Page 3 - JavaScript Modules

Residential iPad User Interface System - Page 3

All information and GUI's in this case study are by Barry Gordon himself. These are put together to suit his unique requirements and are designed to reflect his personal tastes. All visual and back end system design are Barry's own and should be looked as an example, CommandFusion recommends you design a system that suits your own requirements and tastes.

This case study is divided up into 3 pages:

1. System Overview - a detailed overview of the house, and what makes up the system.
2. Interface Display Pages - a detailed look at the iViewer 4 GUI
3. JavaScript Modules - info on the JavaScript modules that Barry uses to provide additional functionality to his system

General Project Design

I am a programmer (coder) at heart. I have always liked to build things. I do cabinetry work, building furniture when the need arises. My late wife liked this aspect of my personality. She would take pictures of my workshop into Home Depot at gift giving times and merely ask “What is he missing?”. I built my own stables and corrals when I was into horses, dog houses and pens when I was into showing dogs. Jill, my late wife, told me not to take up golf as building golf courses was too expensive and time consuming. Programming is an avocation that allows one to build very complex things at relatively low cost and get significant gratification when accomplished.

By “programming” I do not mean the placement of graphics on a page, but rather the writing of code in a recognized programming language. One of the major reasons I chose CommandFusion for my HA efforts, and the Pronto PRO before that, was that they both were based on and reasonably made use of a major programming language to enhance capability and functionality. In fact I had a lot of code re-use between the Pronto PRO and iViewer because I dealt with them in the same manner doing all the work in Javascript.

My “style” in using CF is to have all actions, whenever possible performed by referencing a Javascript code function to perform the work. An excellent example of this is the handling of TCP communications for intractable external systems. It seems that many software designers who work for major component manufacturers (e.g. Denon) have a philosophy of “My way, or the Highway”. Because of the Javascript capability in Command Fusion I was able to handle some pretty unreasonable (IMHO) external systems.

To the credit of the CF team, when I told them what I was going through they added basic functionality that relieved/resolved the issues. I process all feedback in Javascript, using the simplest regex possible for that external system. I do all message detection and isolation in Javascript. In this way I have no one to blame but myself when things don't work. There is a large amount of commonality in the process of detecting a message and isolating it form an input stream. Once a message is isolated, the processing of its content becomes device specific, but can be handled quite easily using the Javascript concept of a sieve, the switch statement.

I try to write all my systems to be table driven to the greatest degree possible. Changing the contents of tables is much less error prone than changing code!

Dynamic Configuration Capability:
The entire system is dynamically configured by the contents of the module “Configuration.js”. This module has several classes of

items.

Control:	Defines the names of the sub-objects in CFG.
Interface:	Defines the external systems and how they are interconnected.
IR Files:	Contains all the data needed by an iTach to send a named IR pattern to a device.
Station Lists:	Contains the information to build lists of TV channels or radio stations.

The Interface object looks as follows:

Interface_config:

```
{"systems":[    {"name":"pageToStart", "device":"iPad", "page":"HC_HomeControl" },

                {"name":"IRsystems", "installed":true, "converters":[{"model":"Global Cache",
                        "device":"GC1"}, {"model":"Global Cache", "device":"GC2"} ] },

                {"name":"iPadPAL", "installed":true, "stack":"ProntoPal",
                            "IP":"192.168.1.204", "protocol":"TCP", "port":"8000" },

                {"name":"Theater", "installed":true, "stack":"Theater",
                        "IP":"192.168.1.205", "protocol":"TCP", "port":"7011" },

                {"name":"XAP", "installed":true, "stack":"XAPUDP",
                        "IP":"192.168.1.255", "protocol":"UDP", "port":"3639" },

                {"name":"GUI_Reload",       "installed":true, "stack":"GUI_Reload",
                        "IP":"192.168.1.255", "protocol":"UDP", "port":"8001",
                        "url":"192.168.1.207/HomePage/CommandFusion/iPads/House.gui" },

                {"name":"Homeseer_PAL",     "installed":true, "stack":"Homeseer_PAL",
                        "IP":"192.168.1.204", "protocol":"TCP", "port":"8009" },

                {"name":"ISY99i",           "installed":true, "stack":"ISY_Subscribe",
                        "IP":"192.168.1.229", "protocol":"TCP", "port":"80",
                        "URL":"/desc", "Lighting_Object":"HS", "ISY_Userid":"admin",
                                                                "ISY_Password":"admin" },

                {"name":"Autelis_Command",  "installed":true, "stack":"Autelis_Command",
                        "IP":"192.168.1.230", "protocol":"TCP", "port":"6000", "Object":"PL"},

                {"name":"Autelis_Broadcast","installed":true, "stack":"Autelis_Broadcast",
                        "IP":"192.168.1.255", "protocol":"UDP", "port":"7890", "Object":"PL"},

                {"name":"GC1", "device":"SerialConverter",  "manufacturer":"GlobalCache",
                        "model":"GC-100-12",  "ip":"192.168.1.70", "protocol":"TCP",
                         "serial":2, "ir":6, "relays":3,
                        "serialPorts":[{"name":"SP1",   "port":4999, "connector":"1:1"},
                                       {"name":"SP2",   "port":5000, "connector":"2:1"}],
                        "relayPorts": [{"name":"RP1",    "port":4998, "connector":"3:1"},
                                       {"name":"RP2",   "port":4998, "connector":"3:2"},
                                       {"name":"RP3",   "port":4998, "connector":"3:3"}],
                        "irPorts":    [{"zone":1, "port":4998, "connector":"4:1"},
                                       {"zone":2, "port":4998, "connector":"4:2"},
                                       {"zone":3, "port":4998, "connector":"4:3"},
                                       {"zone":4, "port":4998, "connector":"5:1"},
                                       {"zone":5, "port":4998, "connector":"5:2"},
                                       {"zone":6, "port":4998, "connector":"5:3"}] },

                {"name":"GC2", "device":"SerialConverter", "manufacturer":"GlobalCache",
                        "model":"iTachIP2IR", "ip":"192.168.1.71", "protocol":"TCP",
```

```
                        "serial":0, "ir":3, "relays":0,
                        "irPorts": [{"zone":1, "port":4998, "connector":"1:1"},
                                    {"zone":2, "port":4998, "connector":"1:2"},
                                    {"zone":3, "port":4998, "connector":"1:3"}] },
              {"name":"GC3", "device":"SerialConverter", "manufacturer":"GlobalCache",
                        "model":"iTachIP2IR", "ip":"192.168.1.72", "protocol":"TCP",
                        "serial":0, "ir":3, "relays":0,
                        "irPorts": [{"zone":1, "port":4998, "connector":"1:1"},
                                    {"zone":2, "port":4998, "connector":"1:2"},
                                    {"zone":3, "port":4998, "connector":"1:3"}]
],     // end of interface configuration entries for "Systems"
"devices":[ {"name":"Master_TV",    "type":"IR",     "rmt":"MTV", "manufacturer":"Vizio",
                 "model":"E552LV", "iface":"GC1", "zone":2, "path":"Vizio_TV", "IRid":"VIZ"},
              {"name":"Master_Tivo",  "type":"IR",     "rmt":"TVO", "manufacturer":"Tivo",
                 "model":"Premiere", "iface":"GC1", "zone":2, "path":"Tivo", "IRid":"TVO" },
              {"name":"Master_Dune",  "type":"IR",     "rmt":"DUN", "manufacturer":"Dune HD",
                 "model":"Base 3.0", "iface":"GC1", "zone":2, "path":"Dune_Player", "IRid":"DUN" },
              {"name":"Master_SWT",   "type":"IR",     "rmt":"SWT", "manufacturer":"DVDO",
                  "model":"Quick6", "iface":"GC1", "zone":2, "path":"DVDO_Quick6", "IRid":"SWT" },
              {"name":"Guest_TV",     "type":"IR",     "rmt":"GTV", "manufacturer":"Insignia",
                 "model":"ZRC-101",  "iface":"GC1", "zone":4, "path":"Insignia_TV", "IRid":"ISG" },
              {"name":"Guest_STB",    "type":"IR",     "rmt":"GST", "manufacturer":"Cisco",
                 "model":"3250HD", "iface":"GC1", "zone":4, "path":"SciAtl_4200HD", "IRid":"STB" },
              {"name":"Kitchen_TV",   "type":"IR",     "rmt":"KTV", "manufacturer":"Insignia",
                 "model":"ZRC-101", "iface":"GC1", "zone":1, "path":"Insignia_TV", "IRid":"ISG" },
              {"name":"Kitchen_STB",  "type":"IR",     "rmt":"KST", "manufacturer":"Cisco",
                 "model":"3250HD", "iface":"GC1", "zone":1, "path":"SciAtl_4200HD", "IRid":"STB" },
              {"name":"Aqualink",     "type":"TCP",    "rmt":"AQU", "manufacturer":"Jandy",
                 "model":"RS_4",     "path":"JandyAqualink" }
             ], // end of interface configuration entries for Devices

 "modules":[ {"name":"ISY99_System", "id":"ISY",
                              "log":"Interface Configuration: Includes ISY99i"},
              {"name":"Homeseer",     "id":"HS",
                              "log":"Interface Configuration: Includes Homeseer" },
              {"name":"IRdevices",    "id":"IR",
                              "log":"Interface Configuration: Includes IRdevices" },
              {"name":"Autelis",      "id":"PL",
                              "log":"Interface Configuration: Includes Autelis Pool Control" },
              {"name":"PictureFrame", "id":"PF",
                              "log":"Interface Configuration: Includes Digital Picture Frame" },
              {"name":"HouseModes",   "id":"HM",
                              "log":"Interface Configuration: Includes House Modes" }
] // End of interface  Configuration for Modules
}, // End of Interface configuration
```

The Configuration entry for the Office which contains lighting devices but is typical for all lighting devices is as follows:

Office_config:

```
{ "devices":[ {"typ":"ISY", "room":"Office", "cls":"Device", "dvc":"Light",
                 "name":"Desk Light", "ID":"LT", "ISY":{"adr":"19.EF.E6"   },
                 "display":[{"type":"title",   "join":"s301", "source":"room+name",  "list":"l3" },
                            {"type":"icon",    "join":"s305",
                             "gifs":{"on":"Resources/icons/lights_On.png",
                                     "off":"Resources/icons/lights_Off.png"},  "list":"l3" }
                           ]
                 },
                {"typ":"ISY", "room":"Office", "cls":"Device", "dvc":"Light",
                         "name":"Ceiling  Lights", "ID":"LT", "ISY":{"adr":"20.93.05"    },
                 "display":[{"type":"title",   "join":"s301", "source":"room+name",  "list":"l3" },
                            {"type":"opacity", "join":{"top":"s305", "bottom":"s304"},
                                "gifs":{"top":"Resources/Icons/lights_Off.png",
                                        "bottom":"Resources/Icons/lights_On.png"},  "list":"l3" },
```

```
                        {"type":"slider",  "join":"a2142", "hidden":true,  "list":""   },
                        {"type":"text", "join":"s306",
                                    "prefix":"", "suffix":"%", "list":"l3"   }
                       ]
            }
        ]
},
```

The Main Module

This is the one of two modules coded in pure Javascript and not as a Javascript object. The second such module holds common subroutines and functions used in other projects (both CF based and non-CF but also javascript based.

After all of the modules are loaded iViewer will transfer control to the function named "CF.userMain" within this module. The main module performs the following actions in userMain:

- Disables all of the external system TCP stacks that are used. It does this using CF.systemSetProperties.
- Sets onConnection watches for the two systems that deal with servers that will be maintaining continuously active sessions as opposed to connecting /disconnecting as transmissions take place.
- Sets up watches for screen brightness, page flip events, and preloading complete
- Sets up the Pseudo room table on the settings page based upon the name of this iPad. This allows the development system to mimic any of the production iPads.

The CF.userFunction exits and the Javascript code is now inactive waiting for the onPreloadingComplete event to fire. When started, the onPreloadingComplete event does the following:

Sets the black screen as the default

- Requests the PAL (an External System) to send Slide Show information so that the iPads can be used as Digital Picture Frames.
- Enables the external system (a UDP Broadcast receiver) to watch for a GUI reload request which when seen will cause the iPad to reload the gui from the designated server.
- Calls upon the SU.setupStart function to load all the configuration information

The remainder of the Main module consists of common project utility functions.

The Common Subroutine Module

The library of support functions changes as I code things and discover better ways of doing things.

As an example of a support function in the Common Subroutine module:

Dialog

This function is used with a dialog subpage. The dialog subpage has the graphic and is a subpage placed on every page in the app so the dialog function can be used at all times. The Dialog function puts up a dialog box, a "popup" for the user with flexibility as to placement, message layout, default visibility time, and default buttons. The buttons do not do anything special at this time but should be associated with a callback function to process the buttons action. At the current time the Dialog module operates as follows: The actual call to the Dialog function is:

Dialog.show(arguments); or Dialog.Hide();

The Dialog.show function takes an object as its single argument, The parameters of the object are as follows:

text:	(Required) The text string for the popup. May include embedded carriage returns and embedded padding (spaces) at line end prior to a carriage return. The value will be split on "\r" and each array element will be one line of the final

	text. The longest line is found by using an average width for the line before it is trimmed. This allows for width padding to handle width inaccuracies. The line is trimmed and then re-joined with "\r" as the separator to make up the final text for the text box.
using:	(Optional) An array of buttons from 0 to 3 in size. Default is 0 which will be the same as 1. The array contains text for each button chosen from the set "OK", "YES", "NO", "ABORT", "CANCEL", "QUIT", "CONT".
at:	(Optional) An object with 2 properties X and Y ({X;nn, Y:nn}). The value of each property is a pixel point on the screen where the upper left corner will be placed. If the value is "Center", then the popup will be centered on the screen in that dimension. The default is "Center" for a missing property or if the entire property (at:{}) is missing.
timer:	(Optional) The amount of time the popup will remain on the screen if no button is pressed. Default is 15 seconds.

Since the most common form is a simple popup with just an OK box, we allow a call to be :
Dialog.show("A string of Text"); where all the rules for a text string as described above apply.

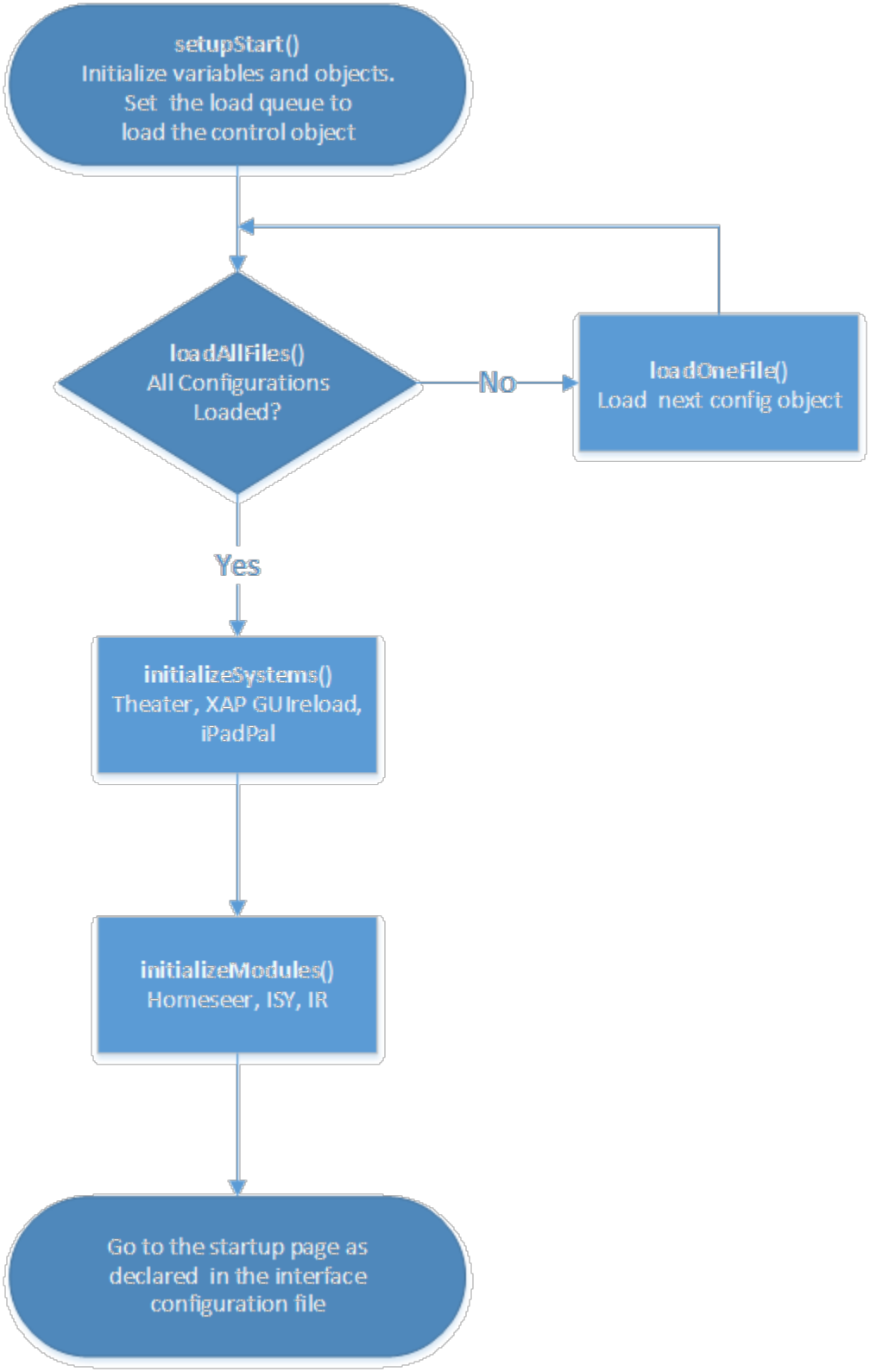
The Setup Module - Setup.js

This Module is the first of the work modules, those set up as Javascript objects. Its purpose is to read all of the configuration information from the configuration module and set up the necessary control systems and IR capabilities based upon the data read from the configuration objects. The Setup module contains the following functions:

SU.setup	Called by iViewer as soon as the module is loaded. Makes a log entry if tracing is on to show it was loaded.
SU.setupStart	Starts the actual setup process by calling SU.loadAllFiles after initializing some variables.
SU.loadAllFiles	This function reads all of the CFG objects as pointed to by the CFG.control object. When all objects have been processed, the functions to initialize systems and modules are called after which setup then exits.
SU.loadOneFile	This function will load one configuration object and process the data contained based upon the type of the control item. It is called with the type of the configuration object and the name of the configuration object as retrieved from CFG.control.
SU.initializeSystem	Initializes all systems based upon information read from the Interface configuration object. The following External systems are setup: Theater, XAP, Gui_Reload, iPadPal
SU.initializeModules	Initializes the three main modules which communicate with external systems; The Homeseer system (HS.initialize); The ISY System (ISY99i.initialize, and the IR devices (IR.initialize).

[NOTE: Initially all the configuration data was held in JSON formatted text files in a folder named Assets. Assets was a sub-folder of the main project folder. There were issues with this paradigm as guiDesigner will not export to an archive any files that are not explicitly referenced in the guiDesigner XML output, the "gui" file. In addition non referenced assets were not cached which become a problem under certain cases of reloading the project. It was never an issue for me until I started looking into loading the production iPads with an exported zip archive.]

Flow Chart of Setup Module Operation



Previous Page

Go back to page 2