The uPHP Compiler

uPHP scripts are written by humans as text files. But to speed things up the Wattmon system actually executes a compiled binary version, which loads and runs much faster after the first time it is created.

Mainline PHP is not a compiled language. However, uPHP is designed to run with limited memory constraints and uses a compiler to reduce requirements and speed loading and execution of scripts. Compilation is performed automatically by the firmware “in the background” as needed: usually the programmer does not need to be concerned about this step after writing a script.

If you have spent some time looking at the files on the Wattmon you may have noticed that there are “.cgi” script files and sometimes a second version with the same name but extension “.cgc” (this is the 'cached' or 'compiled' version). You might also have noticed that there are other uPHP script files that are not of type .cgi, such as .inc files (these other types are not compiled, but they can be included in a compiled script).

This page provides some practical information on when and how the uPHP Compiler works, of interest to those who are customizing the Wattmon, writing their own scripts or modifying the scripts provided with the Wattmon OS.

When Does the Firmware Compile a Script?

Every time a uPHP script with extension “.cgi” is run the firmware system automatically checks first to see if there is an available compiled version of the script (it looks in the same directory as the .cgi file for a .cgc file of the same name):

- If the .cgc file already exists, the system loads and runs it (the .cgi text file is ignored).
- If the .cgc file does not exist, the compiler checks the .cgi for any fatal coding errors. If no compiling errors are found it writes the compiled .cgc file to disk, then runs it.

The compiler skips/ignores comments, checks for fatal coding errors and creates a much more compact binary version that will run without all of this additional overhead. As this usually needs to be done only once when the .cgi script is first executed, subsequent runs of the same script are much, much faster. Once compiled, the script text file does not need to be loaded, which results in reduced memory requirements.

Compiling Errors

Fatal coding errors will cause the compiler to quit without creating a .cgc file (syntax errors, unknown functions, included files not found, etc.). A System Log entry is created showing the error and line number where compilation failed. The compiler error is also displayed if the script is run from a browser or the telnet console.

Of course, if there are compiling errors the script cannot be executed until the coding error(s) are fixed.

Compilation of Included Scripts

A uPHP script can, of course, be written and given a file name with any type or extension, but it will not be directly executable. (The uPHP compiler only directly evaluates files of type ".cgi".)

However, there is a practical reason for writing uPHP scripts in other file types as they can be "included" in a .cgi script at the time of compilation. (The uPHP compiler indirectly evaluates these other file types when included in the main .cgi file.)

If any type of file is included in a .cgi script, the compiler evaluates the code contained in it and inserts it into the .cgc compiled version of *the "calling" script* (if there are no fatal errors). This is why you will not see a .cgc version of an included file (that is not run directly), even if it is of type ".cgi".

For more information see function include.

Conditional Compilation for Included Scripts

If an included file does not yet exist the compiler will fail with a fatal error unless you use a conditional test for the include. For example, use function file_exists to test for existence:

```
if (file_exists("include.inc")) include("include.inc");
```

When the included file is later created recompilation will occur and a new .cgc is created when *the "calling" script* needs to use the included file.

Here is an example from the Wattmon OS file `/scripts/runonce.cgi`:

```
include("/app/package.inc");
$packages=getPackageList();
$keys=array_keys($packages);
for ($vi=0;$vi<sizeof($packages);$vi++) {
    if ($packages[$keys[$vi]]==2) {  // active, enabled
        log("Loading module "."/package/".$keys[$vi].".cgi");
        if (file_exists("/package/".$keys[$vi].".cgi")) {
            include("/package/".$keys[$vi].".cgi");
            if (function_exists($keys[$vi]."_init")) {
                call_user_func($keys[$vi]."_init");
            }
        }
    }
}
```

In this example `/app/package.inc` usually always exists. However, the `/package/` files may vary depending upon the configuration or OS version. They usually all exist, but the compiler will not compile any of them into `/scripts/runonce.cgc` until they have been made *`active, enabled`*. When the configuration is changed to activate or enable one of these packages, the firmware will recompile `/scripts/runonce.cgi` (the next time the system is rebooted) to include the code in the active and enabled package.

The File Manager and the Editor

The tools included with the Wattmon OS automatically delete the compiled .cgc file when a .cgi is uploaded or modified, which will trigger compilation the next time the .cgi is run.

Notes

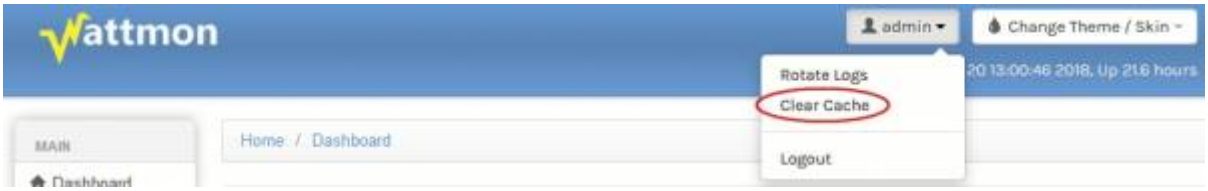
To force compilation of a .cgi script simply delete the corresponding .cgc file and the system will compile it again the next time it is run.

There are a few common situations where recompilation needs to be forced by deleting the .cgc file:

- If another computer is used to make changes to a .cgi script directly on the SD card (without using the Wattmon OS tools such as File Manager or the Editor).
- If changes are made to any included files in the .cgi script (see function include).

In these situations, the system will run an existing .cgc without examining the .cgi text file or included file(s) for changes, so be sure to delete the corresponding .cgc file at the same time. (Note from the author of this page: I learned this “the hard way” because I didn't have this documentation at the time, which led to the creation of this page.)

In most cases, an easy way to delete compiled versions is to use the dropdown menu option “Clear Cache” (in the header of most displayed pages on the Wattmon):



“Clear Cache” deletes all .cgc files in the “/” (root), “/app”, “/scripts”, “/package” and “/shell” folders (if the script you are working with is located somewhere else you will need to find and delete the .cgc file manually).

Also See

include() - Include a file within the current script at the current location

From:
http://wattmon.com/dokuwiki/ - **Wattmon Documentation Wiki**

Permanent link:
http://wattmon.com/dokuwiki/uphp/uphp_compiler

Last update: **2021/09/13 05:57**