Appendix 7
HCA Objects

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

- Introduction
- Example
- HCA Object model

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

Introduction

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

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

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

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

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

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

Example

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

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

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

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

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

HCA Object Model

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

Error Handling

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

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

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

Controller and Device Objects

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

Obsolete Methods

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

HCA Object

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

Name	**GetHomeModeCount**
Description	Get count of home modes
Parameters	None
Return Value	Short: Count

Name	**GetHomeModeName**
Description	Gets the ith mode name
Parameters	Short i
Return value	VARIANT: Mode name

Name	**GetCurrentHomeMode**
Description	Returns the current mode
Parameters	None
Return Value	VARIANT: Mode name

Name	**SetCurrentHomeMode**
Description	Changes the home mode
Parameters	BSTR: New mode name
Return Value	Short: Result code

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

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

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

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

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

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

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

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

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

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

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

Name	**ModificationTimeV**
Description	Same as ModificationTime but returns it as a VT_DATE variant
Parameters	None
Return Value	VARIANT: Time

Name	**FolderCount**
Description	Returns the number of folders in the current design
Parameters	None
Return Value	Short: Count

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

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

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

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

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

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

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

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

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

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

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

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

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

wParam	lParam	Use
1	0	HCA is terminating

Name	**RemoveRemoteAccessCommWindow**
Description	Removes a previously installed comm. window handle
Parameters	None
Return value	Short: Result code

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

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

Device, Group, Controller common methods

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Name	**UPBBlink**
Applies to	DC
Description	Send a UPB Blink command
Parameters	BSTR: Device name
	Short: Rate
Return Value	Short: Return code

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Name	**KeyLabel**
Applies to	DC
Description	Returns the label of the ith button or rocker for this

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

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

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

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

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

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

HCA.Device Object

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

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

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

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

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

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

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

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

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

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

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

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

Name	**IRKeypadButtonName**
Description	Returns the name of the ith button
Parameters	BSTR: Device name
	Short: i
Return Value	VARIANT: Name

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

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

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

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

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

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

HCA.Program Object

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

Name	**Start**
Description	Starts the program running
Parameters	BSTR: Program name
Return Value	Short: Result code

Name	**Stop**
Description	If running, terminates the program
Parameters	BSTR: Program name
Return Value	Short: Result code

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

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

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

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

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

HCA.Group Object

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

HCA.Controller Object

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

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

HCA.Schedule Object

The Schedule object contains methods to work with schedules.

Name	**SetCurrent**
Description	Makes a schedule the current schedule
Parameters	BSTR: Schedule name
Return Value	Short: Result code

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

Name	**Count**
Description	Returns the count of schedules
Parameters	None
Return Value	Short: Count

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

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

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

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

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

HCA.Flag Object

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

Name	**Count**
Description	Returns the count of flags
Parameters	None
Return Value	Short: Count

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

Name	**Get**
Description	Returns the value of the flag
Parameters	BSTR: Flag name
Return Value	VARIANT: Value

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

Note on error return: The Get method can return an error like most methods. You can tell if you get an error return by looking at the type of the variant. In the type is VT_I4 then the variant contains an error code.

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

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

Name	**SetNumber**
Description	Sets the value of the flag
Parameters	BSTR: Flag name
	double: value
Return Value	Short: Result code

Name	**SetInt**
Description	Sets the value of the flag
Parameters	BSTR: Flag name
	int: value
Return Value	Short: Result code

Name	**SetShort**
Description	Sets the value of the flag
Parameters	BSTR: Flag name
	short: value
Return Value	Short: Result code

Name	**SetText**
Description	Sets the value of the flag
Parameters	BSTR: Flag name
	BSTR: value
Return Value	Short: Result code

Name	**SetDate**
Description	Sets the value of the flag
Parameters	BSTR: Flag name
	DATE: value
Return Value	Short: Result code

Name	**SetYesNo**
Description	Sets the value of the flag
Parameters	BSTR: Flag name
	bool: value
Return Value	Short: Result code

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

Name	**GetNumber**
Description	Returns the value of the flag
Parameters	BSTR: Flag name
Return Value	double: value

Name	**GetInt**
Description	Returns the data type of the flag
Parameters	BSTR: Flag name
Return Value	int: value

Name	**GetShort**
Description	Returns the value of the flag
Parameters	BSTR: Flag name
Return Value	short: value

Name	**GetText**
Description	Returns the value of the flag
Parameters	BSTR: Flag name
Return Value	VARIANT: value

Name	**GetDate**
Description	Returns the value of the flag
Parameters	BSTR: Flag name
Return Value	VARIANT: value

Name	**GetYesNo**
Description	Returns the value of the flag
Parameters	BSTR: Flag name
Return Value	bool: value

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

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

Note on error return: An error could be returned in the normal VT_I4 way. For example, password problems would be returned this way. But an error could also occur when the expression text is parsed or when the expression is evaluated. In either case an error string is returned (VT_BSTR) with the first character of that string always a "~". A way to test for an error would be:

```
Dim value
value = hca.flag.evaluate(expressionText)

If (VarType(value) = vbString) Then
   If Left(value, 1) = "~" Then
      MsgBox "Expression evaluation error: " + Mid(value, 2)
   End If
End If
```

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

EvaluateEx returns a string of the form:

<a> , <b> , <c>

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

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

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

HCA.Log Object

Name	**Open**
Description	Opens the HCA log for reading
Parameters	None
Return Value	Short: Result code

Name	**Close**
Description	Close the log previously opened with Open
Parameters	None
Return Value	Short: Result code

Name	**Count**
Description	Returns the number of entries in the log
Parameters	None
Return Value	Short: Count

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

Name	**FilterCount**
Description	Returns the number of filters available
Parameters	None
Return Value	Short: Count

Name	**FilterName**
Description	Returns the name of the ith filter
Parameters	Short: i
Return Value	VARIANT: Filter name

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

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

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

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

Name	**InspectorMessageCheck**
Description	Check-off the ith inspector message. This is the same action as you can do when the troubleshooter is open and you click inside the [] box on the Inspector tab.
Parameters	Short: i
	Short: Check
Return Value	Short: Result code

Name	**AddLogEntry**
Description	Adds an entry to the HCA log
Parameters	BSTR: Text
Return Value	Short: Result code

HCA.Display Object

Name	**Count**
Description	Returns the count of displays
Parameters	None
Return Value	Short: Count

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

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

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

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

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

Name	**Delete**
Description	Deletes a display by name
Parameters	BSTR: Display name
Return Value	Short: Result code

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

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

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

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

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

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

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

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

Name	**GetHTMLPath**
Description	Returns the path to the HTML "result" file for a HTML display
Parameters	BSTR: Display name
Return Value	VARIANT: Path

Name	**GetDXFBackgroundPath**
Description	Returns the pathname for a display's DXF background
Parameters	BSTR: Display name
Return Value	VARIANT: Path

Name	**GetImageBackgroundPath**
Description	Returns the pathname for a display's image file background
Parameters	BSTR: Display name
Return Value	VARIANT: Path

Name	**GetCurrentDisplay**
Description	Returns the name of the current display
Parameters	None
Return Value	VARIANT: Display name

Name	**SetCurrentDisplay**
Description	Changes the current display to the named display
Parameters	BSTR: New display name
Return Value	VARIANT: Old display name

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

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

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

Name	**State**
Description	Return state of room
Parameters	BSTR: Object name
Return Value	Short: state
	0 = Off
	1 = On

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

Passwords

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

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

- Control
- Design
- Remote

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

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

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

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

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

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

Internally, the ValidateAccess function does this:

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

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

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

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

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

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

An application must do these things:

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

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

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

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