Client / Server Protocol. Version 1.1

This protocol is used to define the interaction between the client and server. The client, in this case an IOS application running on an iPhone or iPad is essentially a button processor. The client receives button definitions from the server and then when a button is pressed it sends the button's identifier back to the server. The client has no knowledge of the overall process. It simply receives button definitions and sends button presses to the server. The server has the intelligence to decide what to do with the button press. Typical options include changing the color or text of the button on the client, and/or performing some other action separate from the client such as turning on or off a light or other device. The iPhone/iPad version of this protocol is available on the Apple app store. The server portion of this protocol is left as an exercise for the reader. A sample version of a Python27 server is available that shows basic functionality as a starting point.

A typical installation scenario might have the server talking to the client on one side and out the back side to a lighting controller such as Lutron, perhaps through telnet or RS232. Button presses on the client could then turn off and on lights on the premises by utilizing this back side connection to the Lutron controller.

When the client connects to the server, it sends a version string and password. No data is sent back to the client until this information is verified.

The following types of commands are sent from the server to the client. They are used to define buttons - the labels, positions, devices, and actions.

ALL STRINGS ARE TERMINATED WITH A CARRIAGE RETURN LINEFEED "\r\n".

Protocol Sent to Client

Initialization Messages To Client

In this protocol, the caret "^" is used to delimit separate fields in a message.

To initialize a button on the client, the fields have the following format. The general form of the initialization message is ^INI+ButtonName^column:row^name on button^Device^Action^^\r\n

1. 1st field - ^INI+ButtonName to add a button to the client. ButtonName must be a unique name per button.
2. 2nd field - button location - either 0, or column:row
   0 - means that the button is placed next to the last button. Note that this will place buttons across the page in one row, up until the width of the device, then increment the row and start placing buttons on the next row. It is not a good idea to intermix using "0" with explicit "column:row", as the button placement will be inconsistent.
   column:row - the first position on a view is 1:1 (one based). The first row is reserved for the "TOP" device (which actually show up in the tool bar, but nevertheless have a row value equal to 1), so the row should be 2 or greater. Note, the client is assumed to have a 16x16 grid for

buttons and should be capable of handling that many buttons on a page, though servers can limit the grid size based on the type of device that the client presents.

3. 3rd field - button text - this is what is displayed in the button
4. 4th field - device, all buttons are grouped by device, one device per page. The device TOP is reserved for placing buttons in the tool bar. The device could be thought of as analogous to a room in the house or an area of control, such as upstairs, or downstairs or kitchen etc.
5. 5th field - device dependent action, returned to server when button clicked, unless this field is set to the string IGNORE, in which case button presses are ignored. If this field is set to DISPLAY$DEVICEA, then when this button is pressed, the buttons defined for DEVICEA are displayed. DEVICEA can be any text value corresponding to the 4th field of this message (i.e. any valid device). The buttons for that device are displayed with no need for a round trip to the server. However, if there are no buttons defined for DEVICEA then the message will be sent to the server with a command of "DISPLAY$DEVICEA".
6. 6th field - BOX or ^^, if it is set to BOX then a confirmation dialog box is displayed with YES/NO buttons, to confirm that the button really has been pressed, before sending button info to server.
7. 7th field – the color of the button, or ^^ for the default color

   Examples:
   a. "^INI+Lite148^000^M.Bath - Vanity East^LIGHT^G#107^^\r\n" – this initializes a button with the identifier "Lite148" to go after the previous button, with button text of "M.Bath – Vanity East", in the device "LIGHT", with a field of G#107 that will be sent to the server when the button is pressed.
   b. ^INI+Door1^3:4^Front Door^SECURITY^IGNORE^^\r\n" In this case the button identifier is Door1, located at position column 3, row 4 (note that row 1 is for the TOP buttons so all button rows start at 2 even though they will display on row 1). The button text is "Front Door" and it is part of the SECURITY device. Pressing this button is ignored and there is no command defined to send to the server(^^).
   c. ^INI+Door1^3:4^Front Door^SECURITY^IGNORE^^Orange^\r\n - sets the background color of the button to Orange.

Operation Strings To Client

These strings are ^ delimited and sent from the server to the client under normal operation to change the text or color of a button. The general format is "^Button ID^Action^Additional data^\r\n"

1. 1st field - button identifier (ButtonName specified in the 1st field in the INI+ message)
2. 2nd field - action, one of:
   a. TEXT - append the 3rd field string to the button's text (i.e. the "Additional data" text)
   b. NEW - replace the button's label with the text in field 3
   c. ON - color the button yellow, same as ^COLOR^Yellow^
   d. OFF - color the button the default color, same as ^COLOR^default^
   e. DISPLAY - display the buttons for the Device specified by the button name. This should typically only be used when an on the fly initializing of a device is performed and then to display the newly created buttons.

   f. ^COLOR^color^ - sets the color of the button to the color specified in the third field. The color is one of colors, like Red, Blue, Green, for a complete list of colors see: http://www.flounder.com/csharp_color_table.htm or "default" to set the default background color, which is black.
3. 3rd field – dependent on 2nd field, for example if second field is "TEXT", the 3rd field is the text string to add to the button name (usually preceded by a "\n" (newline) character.

Examples:

   a. ^TEMP010^TEXT^\n75^\r\n – add the text "75" on a new line(because there is the newline character "\n" before the 75) on button TEMP010
   b. ^Lite047^ON^\r\n" – turn on button Lite047, which means change its background color to yellow.

Protocol From Client to Server

This protocol is used when a button is clicked on the client. The following information is sent to the server.

1. 1st field - button name (from INI+ string field 1)
2. 2nd field - device name (from field 4 in INI+ string)
3. 3rd field - device action (from field 5 in INI+ string)
4. 4th field - number of characters that have been added to the text string in the button using TEXT command.
   a. 0 means it's the original text
   b. n means there are n additional characters that were added to the button text on a previous message that the client received from the server – see "TEXT" message type in the section above.

Note: NEW sent from the Server to the Client resets the original text to the NEW string, which means the number in the 4th field can be meaningless. However usually the 4th field is used by the server to determine if there was extra text appended to the original string, and the server can determine easily if it wants to remove that extra text by simply setting the button text again as follows: by sending ^buttonName^TEXT^^\r\n which removes the extra characters and leaves the original text of the button as it had been originally set on the +INI message at initialization.

As an example, assume when a button represents the current outdoor temperature. When pressed, it appends to the text the low and high temp for the day. When pressed again, it removes the low and high temp for the day and just the current temperature is displayed.

example:

"^Output025^OUTPUT^O#025^0^\r\n"

Initialization String Sent from Client to Server

When the client connects, it must send a string identifying itself to the server. The string is of the form

^Version X.XX^DEVICE^PASS^password^\r\n”

Where device is the type of device the client is on such as “IPHONE”, IPAD”, “WINDOWS PHONE”, etc. The password is queried from the user (or built into the client). And the server can handle this string in any way it desires such as using the device to layout buttons in a different order.

Limits –

Button names are limited to 40 characters.
Device names are limited to 40 characters
Total number of devices supported is 130.
Overall commands strings sent and received are limited to 512 characters.