**Working with AHQ / Tulsa Code**

**1 - Anatomy of a Tulsa site**

Our Tulsa websites are templated but with lots of scope for customisation. This enables us to create sites that can be tailored to a client’s requirements, but at the same time benefit from a core of established robust code.

All dealers’ sites, and also instances of the ‘back end’ (called the DMS – Dealer Management System) run on a particular *build* number. (as of 29th April 19 the current build is 123). All client sites are gradually upgraded to the latest version.

The code sits in two main locations:

**PHP files**

This is where the html and php code used on all pages resides. The directory is as follows:

/var/www/3codebase/123/garage-sites/

The 118 directory contains code for a specific build ID.

Inside the /garage-sites directory there are 4 further directories:

/page This contains files with the html used to render the pages. Inside this folder there is a tulsa directory that contains files specific t the theme. In time, as more themes are added to our system, more directories will appear here.

/load This contains php code that is executed before the page is loaded

/include These are include files containing php functionality that is used in function calls. These are usual called from files in the load phase.

/element These are include files that are used as page elements. The usually contain html code and are specific to a theme. These are grouped into directories for each theme. In this directory you will often see files suffixed with a ClientID number. These are used to create portions of a page that are very bespoke to a particular client.

**CSS, JS and WP Theme Files (Tulsa Resources)**

Any files that need to be referred to in http calls to render a web page are stored in a different location. We tend to call these ‘resource files’, and in the case of the Tulsa theme they are stored in the following directory:

/var/www/tulsa-rsc/public\_html/rsc/118

This is the directory specifically for build 118.

Inside this directory there are three main subdirectories:

/wp-theme This contains the Tulsa WordPress theme

/js All the JavaScript code

/css This contains all the CSS files used in a tulsa site

The /css folder also contains subdirectories for specific clients. This is where customisation occurs. Even if a client’s site does not deviate far from the core Tulsa design, their client CSS files will at least contain their colour definitions.

As a general rule, the Tulsa CSS files contain definitions that apply to all Tulsa sites and deals with sizes, positioning, fonts, etc. The client CSS definitions tend to deal with colour, or anything custom.

**2 – Using the GIT**

We use a version control system to keep everyone’s copy of the code up to date. This allows several people to work on the same project at the same time. It even allows us to work on the same file – although we need to take care and used best practice at all times in order to avoid creating conflicts.

The basic principle of the GIT is that it contains a central repository of the code in the cloud. Individual users can then create their own local repository by cloning it from the cloud.

Once they have a local repository they can make changes locally and then ‘push’ those changes to the local repository, so that other users can subsequently ‘pull’ those changes in to their local repositories

**Repositories Used**

* php
* tulsa rsc
* js-css (used in DMS only)
* Client site folders (These are numbered by client ID and contain the actual files used in the public\_html directories [with the exception of the wp build], and contain mostly images)

**Committing**

Committing is a process that stores incremental changes into your local repository. If you commit your changes frequently this makes it easy to see what changes have occurred and also makes it easier to figure out If changes subsequently need to be undone.

**Pulling**

Pulling down changes from the server allows you to keep your local repository up to date. So that you don’t risk overwriting your recent changes, you should commit them before you pull. Its also important to pull before you push!

**Updating servers**

When you push changes to the central repository, they are then automatically copied on to a staging server (typically stg01). This process is known as ‘deployment’.

Changes to the live server(s) are deployed manually by BG.

Best practice:

* Push your changes frequently
* Always use this process:

*Commit > Pull > Push*

* If it reports a conflict – Let Barry know immediately!