C-ChangePHP

A web-based interface for controlling Clipsal's C-Bus building management and automation family of products

Greig Sheridan

Gspam1@bimbonet.net

Sydney, Australia.

21st June 2011

1 Contents

2   Background

I was an early adopter of Clipsal's C-Bus building automation system, adding a small installation to our cosy circa-1917 semi back in 1999.

We started with a grand total of 4 or 5 devices. We took it with us when we moved into our current home, and we now have more than 30 units on the network, with a PAC, lots of dimmers, relays, auxiliary inputs, analog outputs (for 0-10V dimming control of LEDs) and even a DMX interface.

With so many individual Groups at our control, the number of outputs eventually exceeded the number of switches, so an alternate means to control everything was required – and so the web interface was born.

Clipsal does offer one or two products that permit some kind of web control, but they were either too complex for me to code, too kludgy, or too expensive for a home environment like this.

So I trawled the web, found some bits and pieces that I liked, and wrote my own. (I've used code from 2 sources in this project, and each is credited).

"C-ChangePHP" is far from perfect but it serves my purposes, and having run it successfully at home for almost a year I've decided to publish it as open-source.

There are 2 functional blocks to this distribution: the communications "engine" and the user interface. The engine is intended to be a universal interface (a driver) between the user-layer "presentation" code and the physical C-Bus network.

My focus here has been the engine, but I've provided 2 sample user interface pages to give examples of what you can do.

I'm not a professional coder, so pro's will find the code kludgy and poorly-structured. I've not used a lot of CSS in the user interface, primarily due to my inexperience with it, and this is one area where the code can definitely be enhanced to more accurately render on different sized screens or different browsers. I'll leave that to those who know what they're doing. :-)

After completing the steps in the Deployment & initial configuration section, you should have a fully-functioning browser-based control of your C-Bus installation. The "Table View" will be 100% based upon your installation (courtesy of its XML tags file), whilst the "Graphical View" will be somewhat unpredictable due to the random selection of Groups that are hard-coded in this page – but that's easily tweaked.

I hope you enjoy C-ChangePHP, build upon it and choose to publish your improvements.

Greig Sheridan
Sydney, Australia
June 2011.

3   What is C-ChangePHP?

C-ChangePHP is presented as a sample website that provides a 2-way browser-based interface to a C-Bus installation.

Files included in the distribution provide the communications engine to interface to the C-Bus network, as well as two example PHP files & supporting images that demonstrate the functionality delivered to the user.

Those familiar with PHP and HTML will be easily able to customise or completely replace the user interface with one more suited to their installation, perhaps retaining only the communications engine.

Those with limited programming experience can simply customise the provided examples for fully-functional control of their own C-Bus installation.

The two working pages provided in this distribution are a "graphical view" and a "table view". Click on each of the images that follow to be taken to a fully working demo of C-ChangePHP.

3.1 Graphical view

The "graphical view" presents to the user a line-art drawing of the building with a simple HTML table normally layered invisibly over the top. I've deliberately left the room outlines coloured in this distribution so you can see how the hot-spots are defined on the page. Clicking on any of the 'rooms' pre-selects the room, then you click on the action required in the menu on the right: on, off, or ramp.

One of two lamp icons is overlayed on each room to indicate when the light/Group is on. The lamp appears as fully yellow when the channel is at 255 (maximum), or partially yellow if the channel is less than 255 and greater than 0.

There are a couple of utility buttons at the bottom. In my real-world application I've coded some of these as "read-only" and they show things such as when the building thinks it's dark, or other such events. The coloured ball is red for off, green for on, or orange for dimmed (anywhere from 1 to 254).

In this distribution, the Graphical View has a button to toggle between itself and the Table View (with both coded into the same page). The Table View described below (visibly identical) is presented in this distribution as a stand-alone file, so there's no button to switch to the Graphical View.

3.1 Graphical view

3.2 Table View

The Table View simply reads all of the available Groups from C-Bus, then looks up the C-Gate Tags file and presents each Group to you in the table. As with the graphical version, you simply choose one or more Groups, then select the action.

Network	App	App Name	Group Address	Label	Value	Selected
254	202	Trigger Control	69	VCR PAUSE button	-	☐
254	202	Trigger Control	70	VCR FFWD button	-	☐
254	202	Trigger Control	71	VCR REW button	-	☐
254	202	Trigger Control	72	VCR STOP button	-	☐
254	202	Trigger Control	73	VCR PLAY button	-	☐
254	202	Trigger Control	74	VCR POWER button	-	☐
254	202	Trigger Control	78	CHANNEL 1 button	-	☐
254	202	Trigger Control	79	CHANNEL 5 button	-	☐
254	202	Trigger Control	80	CHANNEL 0 button	-	☐
254	202	Trigger Control	82	TV PWR ON/OFF button	-	☐
254	202	Trigger Control	83	TV ON button	-	☐
254	202	Trigger Control	84	TV OFF button	-	☐
254	202	Trigger Control	85	Mailbox	-	☐
254	202	Trigger Control	86	Smoke Alarm Signal	-	☐
254	202	Trigger Control	87	VOLUME UP button	-	☐
254	202	Trigger Control	88	VOLUME DN button	-	☐
254	202	Trigger Control	89	CHANNEL UP button	-	☐

redraw

on

off

Ramp to

60% ▼

or

specify 0-255

over

4 s ▼

ramp

4 Structure

The code described here is written in PHP, served from a standard web-server. I use WAMP running on a virtual 32-bit Windows7 machine under Hyper-V. When you send commands to, or query, the C-Bus, it opens a Telnet session to the C-Gate application (which must be running on a machine somewhere), squirts the appropriate commands and feeds the response back to the user.

The basic network architecture looks something like this:
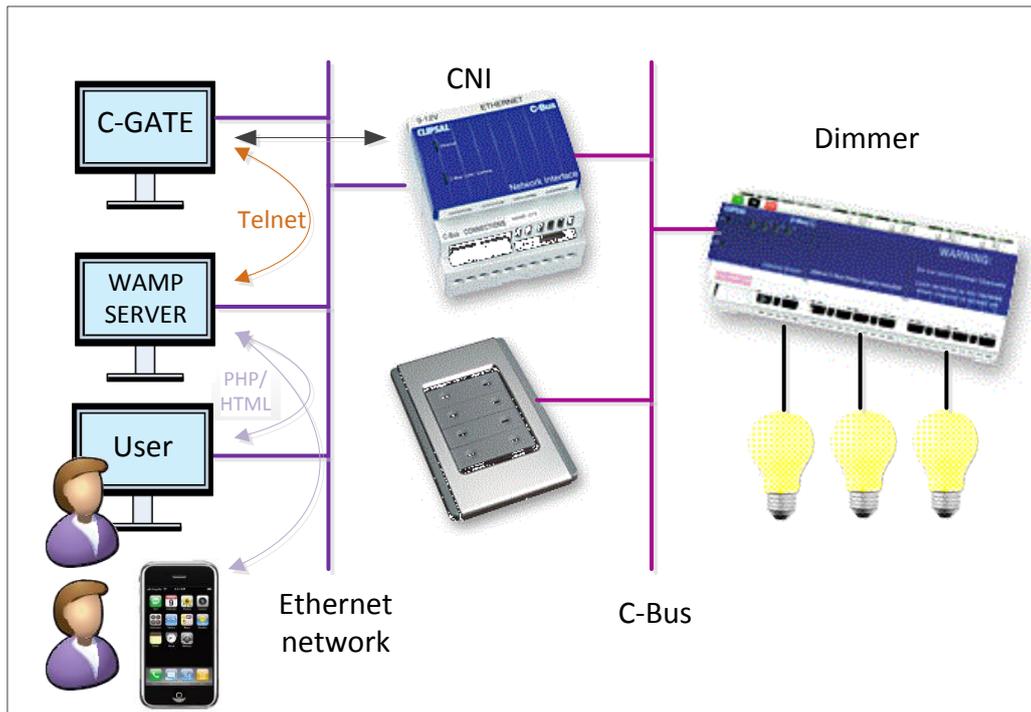
When the user types a URL into their browser (or refreshes an existing page), the page load process commences. Before the user sees any response, C-ChangePHP will open a Telnet session to C-Gate and read the status of the bus. It will then build and push a web-page to the user that shows the status of the C-Bus network at that moment.

When the user makes a selection and presses an action button (On, Off or Ramp), the relevant commands are sent individually to the bus, and then the page load process is repeated with the updated group status being read.

> Note that one weakness of this structure is that a Group in the process of Ramping to a new value will report its value at the instant the status is read and not the end value it is ramping towards.

Clicking the "redraw" button will re-initiate the page load process, updating the values of all Groups without making any change.

At the end of each page is an HTML redirect, which is actually the mechanism by which the page reloads the new values from the bus:

```
// Now redirect back to itself:
$self = $_SERVER['SCRIPT_NAME'];
header("Location: $self?ramp=" . $ramp . "&level=" . $level);
```

This code also ensures that the page doesn't accidentally re-send a previous change to the Bus if the browser page is refreshed ("F5") or by using the browser's Forward or Back arrows.

Due to the static nature of PHP code, a fresh connection is made to the Bus with each status query or change transmitted, then released. The Bus initialisation process is sped by hard-coding the Project name into the "C-GateConfig.txt" file.

5 Pre-requisites

5.1 Server-Side

1   A working C-Bus installation is a must! :-)

2   It must also have some means of programming it, so this will be one of the following:
- USB PC Interface 5500PCU
- RS-232 PC Interface 5500PC
- Network Interface 5500CN
- Pascal Automation Controller 5500PACA
- Legacy first-generation PC interface 5100PC

(Not that this project has currently only been tested with a 5500CN / CNI as the interface).

3   A PC on the network (which can be the same one hosting the website) must have C-Gate installed and running permanently. (C-Gate is normally installed with the C-Bus Toolkit).

4   If WAMP and C-ChangePHP are installed on a different machine to C-Gate, the C-Gate ini file must be edited to permit connections from the WAMP machine.

5   A PC on the network to host the WAMP web-server. This need not be the same as the C-Gate machine, but as both need to be running it might make sense to host them together.

5.2 Client-Side

The Graphical View requires Java scripting be enabled for the rooms and buttons to be selected.

If the user has Java scripting disabled in their browser the page will not work, although will present an error message to indicate this requirement is not met. Simply add an exception for this page/site.

6   Deployment & initial configuration

6.1   C-Gate

Install C-Gate and/or C-Bus Toolkit and follow the default prompts. As you're presumably adding C-ChangePHP to an existing C-Bus installation, this step has probably already been completed.

6.1.1   Add remote machine permission

If WAMP and C-Gate are on different machines, C-Gate needs an entry added to its "access.txt" file, usually located in the C:\Clipsal\C-Gate2\config folder. The syntax of the entry required is:

```
remote <WAMP PC's IP address> Program
```

If C-Gate is already running you'll need to exit and re-launch it to read this new value.

6.1.2   Hard-code the Project name

The Project name needs to be hard-coded into C-Gate's INI file. This is done to reduce some of the initialisation overhead of each communications session. As a bonus, you'll find Toolkit opens faster and automatically starts the network as a result.

In C:\Clipsal\C-Gate2\config\, open "C-GateConfig.txt" in a text editor. Search for the following two entries and change them to reflect the name of your project. (Mine is "19P" as shown below). Out of the box in a default C-Gate installation there is no text after the equals sign.

```
#### project.default:
# Name of the default project
# Default Value:
# Scope:     global
# Effective on: restart
###
project.default=19P

#### project.start:
# Space-separated list of projects to start on server startup
# Default Value:
# Scope:     global
# Effective on: restart
###
project.start=19P
```

If C-Gate is already running you'll need to exit and re-launch it to read these changes.

- End of this section -

6.2 WAMP

Download [WAMP](#) and run the install file. At the time of writing this was called "WampServer2.0i.exe". Click Next through the installer and accept the GNU licence.

This document assumes you have accepted the default installation folder of c:\wamp.

On the "Select Additional Tasks" dialog, customise the options as you see fit. I leave them off to reduce desktop clutter.

When the "Please choose your default browser" dialog opens, just click Open, as prompted, or navigate to the location of your preferred browser.
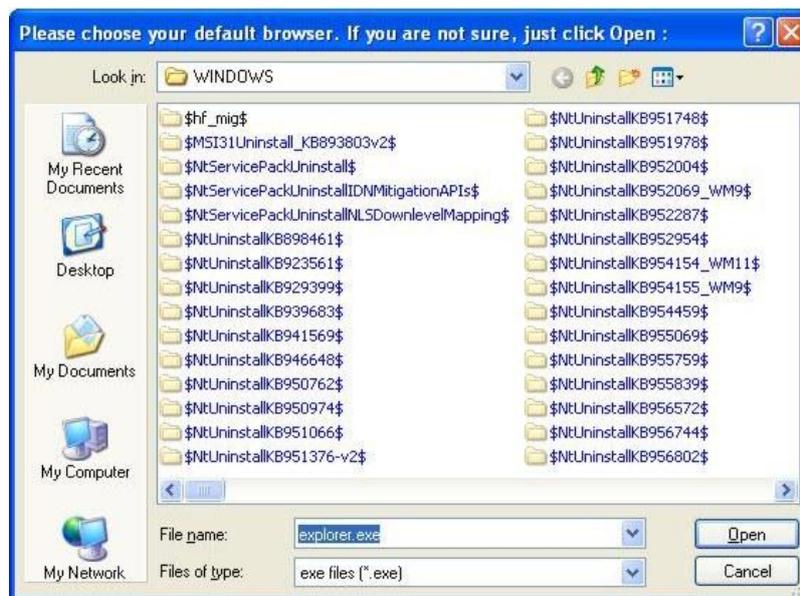
When the "PHP mail parameters" dialog opens, just click Next.
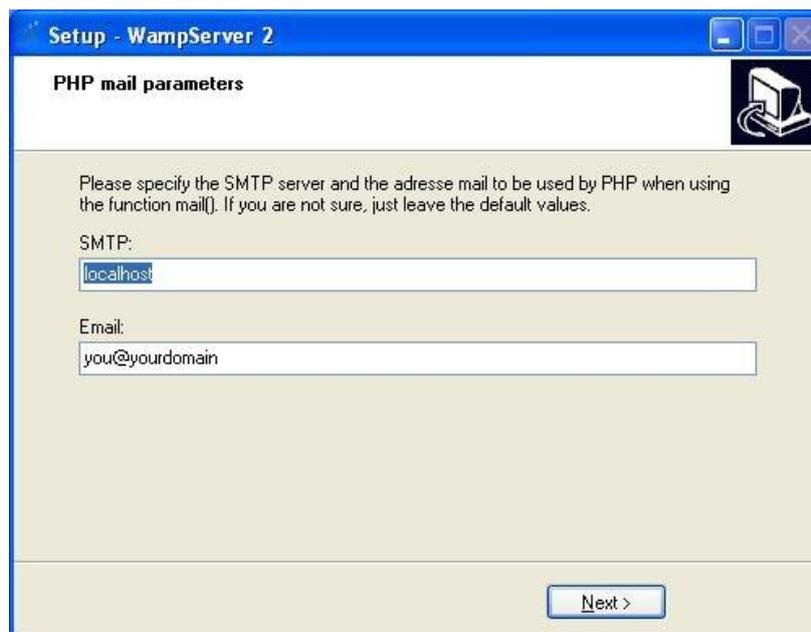
When the installation has completed, leave "Launch WampServer 2 now" selected and click Finish.

The WAMP icon should appear in the machine's tray. It's the left-most icon below:

If you hover over it, the pop-up will tell you WAMP is offline. Left-click on the icon and then select "Put Online" at the very bottom:

If you now hover it should say WAMP is online:
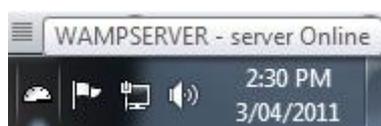
Now test that WAMP has access to port 80. Left-click on the WAMP icon in the tray, then select Apache / Service and Test Port 80:
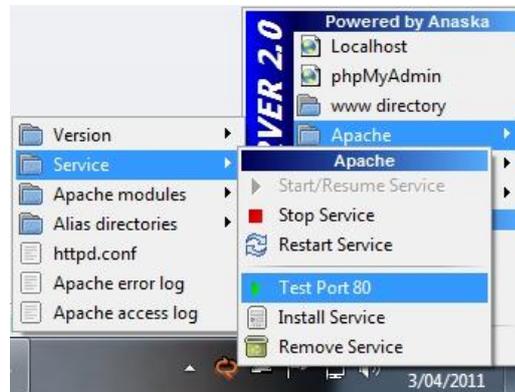
You should see this screen:

If the screen above instead says "might be Skype", refer to the fix for this in the debugging section.

There are no other changes or tweaks that need to be made to the standard WAMP installation.

- End of this section -

6.3 C-ChangePHP

Copy all of the C-ChangePHP files into the C:\wamp\www directory. You should end up with a structure that looks like this:
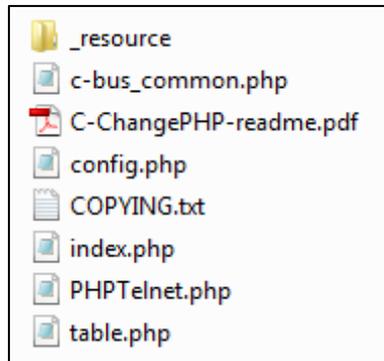
The "wamp\www" directory is the "root" directory for the files being served by the machine. To get to these files, simply open your browser and type the machine's name and then the name of the index or table file. If the above doesn't work, try the machine's IP address. If you're on a domain, you might need to use the FQDN of the machine instead.

- wampmachinename/index.php
- 192.168.1.1/index.php
- wampmachinename.yourdomain.com/index.php

The distribution of C-ChangePHP ships with demo mode enabled. This will greatly improve the chances of the default pages delivering a result "out of the box".

Open **config.php** with a text editor and make the following changes:

- Update the value for "`$demo = 1;`" if you're ready to talk to your network. Change it to a 0 to operate through C-Gate, or otherwise, leave it as a 1. Don't forget the ";" at the end of the line.

- Update the value for `$cgate = "";`
  If C-Gate is running on a different PC, place its name or IP address inside the quotation marks. Otherwise, leave the value empty and the code will assume C-Gate is running on the localhost.

- Add the name of your C-Bus "Project" - the top level node in Toolbox. If it's left blank we'll assume the Project Name is the same as the filename of the tags file (which it usually is).
  ```
  // e.g. $project = "19P";
  $project = "";
  ```

- Add the number of the C-Bus Network. This number is used in some code pages to pad out an ambiguous request to the full "<project>/<network>/<application>/<group>" command. (The default value for networks is 254 - most installations won't need to change this).

```
// e.g. $network = "254";
$network = "254";
```

- Update the value for "$tagsfile =". Revise the value here to be the path and file of your C-Gate tags file. Note the double back-slashes in the folder structure and don't forget the ";" at the end of the line. The tables won't draw if this is not set.

```
//e.g. $tagsfile = "C:\\Clipsal\\C-Gate2\\tag\\19P.xml";
$tagsfile = "C:\\Clipsal\\C-Gate2\\tag\\19P.xml";
```

- Update the value for "$comfortfile =". If a "Comfort" Alarm Panel is attached to C-bus it can be used to populate the Zone Names until Comfort supports the 'Zone Name Query' command. If you have no Comfort, just leave this entry as an empty string (""). Note the double back-slashes in the folder structure and don't forget the ";" at the end of the line.

```
//e.g. $comfortfile = "C:\\Comfort\\ComfortFile.cclx";
$comfortfile = "";
```

- End of this section -

7 Debugging – common faults and their fixes

7.1 "Internet Explorer cannot display the webpage"

WAMP isn't running or Port 80 (or the web-server itself) is not accessible. Possible causes are:

- Check you have the right IP address or domain name for the web-server.

- Does the url have the correct path to the PHP file?

- Is there any firewall, router or other device that is blocking your access to the web-server? (Try pinging it).

- The WAMP service isn't started on the web-server. Check in the PC's system tray to check WAMP's status is good. (The left-most icon is WAMP and should be white with the needle at 3 o'clock as per the below. Red or yellow visible in the icon, and the needle at any other position isn't good):
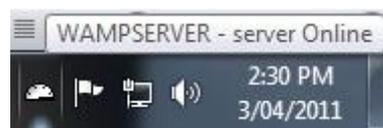
  Start WAMP if it's not running by clicking Start / All Programs / WampServer / start WampServer.

- Installing Skype will break WAMP. This can be confirmed by running a test of Port 80 from WAMP. Left-click on the WAMP icon in the tray, then select Apache / Service and Test Port 80:
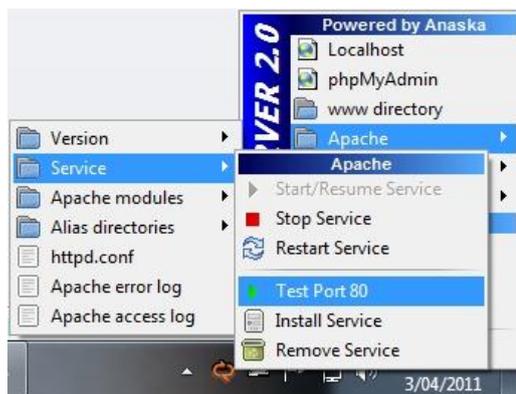
```
c:\wamp\bin\php\php5.3.0\php.exe

Your port 80 is actually used by :

Information not available (might be Skype).

Press Enter to exit...
```

To fix this, launch Skype, then select Tools / Options / Advanced / Connection and de-select "Use port 80…" as per the screen-capture below. (Accurate as at April 2011 and Skype version 4.2.0.187).

```
Notifications          Use port 80 and 443 as alternatives for incoming connections

Calls                  Automatic proxy detection          ▼

IM & SMS               Host                        Port

Advanced               Enable proxy authentication
                       Username                    Password

Advanced settings      ☑ Enable uPnP

Connection
```
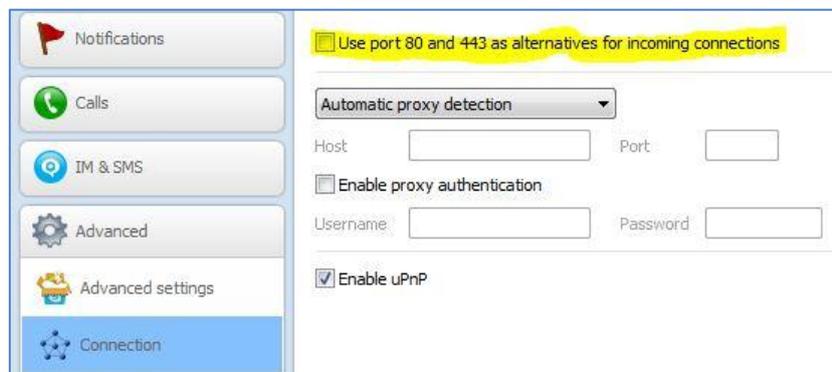
Exit and re-launch Skype for this change to take effect. Make sure you close the program completely before re-launching it – make sure the green Skype "tick" icon isn't still present in the tray – otherwise you're only re-opening the user interface. Repeat the "Test Port 80" process to confirm it now reports success:

```
c:\wamp\bin\php\php5.3.0\php.exe

Your port 80 is actually used by :

Server: Apache/2.2.11 (Win32) PHP/5.3.0

Press Enter to exit...
```

7.2 "[C-Change PHP] Connect failed: Unable to open network connection"

C-ChangePHP was not able to contact C-Gate. Possible causes are:

- The C-Gate application is not running. Check and restart it if required.
- The C-ChangePHP config has not been correctly set following installation. Check config.php.

7.3 "[C-Change PHP] Connect failed: Login failed"

C-Gate rejected the connection attempt from C-ChangePHP. This happens when WAMP and C-Gate are on different machines.

C-Gate needs an entry added to its "access.txt" file, usually located in the C:\Clipsal\C-Gate2\config folder. The syntax of the entry required is:

```
remote <WAMP PC's ip address> Program
```

If C-Gate is already running you'll need to exit and re-launch it to read this new value.

7.4 "[C-ChangePHP] Command failed: C-Gate not sync'd to the network. Wait and repeat"

This one appears after C-Gate has been started but has not finished synchronising to the network. Wait a minute or two and it should be fine. Reading the bus should be OK during this process, but the value of Groups not yet synchronised will show on the table view as a "-", and on the pictorial view as "off".

If further debugging is warranted, Telnet to the C-Gate server on port 20023 and check its status with the command:

```
get //<Project Name>/<Network>
e.g. get //19P/254.
```

The response will indicate if the network is ready to accept user input or not. The network needs to be in the 'ok' state to be fully functional.

```
get //19P/254 state
300 //19P/254: state=sync
get //19P/254 state
300 //19P/254: state=ok.
```

7.5 "Failed to open tagsfile <path>\<filename>.xml"

Pretty self-explanatory.

- Is the tag file value in config.php set correctly?
- Are there double-backslashes between directories? E.g. C:\\Clipsal\\C-Gate\\etc…

7.6 "Failed to open comfortfile <path>\<filename>.xml"

Pretty self-explanatory.

- Is the comfort file value in config.php set correctly?
- Are there double-backslashes between directories? E.g. C:\\Clipsal\\C-Gate\\etc…

7.7 "No tags file and no project. Check config.php"

Have the installation-specific values for the Tags file and Project been entered into config.php?

7.8 None of the Group names shown in the Table View are familiar

It's possible that you've accidentally chosen one of the demo tag files from the \C-gate\tags\ folder, or not set the tags file in config.php. (If no tags file is provided in config.php, we'll use the Clipsal demo file home.xml instead).

7.9 Changed or updated Groups still show their old names (1)

C-ChangePHP reads the Tags XML file when the page is first loaded and saves this information into session memory. If you add new Groups (e.g. through Toolkit) or rename existing ones, these changes won't automatically be found.

To test for this, exit and re-launch your browser. If this is not preferred, there are 2 work-arounds:

- Reload the page and add "&gettags" (without the quotes) to the end of the URL. This forces a manual reload of the tag file and other configuration information.
- Hard-code a button into the page to force a manual reload of the tags file.

7.10 Changed or updated Groups still show their old names (2)

Is it possible C-ChangePHP is referencing an obsolete version of the Tags XML file? Did you copy the file across to the WAMP server and it's not referencing the active XML file? Might you (or someone else) have made any changes to the C-Bus config using a different installation of Toolkit and a different C-Gate server?

If it's correctly referencing the right Tags file, exit and re-launch your browser. If this is not preferred, there are 2 work-arounds:

- Reload the page and add "&gettags" (without the quotes) to the end of the URL. This forces a manual reload of the tag file and other configuration information.
- Hard-code a button into the page to force a manual reload of the tags file.

7.11 "Headers Already Sent" error

If your browser outputs an error message like the sample below, it's indicating a PHP config switch needs to be changed:

```
Warning: session_start() [function.session-start]: Cannot
send session cache limiter - headers already sent (output
started at /…/c-bus_common.php:407) in /… /index.php on
line 11
```

Find the PHP.INI file and change "output buffering" to ON:

```
output_buffering = On
```

7.12 Browser (in)compatibility

A reasonable amount of effort has been expended in trying to get the site to render correctly for the usual browsers. It has been tested against and presents consistently with the following browser versions:

Android Internet Browser	2.2
Internet Explorer	8
Firefox	3.6.9
Google Chrome	10.0.648.151
Google Chrome	10.0.648.204
Google Chrome	11.0.696.65
iPhone 3GS / Safari	version 4.x

If you make changes to the code (especially the CSS), you might find any or all of the following problems will materialise:

- The labels on the buttons on the bottom of the Graphical View will be off-set and perhaps be off the button area.
- The labels on the buttons on the bottom of the Graphical View cease to be clickable, whereas the area surrounding them remains active.
- If you click on the lamp icon (when a room's channel is Dim or On) the room fails to select.

8   Hooks and returns

Communication between the user-layer application and C-Bus takes place by calling certain procedures/functions, and responses are either returned directly to the calling code, or created as session variables for the calling code to reference.

The hooks (calls) and returns are described in this chapter.

If you're writing your own user-layer page (to replace the provided table.php or index.php), these are the only calls and returns you need to send data to and read data from the C-Bus and the XML and Comfort files.

8.1   Hooks

There are only three procedures that the user-layer application calls.

8.1.1   function read_tags()

"Read_tags" must be called when the user first loads the page.

It is passed no arguments, and performs the following steps:

1. Checks if the variable $project is set, and if not, sets it as an empty string.
2. Checks if the variable $network is set, and if not, sets it as the usual default value (254).
3. Checks if the variable $tagsfile is set. If not, it forces it to "home.xml", the demo tag file usually installed with C-Gate.
4. If the tags file exists, it opens it and reads all of the Networks, Applications & Group names into session memory. If an empty Application is read, it will be discarded and not written to memory.

Non-fatal errors are written to the screen and the procedure continues, whilst fatal ones are written to the screen and the page stops executing.

8.1.2   function read_bus($passed_command)

"Read_bus" queries the C-Bus network and populates the appropriate session variables in memory.

"Read_bus" accepts a vast range of potential inputs:

- If an empty string (e.g. "") is passed, func read_bus will query *ALL* networks and apps from the tags.XML file. (It will skip any that are determined to be empty)
- If only an App (e.g. "56") is specified, func read_bus will pad with <Project> and <Network> from config.php
- If Network and App (e.g. "254/56") are specified, func read_bus will pad with <Project> from config.php
- If Project, Net and App (e.g. "19P/254/56") are specified, func read_bus will not pad
- If an entire command string is specified (e.g. "get 19P/254/56/* level"), func read_bus will transmit it verbatim

Multiple commands (e.g. "56,254/202,get 19P/254/56/* level") can be passed to read_bus, each separated by a comma. The padding rules from the above bullet-points will be applied to each separate command in the string in turn.

They will be transmitted to the bus within the same Telnet session, but treated individually and subject to the formatting as outlined above.

"read_bus" is really only a formatting stage – it sends its output direct to "send".

8.1.3   function send($msg)

"send" is called to transmit a change or query to the bus.

Unlike "read_bus", "send" must be passed a correctly-formed, self-contained command to transmit. Examples of correctly formatted commands to send are:

- "TRIGGER EVENT //19P/254/202/34 255"
- "ENABLE SET //19P/254/203/34 255"
- "RAMP //19P/254/56/34 255 20"

Multiple commands can be concatenated together, separated by a comma, and will be transmitted to the bus independently of each other from within the same Telnet session. It's acknowledged that long sequences of multiple commands are likely to impede the efficiency of the C-Bus network, so should be avoided where possible.

"send" returns "OK" if successful. Any other response is considered fatal and in the example files, fatal messages are written to the screen and the page stops executing.

8.2   Returns (Session variables)

When "read_tags" and "send" execute, they create and/or populate various session or global variables.

Fixed values (e.g. Application numbers & names, Group names) are stored in Session variables. These are held in the browser's memory until destroyed by code command, or the session ends (by the user closing the browser window).

8.2.1 $_SESSION["tag[$network][$AppNum][$group]"]

The name of the Group in the $network and $AppNum, as read from the tagsfile.

8.2.2 $status[$network][$AppNum][$group]

"status" is not a session but a global variable. Its value is only useful at the moment the bus is queried, so there's no point storing it for later.

Status reports a value from 0-255 that is the level of the Group identified by the $network and $AppNum values.

8.2.3 $Zones[($loop-1)]

If a "Comfort" alarm system exists in your deployment and the $comfortfile value is provided, "$Zones[$Group]" contains the names assigned to the Comfort Alarm system zones. These align 1:1 with the Groups in Application 1.

9 A Tour of Index.PHP – what's going on?

There are 2 ways any web-page can be called. The first is as a result of the page being called manually by the user typing the URL or clicking on a link (with or without any extra parameters passed from the URL) and the second is as a result of a form on the page having been submitted. These use the "GET" and "POST" methods respectively.

I'm using both in the same page, so we need to determine what's going on and take the appropriate action.

At its most rudimentary, both index.php and table.php are structurally identical. Both are simply an HTML form, with the room (Group) selection being performed by ticking and clearing checkboxes on the form. This is described in more detail in this and the following chapter.

The high-level flow of the page is thus:

9.1 Initialisation

First off we initialise the page and check if the "_SESSION" variables exist. These variables include parameters read from the config.php file and also the names for all of the Groups (as read from the tags XML file, identified in config.php).

If these variables aren't present, we call read_tags() - located in c-bus_common.php - to check and populate them. (Exiting and re-launching your browser will automatically discard these values).

If parameters were passed from the URL (including "level", "ramp" and "page") we declare and format the appropriate variables before proceeding.

9.2 Was the Form Submitted?

After the initialisation is complete, the page tests to see if the form on the page was responded to by the user selecting any of the actions: Redraw, change page, On, Off or Ramp.

If the user pressed the "Table View" / "Graphical View" button (which changes its label depending upon which "page" you're on), the page will re-call itself, but instead of "POST"-ing the value it will instead request the new page as part of the URL ("&page=0" / "&page=1").

```
if(isset($_POST["page"])) {
    // Now redirect back to itself:
    $self = $_SERVER['SCRIPT_NAME'];
    if ($page == 1) {
        header("Location: $self?ramp=" . $ramp . "&level=" .
            $level . "&page=0");
    } else {
        header("Location: $self?ramp=" . $ramp . "&level=" .
            $level . "&page=1");
    }
}
```

The next line starts an "IF" statement block that decides the flow of the remainder of the code on the page. If "state" is NOT set (the form was NOT submitted), run the next section:

```
if(!isset($_POST["state"])){
```

9.2.1   If the form was NOT submitted

```
if(!isset($_POST["state"])) {
```

"state" was NOT set. The user did NOT submit the form to get to this point. This section runs when the page was loaded 'cleanly', from either a browser Refresh ("F5"), or anew.

It queries C-Bus, draws the resulting table to the page, then ends, waiting for the user to press a button.

```
read_bus("56");
```

This returns after having read from C-Bus with all of the Group values ready for the code below to format and display. "`read_bus`" accepts a vast range of potential inputs:

- If an empty string (e.g. "") is passed, func read_bus will query *ALL* networks and apps from the tags.XML file. (It will skip any that are determined to be empty)
- If only an App (e.g. "56") is specified, func read_bus will pad with <Project> and <Network> from config.php
- If Network and App (e.g. "254/56") are specified, func read_bus will pad with <Project> from config.php
- If Project, Net and App (e.g. "19P/254/56") are specified, func read_bus will not pad
- If an entire command string is specified (e.g. "get 19P/254/56/* level"), func read_bus will transmit it verbatim

Multiple commands (e.g. "56,254/202,get 19P/254/56/* level") can be passed here, each separated by a comma. The padding rules from the above bullet-points will be applied to each separate command in the string in turn.

They will be transmitted to the bus within the same Telnet session, but treated individually and subject to the formatting as outlined above.

In the example files provided we only query Application 56 (Lighting).

```
?>
<div id="container">
```

At this point we drop out of PHP back to HTML (that's the "?>") and simply draw the page on-screen. The first element to place on the screen is the division called "container". Divisions and the screen-layout are discussed in the next chapter.

We briefly jump back into and out of PHP in some of the commands. In the drop-down lists for "level" and "ramp" in the menu we pre-select the last-used values, making them 'sticky' between page-draws. If the page is drawn fresh, the initialisation code has already set a default of 60% and 4s for these values.

```
<option value="60" <?php if ($level == "60") {echo "selected";}?> >60%</option>
<option value="8" <?php if ($ramp == "8")  {echo "selected";}?> >8 s</option>
```

So if $ramp="8", the HTML for this option has "selected" added to the code sent to the browser.

Once the page has been drawn to screen the code stops, waiting for the user to select some options and press a button. When that happens, the entire process starts from the Initialisation.

9.2.2   If the form WAS submitted

If the form was submitted, (and we've already addressed the "change page" button) then the only remaining values are On, Off and Ramp – and so this load of the page is required to transmit a change to the bus.

The next code sections determine the button that was pressed, what Groups / rooms were selected, and if any of the other parameters were entered or are applicable:

- The dim percentage
- a "specify" value
- The ramp time

All of the checkboxes on the form belong to an array called "channel[]", and when checked, the value placed into the array corresponds to an Application and Group number:

```
<p><input type="checkbox" name="channel[]" value="56/33"
       id="select_33">Group33</p>
```

After the dim/specify and ramp values are determined and the action to be taken is known, we run through the array and extract all of the checkboxes that were selected:

```
$each_ch = strtok ($channel, ",");
while ($each_ch !== false) {
       $transmit = $transmit . "ramp //" . $project . "/" . $network .
            "/" . $each_ch . " " . $binary . " " . $tx_ramp . ",";
       } // Note that this code example is edited for clarity. App tests are not shown.
$each_ch = strtok (",");
```

This process (slightly edited above) loops through all of the $channels (checkboxes), and for those "not false", their value is added into the "$transmit" string.

"`transmit = $transmit .`" builds a comma-separated list of the commands to be sent. At the conclusion of this while loop, we have a command string to send to the bus:

```
if ($transmit == ""){
      // Do nothing - no command selected (probably "Redraw")
} else {
      $retval = send($transmit);
      if ((strpos($retval, 'OK') === false)) {
      echo ("<br><br>ERROR. Return value = " . $retval . "<br>");
      die;
      }
}
```

… and the page recalls itself from the top:

```
// Now redirect back to itself:
$self = $_SERVER['SCRIPT_NAME'];
header("Location: $self?ramp=" . $ramp . "&level=" . $level .
"&page=" . $page);
```

From here, the flow of the page is back to Initialisation.

10 The Visual Aspects of Index.php in More Detail

If you're not familiar with PHP or HTML, this section will provide some tips on just what the default index.php file is doing and how it can be customised to your installation.

10.1 DIV's

The page is physically drawn on the screen as a number of HTML "divisions". You'll find them in the code with the <div id= "name"> and </div> tags. The closing tags are often some distance from the opening, but I've commented each </div> so you know what it's closing.
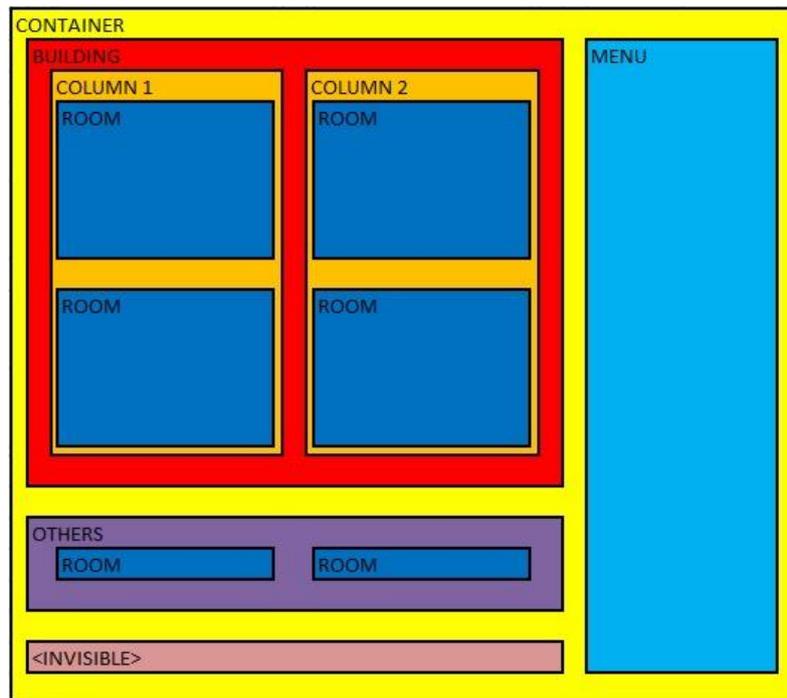
You can see from the drawing that all of the divs are nested inside others. Some (the rooms and buttons) are drawn as part of the functions "display" and "buttons":

```
function display($net, $app, $gp, $room, $width, $height){
echo ("<div id=\"" . $room . "\" class=\"photo\" >");
        <stuff happens here>
echo ("</div> <!-- " . $room . " -->");
}
```

The size and placement of each of the div's on the page is ruled by the CSS file. Here are 3 edited examples and I've highlighted some of the particularly relevant settings - the size, the "float" and the border:

```
#container {
        width: 800px;
        height: 570px;
        }
```

```
#building   {
        float: left;
        position: relative;
        width: 600px;
        border: 1px solid black;
        }

#menu {
        float: right;
        position: fixed;
        width: 100px;
        border: 2px solid gray;
        margin-left: 620px;
        text-align: center;
        }
```

For the most part, I've left borders enabled, in a range of colours to help you identify which div is where on the screen. Once you're happy with the design and layout, just turn the relevant borders off by editing the CSS:

```
border: none;
```

10.2 It's really just a giant, pictorial form!

The mess of divs became necessary to turn what is really just a giant HTML form into the end result.

Under the skin, each "room" or button (horizontally across the bottom of the screen) is actually just a check-box on a form, hidden from the user.

```
<div style="display:none">
        <p><input type="checkbox" name="channel[]" value="56/33" id=
            "select_33">Group33</p>
        <p><input type="checkbox" name="channel[]" value="56/34" id=
            "select_34">Group34</p>

<!-- Not selectable: -->
        <p><input type="checkbox" name="channel[]" value="56/20" id=
            "select_20" disabled>Group20</p>
</div> <!-- Invisible section of the menu -->
```

The form's "Submit" buttons are in the Menu div along the right-hand side.

Because the check-boxes aren't visible (`style="display:none"`) we need another means to access them, and this is the job of the JavaScript, described shortly.

As an experiment, delete the "display:none" and see what happens on-screen.

All of the following code in this chapter was required to:

- lay out the hot-spots that functionally align with the check-boxes
- "check" and "uncheck" the boxes ready for when the form is submitted
- add and remove the 'tick' to show the box had been selected or not; and
- then display an icon to show the status of each Group.

In comparison, the raw "table" code that's inbuilt into the same page delivers *exactly* the same functionality at a fraction of the effort.

Review the code in between `<div id="table_div">` and `</div> <!-- table_div -->`.

```
//echo ("This is the table view");
?>
<div id="table_div">
<table border=1>
        <see how little there really is in here. The 'complex' stuff in here is just
        getting the tag names right. >
</table>
</div> <!-- table_div -->
```

It's worth mentioning here that I *incorrectly* refer to a "Graphical Page" and a "Tabular Page" repeatedly in the code and documentation. Both screens/views are presented to the user from within this one PHP page, but *appear* to be totally different pages. The only thing that changes in the url is the "&page=n".

If you happen to have a multi-story building, this is where (and how) you can revise the code to add extra "pages" to select a different floor.

10.3 Functions "display" & "buttons"

Functions "display" and "buttons" are called when each of the DIV 'columns' is placed on the page, with "display" being called once for each room and "buttons" called for each of the buttons.

The parameters passed to the two functions vary slightly, but for the rooms (the more complex), we pass the Network, Application, Group, its name, width and height:

```
<div id="column1">
      <!-- SYNTAX display(Group address, room name, width, height) -->
      <?php display(254,56,33,"UpstairsL", 185,160)?>
      <?php display(254,56,34,"DownstairsL", 150,160)?>
</div> <!-- Column1 -->
```

The functions place the clickable hot-spots in position on the page and add the appropriate status icon.

So you can tell if a Group is on, off or dimmed I added the light-globe icon. The globe is absent if the Group is off (intensity=0/255), partially yellow if it's dimmed (1-254) and fully yellow if it's fully on (intensity=255/255). Relays and other non-dimmable Groups only report values of 0 or 255.

I decided that these 3 status indicators were sufficient for the purposes of the index page, but you can easily make this more granular by adding other tests to "`function display`" and "`function buttons`".

```
if ($status[$net][$app][$gp] < 1) {
      // It's OFF
      <display "transparent.gif">
} elseif (($status ... [$gp] > 0) && ($status ... [$gp] < 255)){
      //It's 'dimmed'
      <display "lamp_dim.gif">
      <display "transparent.gif">
      } else {
      // It must be full on
      <display "lamp_on.gif">
      <display "transparent.gif">
}
```

The only real difference between the two functions is that the buttons are all hard-coded the same size (so you don't need to pass dimension parameters to the function), and that you can make the buttons read-only, so they can't be clicked/selected.

Why are we displaying two icons in each of the cells?

"Transparent.gif" is the entire size of the division / room and it's the clickable entity. When clicked, it calls the JavaScript code (see the next section) which changes the HTML "class" and applies or removes the tick image.

The lamp_on.gif and lamp_dim.gif are smaller, and are centred on each of the cells with the following code extract:

```
style=\"left:" . ((0.5*round($width))-25) . "px; top:" .
        ((0.5*round($height))-25) . "px; width=50px height=50px
```

They're not clickable, but are placed on the page first, underneath transparent.gif.

This layering has resulted in a page that can be up to 4 layers high on-screen.

transparent.gif
selected.gif
lamp_on.gif
house.gif
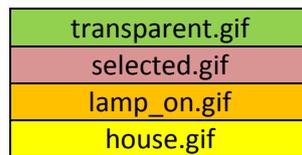
Obviously, the sequencing of this is critical. If you get it wrong, you might find that once a Group is On or Dim, you can't select the room by clicking on the lamp icon itself, and the space around the icon is the only area still 'hot'.

The layering is evident when you click on a Group that's already on. The "tick" image completely covers the lamp icon.

10.4 The JavaScript element

```
<script type="text/javascript">
// with thanks to:
// http://blog.jellytriangle.com/2009/12/24/fantastic-graphical-
checkboxes-using-css-and-javascript/
// for the funky checkbox JavaScript.

function selectApp(id,divid,what)
{
        if(document.getElementById(id).checked==true) {
                document.getElementById(id).checked = false;
                document.getElementById(divid).className = what;
        } else {
                document.getElementById(id).checked = true;
                document.getElementById(divid).className = (what +
'_checked');
        }
}
</script>
```

The JavaScript on the page exists for one purpose – to provide feedback to the user that they've selected a "check-box" on the form. It does this by overlaying the "tick" graphic on top of the form (on top of the image).

Without this code (or with JavaScript disabled), there'd be no means to know which Groups / rooms you'd selected prior to "submitting the form" with the appropriate On, Off or Ramp button.

11 References

11.1 PHP website

I've leant heavily on the [PHP website](#) throughout the design process and I commend it to you heartily if you're wanting any help with the syntax of PHP and the commands available.

11.2 The Missing Manual – CSS

"The Missing Manual – CSS" helped me get the presentation side of the site to the sorry state it's in. It's published by Pogue Press / O'Reilly, written by David Sawyer McFarland.

ISBN 978-0-596-52687-0. My copy cost me AU$75 from Dymocks.

11.3 The C-Gate documentation

It's installed automatically when you install C-Gate. "CGateManual.pdf" lives in C:\Clipsal\C-Gate2\.

12 Acknowledgements & Credits

C-ChangePHP has been a great challenge and a great accomplishment. Along the way I've encountered a few people who've helped and guided me:

Clipsal. For producing such a wonderfully flexible control system, and having the confidence to open the control interface to third parties like me (and you!).

JellyTriangle. For the [" fantastic-graphical-checkboxes-using-css-and-javascript"](#).

Antoine Roundy. For ["PHP Telnet"](#). Without this I'd be nowhere.

Leichhardt Electrical Wholesalers. The best little toyshop in all of Leichhardt.

Ingo de Jager. My amazing beta-tester. The man who inspired me to evolve C-ChangePHP from only controlling the Lighting Application, to making it the much more comprehensive control mechanism you have today. And for adding almost 2 months to the Beta process with countless bug reports and suggestions, resulting in many more late nights than I had ever intended.

Jon, Kelvin. PHP and CSS review and incredibly diplomatic critiques.

Larry, **Gus.** For their techy input.

Rocky. For putting up with my madness all these years.

13 Support

13.1 C-Bus forums

The C-Bus forums website is another great resource, full of really helpful people. There's an entire section devoted to C-Gate and Toolkit. They won't necessarily be able to help with my code, but if you're tinkering with C-Gate you'll find lots of useful info and knowledgeable contributors.

13.2 From the author

C-ChangePHP is supported on a best-endeavours basis by the author. Please e-mail gspam1@bimbonet.net with as much information as you can.

To aid debugging, please leave an original copy of the C-ChangePHP files on your PC and revert to these when-ever you're experiencing problems. This may help determine if the errors are in the communications engine or your presentation code.

14 Software Licence

C-ChangePHP is Copyright 2010, 2011 Greig Sheridan.

This file is part of C-ChangePHP.

C-ChangePHP is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

C-ChangePHP is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with C-ChangePHP. If not, see http://www.gnu.org/licenses/.

- end of document -