HCA 201 – Part 4

During the last three weeks we looked at a tiled display to show the status of the rooms, alerts, and interfaces in a design. The first week we created a tiled display for this, the second week we created the programs to update it, and in the third we improved those programs by adding additional function. Now we can turn to the last part: getting it ready to share with others.

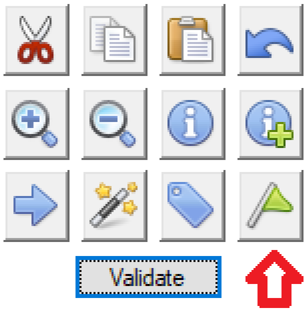
No magic here, just a series of small steps to take. Let's go through each one.

But first: References that may help if this goes into areas of HCA that you have not yet worked with:

Import Export User Guide Chapter:
http://www.homecontrolassistant.com/download/V15/Doc/27_Import%20Export.pdf

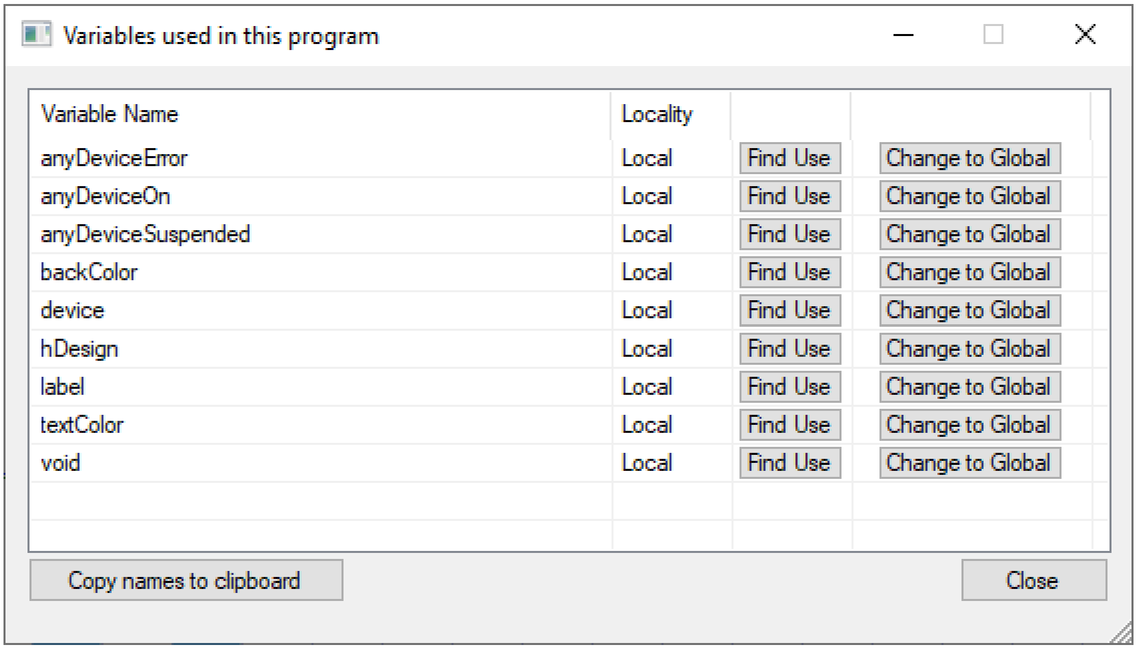
V15 release notes on the online library
http://www.homecontrolassistant.com/download/V15/V15OnlineLibrary.pdf

Task 1. Check for unneeded global variables



In previous weeks we discussed local and global variables. Now is the time to check all your programs to make sure that there are no global variables that don't have to be global. All programs should be checked to see if any variables used in the program are global, but could be made local.

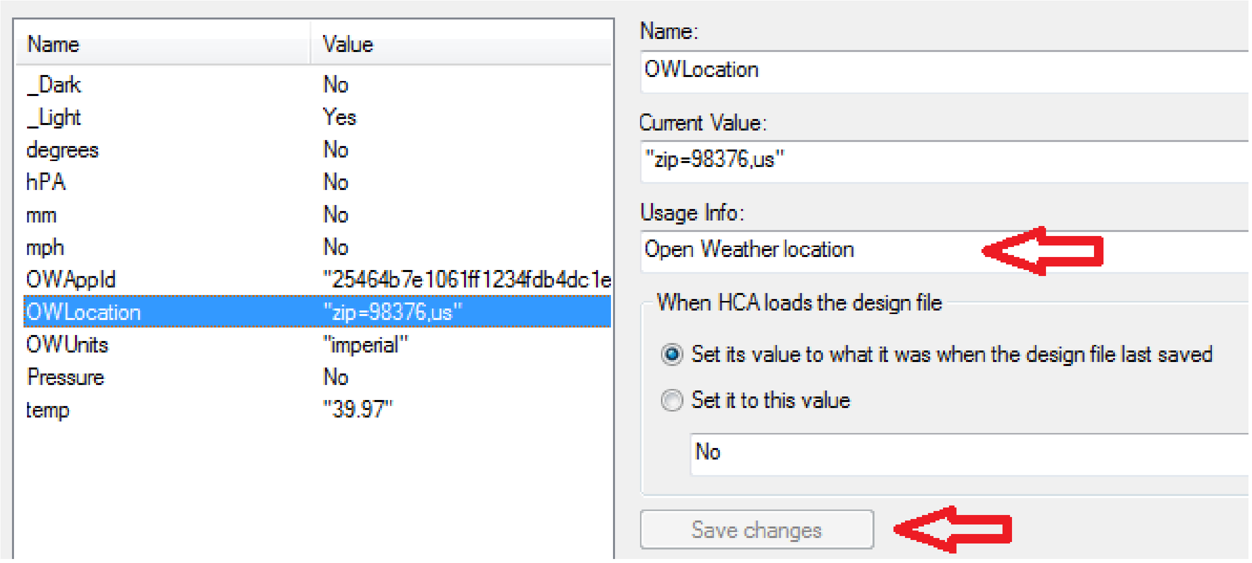
Open each program and on the Visual Programmer tab you can open the variables list and see which are local and which are global:


Variables used in this program

Variable Name	Locality		
anyDeviceError	Local	Find Use	Change to Global
anyDeviceOn	Local	Find Use	Change to Global
anyDeviceSuspended	Local	Find Use	Change to Global
backColor	Local	Find Use	Change to Global
device	Local	Find Use	Change to Global
hDesign	Local	Find Use	Change to Global
label	Local	Find Use	Change to Global
textColor	Local	Find Use	Change to Global
void	Local	Find Use	Change to Global

Task 2: Add Global variable comments

All global variables can have notes associated with them to explain their usage. Open the variable inventory and select the variable and fill in notes on what it is used for.



Name	Value
_Dark	No
_Light	Yes
degrees	No
hPA	No
mm	No
mph	No
OWAppId	"25464b7e1061ff1234fdb4dc1e
OWLocation	"zip=98376,us"
OWUnits	"imperial"
Pressure	No
temp	"39.97"

Name: OWLocation

Current Value: "zip=98376,us"

Usage Info: Open Weather location

When HCA loads the design file

● Set its value to what it was when the design file last saved

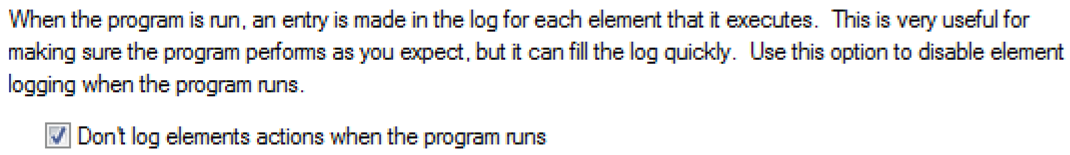
○ Set it to this value

No

If you make a change to add or modify the text, don't forget to press the "Save Changes" button before going on to the next variable.

Task 3: Disable Logging on any programs in the package

After you have tested the programs then make sure that logging is disabled. This is done on the program's "Log" tab. You don't want others to have their logs filled with entries from your programs.



When the program is run, an entry is made in the log for each element that it executes. This is very useful for making sure the program performs as you expect, but it can fill the log quickly. Use this option to disable element logging when the program runs.

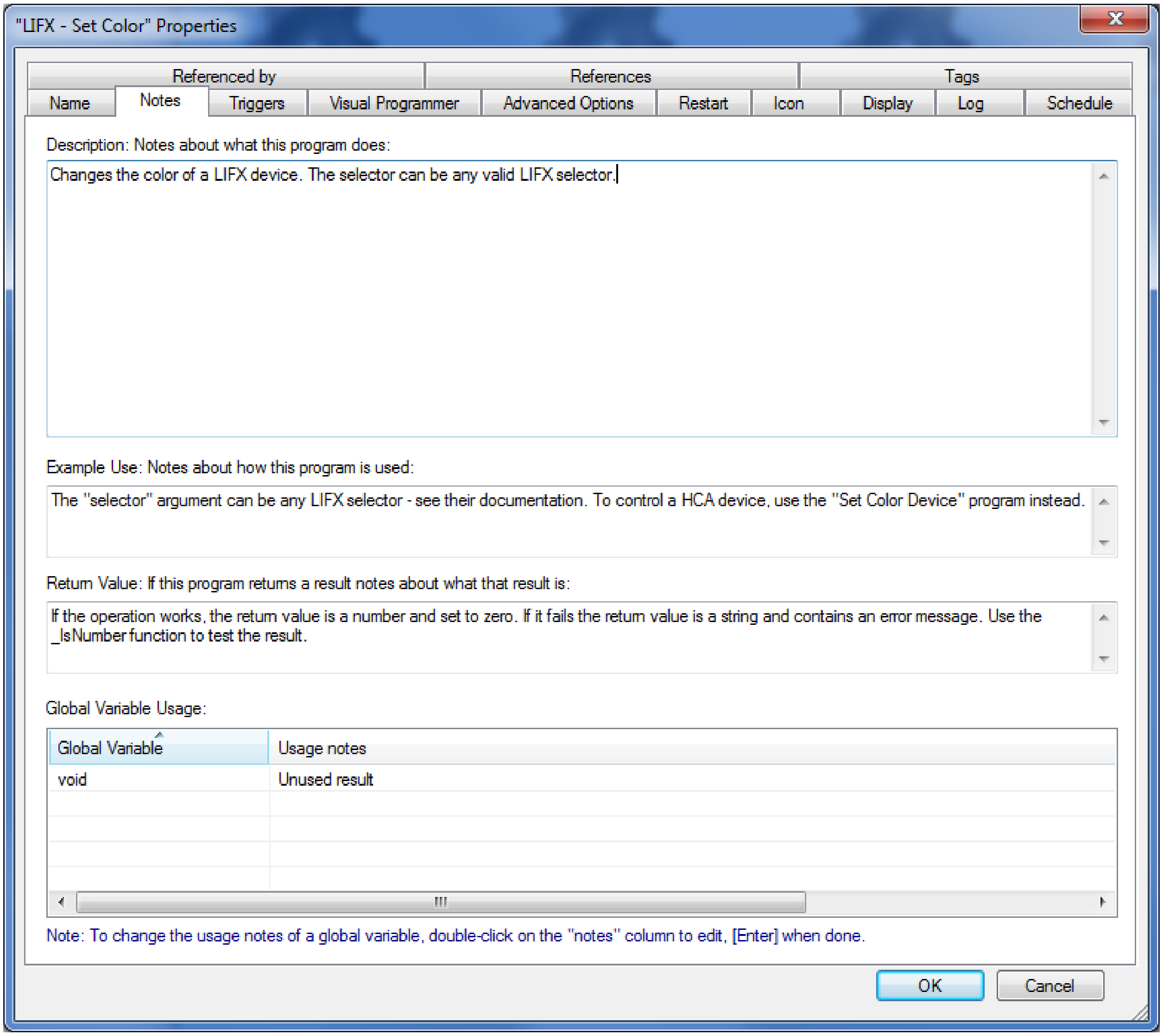
☑ Don't log elements actions when the program runs

Since the program now don't log, make sure that any problems or errors that must be reported to the user are made by either creating an Alert using the program function _AlertAdd, or the Add-to-log programmer element.

Task 4: Create Program documentation

Even if you will not be sharing these programs, it is a good idea to make some notes as to their function. You might have created the program for your own use but if you come back to it maybe months later, you will forget many things.

For each program, open its properties and chose the *Notes* tab



Description: Notes about what this program does:

Changes the color of a LIFX device. The selector can be any valid LIFX selector.

Example Use: Notes about how this program is used:

The "selector" argument can be any LIFX selector - see their documentation. To control a HCA device, use the "Set Color Device" program instead.

Return Value: If this program returns a result notes about what that result is:

If the operation works, the return value is a number and set to zero. If it fails the return value is a string and contains an error message. Use the _IsNumber function to test the result.

Global Variable Usage:

Global Variable	Usage notes
void	Unused result

Note: To change the usage notes of a global variable, double-click on the "notes" column to edit, [Enter] when done.

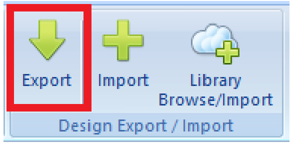
On this tab you can provide all the information about the program function, how it should be used, if it generates a result, and any information about the use of any global variables. This is another location where the global variable info, also shown in the variable inventory, can be viewed and modified.

Task 5: Add element Comments

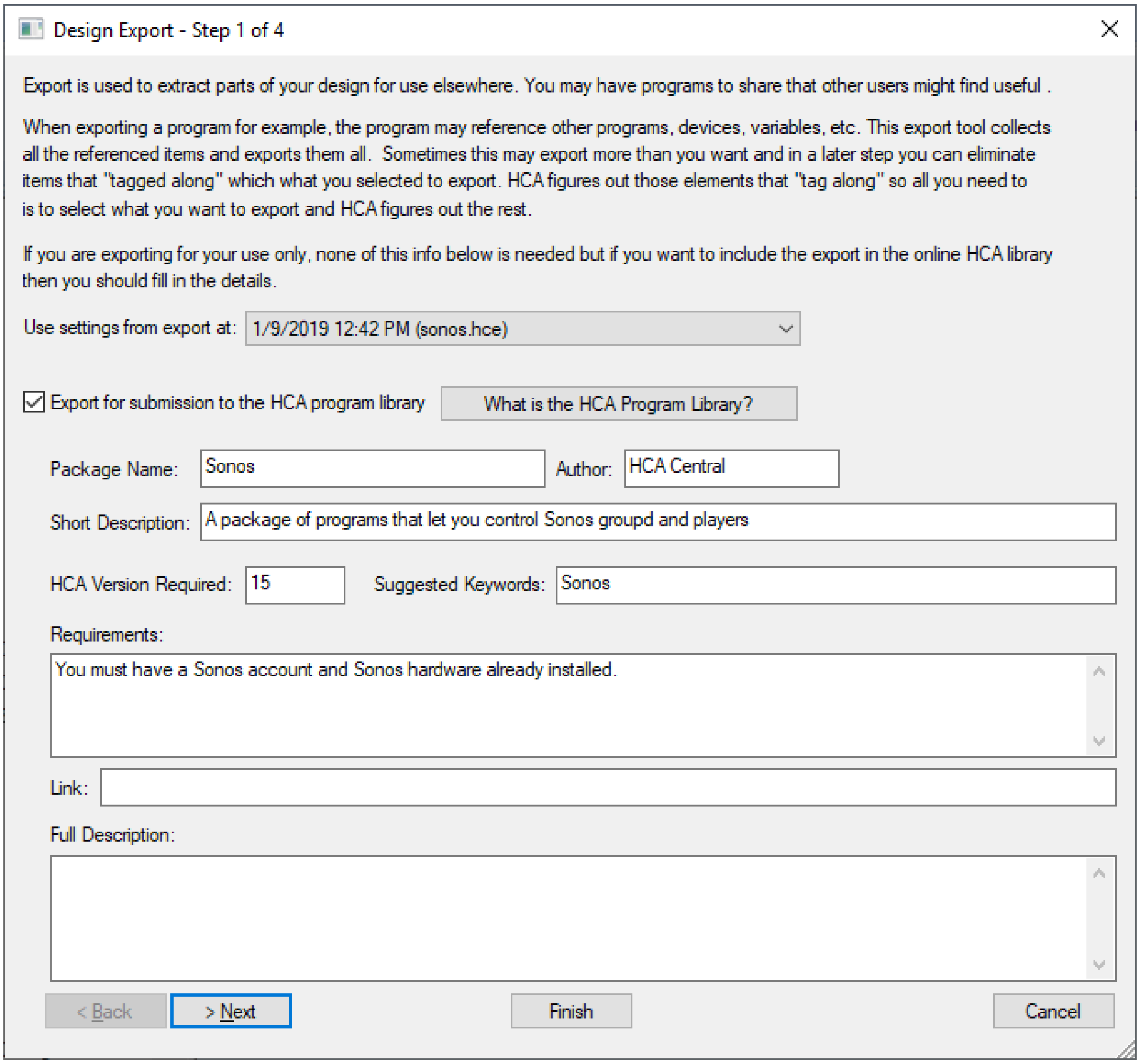
The final level of detail, and this is optional, is to add any comments you may have about the action of program elements in implementing the program. In the Visual Programmer you can right-click on an element and select “Comment” from the popup menu to open a dialog where you can add notes about that element.

Task 6: Do the export, but first what's a package?

Now that you have completed these tasks, it is finally time to discuss the export. Each item in the library is called a "package". A "package" contains one or more programs that work together for one task. For example, the Sonos package has 19 programs. It is important that a package doesn't contain any programs you may have created for testing or any test devices. It is best if all the programs that form the package be contained in the same folder. That folder name is important as it is the name of the folder created when a user imports the package.



When everything is ready, start the export process from the Design ribbon category. The package name and short description are the most important as that is what users see when they browse the library. The other fields are optional.



Design Export - Step 1 of 4

Export is used to extract parts of your design for use elsewhere. You may have programs to share that other users might find useful .

When exporting a program for example, the program may reference other programs, devices, variables, etc. This export tool collects all the referenced items and exports them all. Sometimes this may export more than you want and in a later step you can eliminate items that "tagged along" which what you selected to export. HCA figures out those elements that "tag along" so all you need to is to select what you want to export and HCA figures out the rest.

If you are exporting for your use only, none of this info below is needed but if you want to include the export in the online HCA library then you should fill in the details.

Use settings from export at: 1/9/2019 12:42 PM (sonos.hce)

☑ Export for submission to the HCA program library

Package Name: Sonos

Author: HCA Central

Short Description: A package of programs that let you control Sonos groupd and players

HCA Version Required: 15

Suggested Keywords: Sonos

Requirements: You must have a Sonos account and Sonos hardware already installed.

Link:

Full Description:

The next steps of the export are as described in the import-export chapter of the user guide. It’s a good idea to check that as you select the programs of your package for export that nothing unexpected “tags along” in the export list. When you reach the last step of the export wizard and save to the export file, for a class package the file type is .hclass, and for non-class exports the file type is .hce.

Also created with the hce/hclass file is a second file with the package name and the file type .json that holds a synthesis of all your package and program documentation.

Final Task: Library submission

If you have created something useful that you want to share, please let us know! Send us a message and tell us about your package and attach the .hce or .hclass file and the .json file and we will take a look.

##end##