



EXEOUTPUT for



# USER GUIDE

G.D.G. Software



# **ExeOutput for PHP User Guide**

**Copyright © G.D.G. Software 2011-2012**

All rights reserved. No parts of this work may be reproduced in any form or by any means - graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems - without the written permission of the publisher.

Products that are referred to in this document may be either trademarks and/or registered trademarks of the respective owners. The publisher and the author make no claim to these trademarks.

While every precaution has been taken in the preparation of this document, the publisher and the author assume no responsibility for errors or omissions, or for damages resulting from the use of information contained in this document or from the use of programs and source code that may accompany it. In no event shall the publisher and the author be liable for any loss of profit or any other commercial damage caused or alleged to have been caused directly or indirectly by this document.

Created in April 2011 - Updated in March 2012

# Table of contents

<b>1</b>	<b>DESCRIPTION .....</b>	<b>5</b>
<b>2</b>	<b>STARTING A NEW PROJECT .....</b>	<b>5</b>
<b>3</b>	<b>DEVELOPING AND TESTING YOUR PROJECT.....</b>	<b>7</b>
3.1	PROJECT FILES .....	7
3.2	COMPILING AND TESTING YOUR PROJECT.....	8
3.3	PHP EXTENSIONS.....	9
3.4	PHP.INI .....	10
<b>4</b>	<b>CUSTOMIZING THE APPLICATION.....</b>	<b>10</b>
4.1	BROWSER ENGINE .....	10
4.2	VISUAL APPEARANCE .....	11
4.3	EXE ICON AND VERSION INFORMATION.....	12
4.4	SPLASH SCREEN.....	12
4.5	LOCALIZATION .....	13
<b>5</b>	<b>NAVIGATION IN COMPILED WEBPAGES .....</b>	<b>13</b>
5.1	OPENING COMPILED FILES IN EXTERNAL APPLICATIONS .....	14
<b>6</b>	<b>ACCESSING FILES WITH PHP .....</b>	<b>14</b>
6.1	ACCESSING COMPILED FILES FROM PHP.....	14
6.2	PHP'S INCLUDE, INCLUDE_ONCE, REQUIRE, REQUIRE_ONCE .....	15
6.3	FORCE EXEOUTPUT FOR PHP TO MAKE A COMPILED FILE AVAILABLE TO PHP .....	15
6.4	GET THE PATH TO THE FOLDER CONTAINING YOUR EXE FILE .....	17
<b>7</b>	<b>MODIFYING AND SAVING FILES WITH PHP .....</b>	<b>17</b>
<b>8</b>	<b>WORKING WITH DATABASES .....</b>	<b>18</b>
8.1	USING SQLITE.....	18

---

8.2 USING MYSQL.....	20
<b>9 SESSIONS, COOKIES AND GLOBAL VARIABLES.....</b>	<b>21</b>
9.1 SESSION AND COOKIES.....	21
9.2 GLOBAL VARIABLES.....	22
<b>10 CUSTOMIZING YOUR APPLICATION WITH HESCRIP T.....</b>	<b>23</b>
10.1 WHAT IS HESCRIP T .....	23
10.2 ABOUT THE USERMAIN SCRIPT .....	24
10.3 ABOUT THE MACROS SCRIPT.....	25
10.4 CALLING HESCRIP T FROM PHP, JAVASCRIP T AND WITH LINKS.....	25
10.4.1 From PHP.....	25
10.4.2 From hyperlinks .....	26
10.4.3 From JavaScript .....	26
<b>11 SECURITY OF YOUR APPLICATIONS .....</b>	<b>27</b>
11.1 SECURITY CONCERNS.....	27
11.2 PHP SCRIPT ENCODING AND NO CACHING OPTIONS .....	28
11.2.1 Encoding with bcompiler.....	29
11.2.2 Do not cache PHP files into memory.....	29
11.3 SECURING YOUR SENSITIVE STRINGS .....	29
11.4 GLOBAL PASSWORD, EXPIRATION DATE.....	31
<b>12 LINKS TO SUPPORT AND DOCUMENTATION .....</b>	<b>32</b>

## 1 Description

---

ExeOutput for PHP is a PHP and website compiler for Windows. It creates stand-alone Windows applications that run and display PHP websites natively.

An application you create with ExeOutput for PHP acts like a normal web browser: the end user browses your PHP pages as if he was using his web browser, except that no internet connection, PHP installation or remote web server is required.

PHP scripts are interpreted by the built-in PHP runtime and results are immediately displayed to the end user. Other companion files such as HTML, images, JavaScript, XML, CSS... are of course handled too.

You can create feature-rich applications for the desktop: they combine several technologies such as PHP, JavaScript, the Trident (MSHTML) or Chromium (WebKit) rendering engines, HEScript and Windows API together.

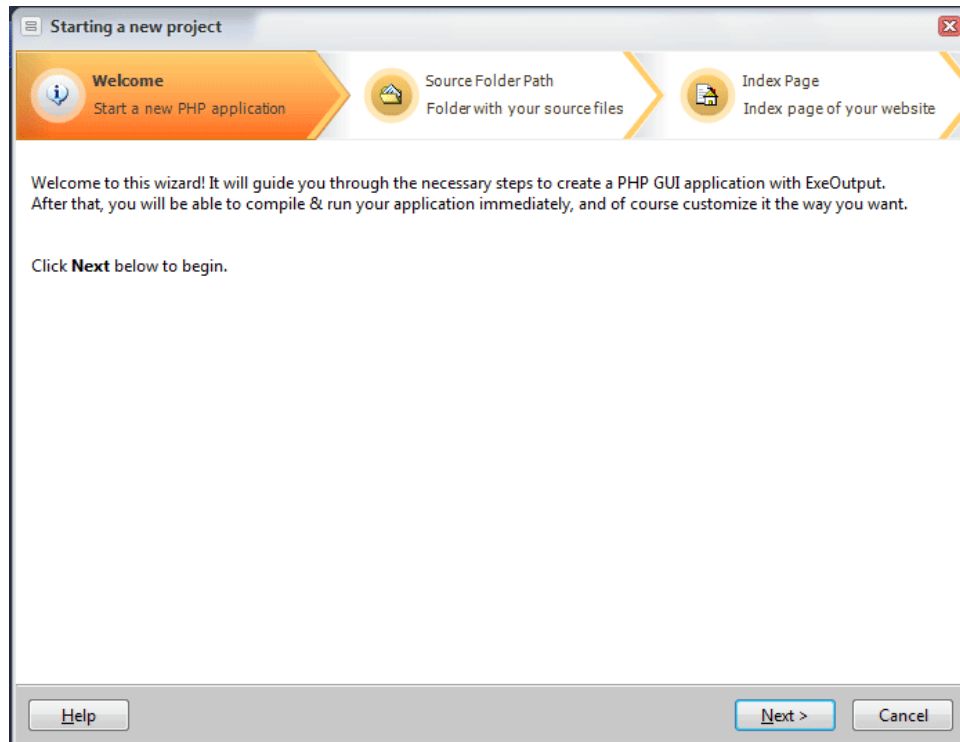
## 2 Starting a new project

---

- Prepare the website you want to compile: all source files must be in a folder that will become the root folder of your compiled website. The directory structure of your website is stored in the compiled application. File paths will be stored relative to this source folder so links between HTML pages will work exactly as if you were navigating through a website on a server.



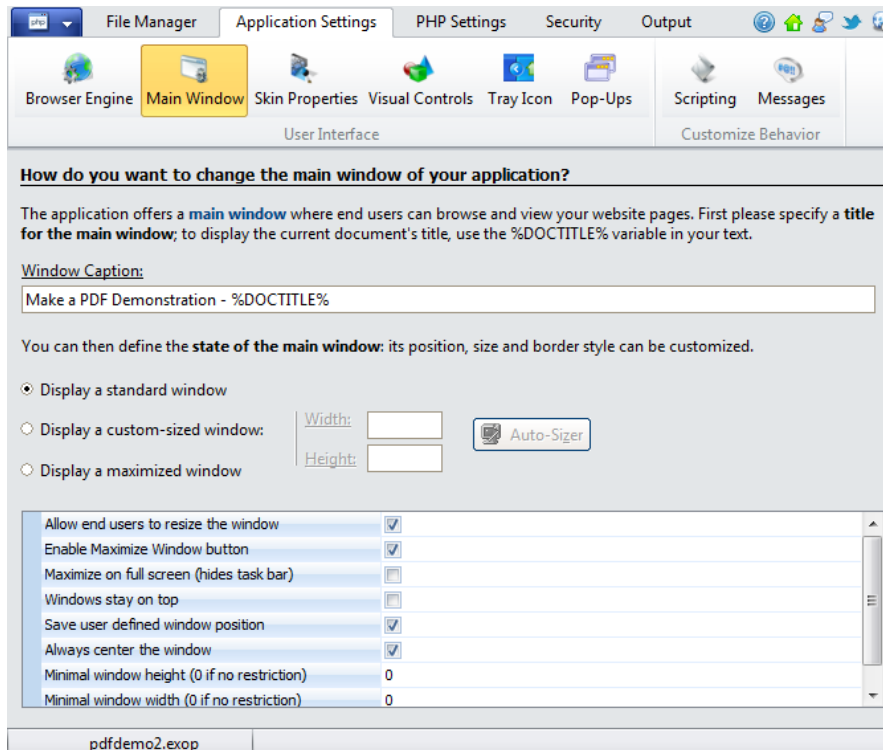
- In ExeOutput for PHP, select “Start a new project”. The following window appears; it is called the New Application wizard and will assist you in the creation of your new project.



- Click Next and now choose the source folder that contains all of your website files. All the files inside this folder and its subdirectories are automatically considered as part of the application. You can, of course, add/remove files later if you wish.
- The next step consists in choosing the index page of your application. The index page is the first page displayed when the application is run. It is considered the Home page of your website. Select which PHP or HTML page you want to set as the index page. A PHP/HTML file inside a subfolder is accepted.
- Final step: you have to specify the path to the application .exe file that will be output by ExeOutput for PHP. It must be a full path including directory, filename and extension. You can select it by clicking the Browse button. It is recommended that you do not put your output .exe file in the source folder, but rather in a different one. After that, give a title to your application. The title will appear on all dialog and message boxes displayed to end users as well as in the Windows task bar. IMPORTANT: if your title contains a quote (") character, be sure to replace it by two quotes "" in order to avoid script compilation errors.
- Finally click Finish to create your project.

ExeOutput for PHP will then prepare a blank project, configure all settings by default and automatically add files from the source folder. It generally takes a few seconds depending on the number of files you have in your source folder.

When it is ready, the "Main Window" page is displayed: you are ready to edit your project and you can also immediately compile it if you wish.





## 3 Developing and testing your project

### 3.1 Project Files


ExeOutput for PHP stores all of its settings (the project) in a file called project file (file extension: .exop). You can use the Load/Save buttons in the main window's bar to open or save project files at any time. We recommend you to regularly save your project.

### 3.2 Compiling and testing your project

To compile your project, you have several ways:

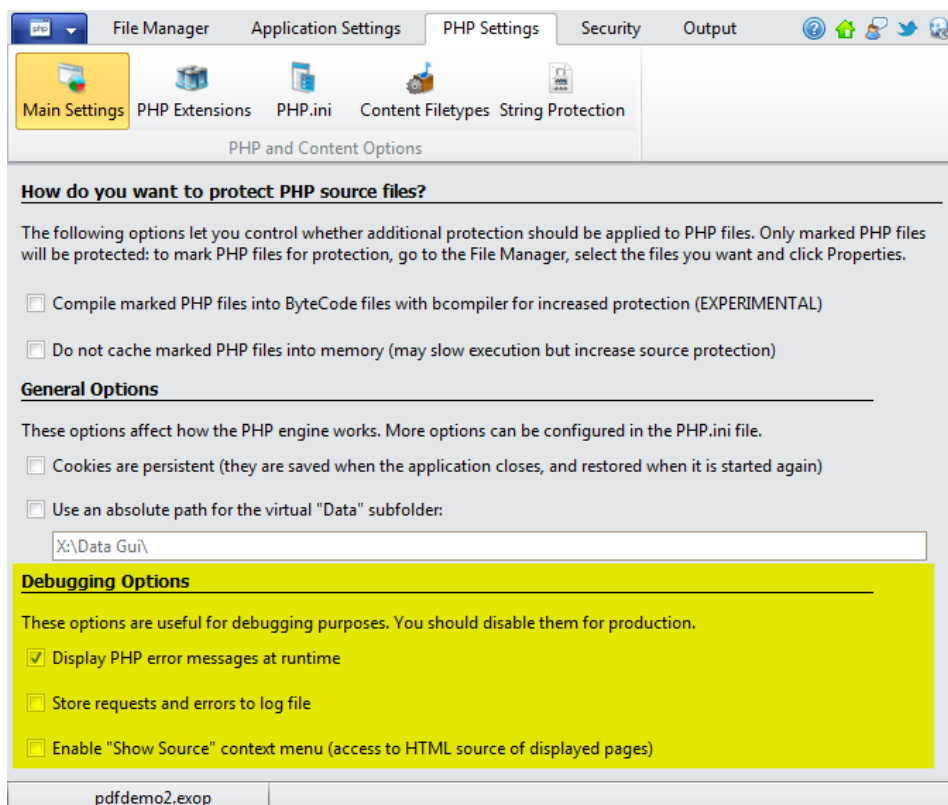
- Click Build  in the toolbar.
- Use the Build commands from the Application menu: 
- Press F5 for compilation, F9 for compilation and run, and F10 for full build.

Please keep in mind that all source files must exist when you compile the application.

Hint: to launch the application, you can press F9 or . If the project was modified, ExeOutput for PHP will compile it and launch it immediately.

Your project may not immediately work in a compiled state: you may have to make modifications.

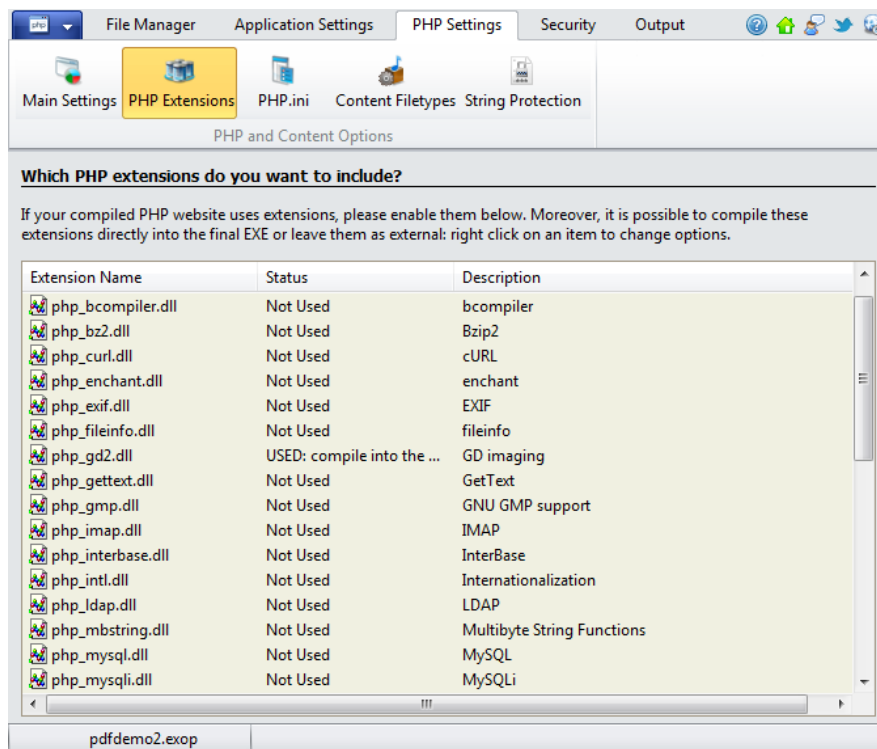
**In order to detect PHP errors**, during the development of your project, you have several debugging options that should make your debugging job easier. For instance, error and warning messages are displayed by default when you start a project. You can disable them in the PHP Settings => Main Settings page:



### 3.3 PHP Extensions

If your PHP website uses PHP extensions, you must configure them: PHP comes with several extensions that you can use in your php applications. ExeOutput for PHP is able to compile these extensions in the EXE directly or you may prefer to leave them as external files. In that case, they must be deployed with your EXE file.

You must tell ExeOutput for PHP which extensions are required by your application: this can be done in the "PHP Settings -> PHP Extensions" page. No need to edit the PHP.INI yourself: ExeOutput for PHP manages the "Extensions" section itself.



To indicate a required extension, right click on it in the list and choose between the two options:

- The extension can be compiled into the EXE file. No need to deploy it separately.
- It can be left outside the EXE: in that case, it must be placed in the "Data\ext" subfolder. For instance, if your EXE is located at "c:\my documents\", the subfolder would be "c:\my documents\Data\ext". ExeOutput for PHP will copy the extensions to this folder automatically when compiling the application. When you deploy your application, this folder structure must be kept.

Note: if you are using cURL: the cURL php extension requires two additional libraries named libeay32.dll and ssleay32.dll. If you want to compile them too in your application, right click on php\_curl.dll and you have the option to include or not the two DLLs required by cURL.

### 3.4 PHP.ini

The PHP's initialization file, generally called php.ini, is responsible for configuring many of the aspects of PHP's behavior. ExeOutput for PHP lets you configure the php.ini that will be used for your application.

See the PHP docs for more specific information: <http://php.net/configuration.file>

Note: the default php.ini used is available in the "PHPRuntime" subfolder.

Some sections and values are automatically managed by ExeOutput for PHP and should not be modified. The php.ini is automatically compiled in the EXE. No need to deploy it separately.

## 4 Customizing the application

---

### 4.1 Browser engine

A browser engine takes marked up content (such as HTML, XML, image files, etc.) and formatting information (such as CSS, XSL, etc.) and displays the formatted content on the screen (source: Wikipedia).

ExeOutput for PHP lets you choose between two browser rendering engines: Chromium (WebKit + V8 JavaScript engines) or Trident (used by Internet Explorer and internally by Windows).

You can switch between the two engines without creating a different project.

Both engines allow you to use HTML 5 and CSS 3 in your applications.

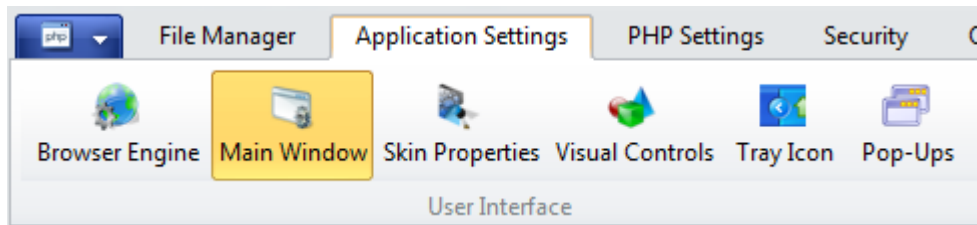
The Chromium engine is still experimental so we recommend you to stay with the Trident engine if you have trouble.

For the differences between the two engines, please refer to the documentation.

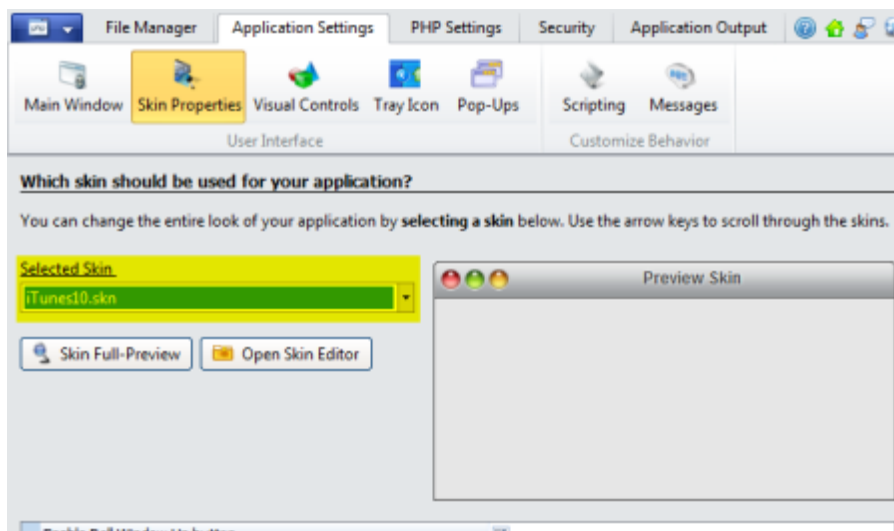
## 4.2 Visual Appearance

ExeOutput for PHP lets you completely customize the visual items of your application.

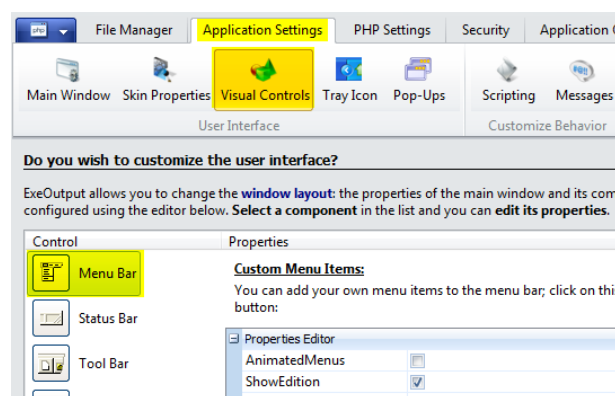
The Application Settings ribbon gives you access to different window and control features like the toolbar, options related to the browser engine, pop-ups...



One of the most interesting features is skin support. The entire look and feel of the application (its windows and controls) may be changed with a simple list thanks to skins.



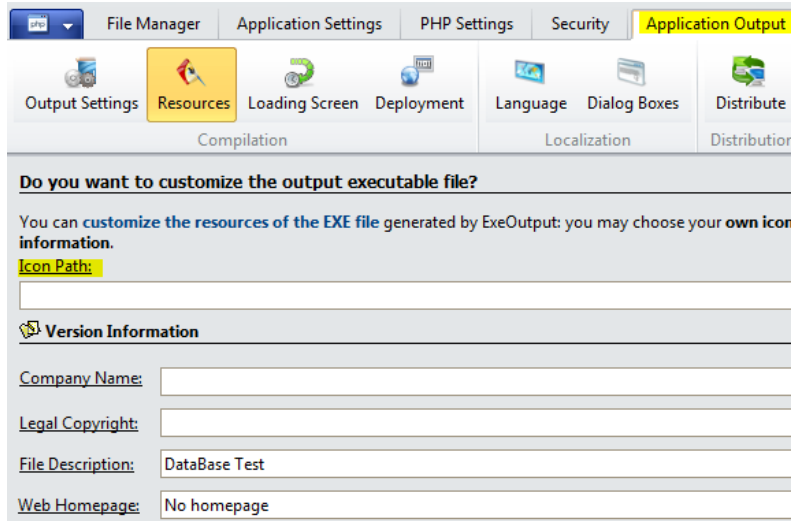
Menu bar, toolbar, status bar can be changed the way you want:



You may insert your own menu items and buttons in the toolbar.

### 4.3 EXE icon and version information

A program EXE file can hold an icon and version information. ExeOutput for PHP lets you choose an icon that will represent your application and change the associated version information.

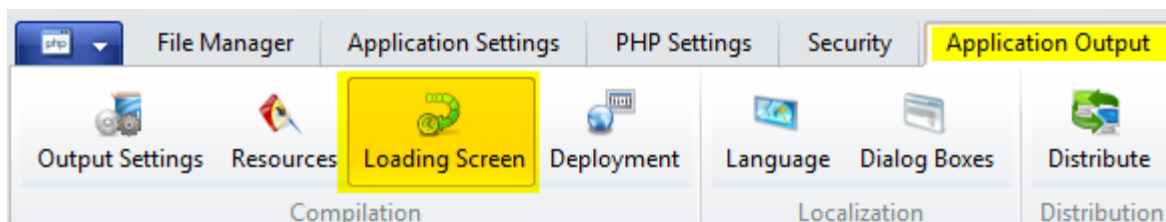


You must specify the full path to a real icon file (Windows ICO format). ExeOutput for PHP comes with some icon files you can find in the Resources subfolder.

Note: if you want to create your ICO file from an existing image, use our GConvert software program. You can download GConvert from our website at <http://www.gdgsoft.com/gconvert>

### 4.4 Splash screen

Your application can show a splash screen while it is loading.



Create your splash screen with your favorite image editor, save it as a JPEG image and specify the full path to the JPG file in the appropriate field under “Loading Screen”.

## 4.5 Localization

If you want to localize your application in other languages, ExeOutput for PHP lets you change any text or message used by the application.

## 5 Navigation in compiled webpages

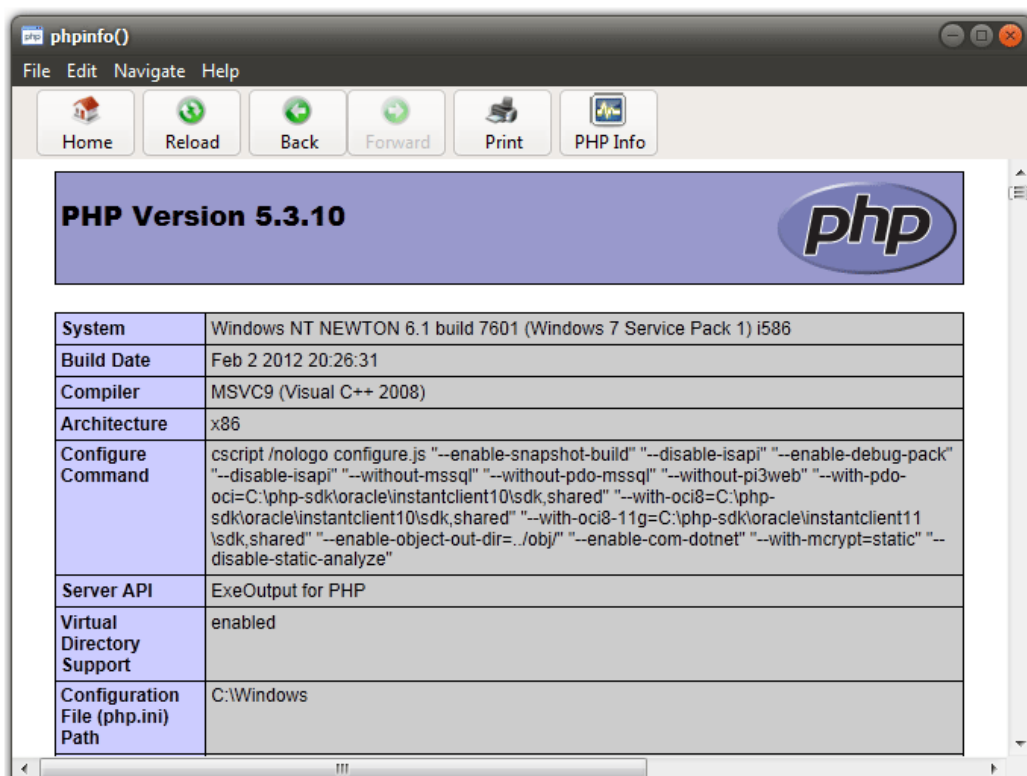
Your application works as if it was server software, serving webpages and related files through the HTTP protocol or a built-in protocol named GHE similar to HTTP.

ExeOutput for PHP defines its own pluggable protocols (`http://heserver/` and `ghe://heserver/`) to display pages in the web browser. Any compiled webpage in your application can be accessed from the application's browser using the URL:

`http://heserver/[virtual path]` or `ghe://heserver/[virtual path]`

When your application starts, the browser automatically navigates to your index page.

The following screenshot shows you a simple application that displays the result from the `phpinfo()` function:



Security note: protocols used by ExeOutput for PHP work only in your application. If you try to open URLs beginning with `http://heserver/` or `ghe://heserver/` from another web browser such as Internet Explorer, they won't display any document.

## 5.1 Opening compiled files in external applications

If the browser can't display some files such as Adobe PDF or Microsoft Office DOCX, you can view them in their default associated application. For instance, PDF files could be displayed in the default PDF viewer, like Adobe Reader.

You have several ways to open compiled files.

For Trident applications, hyperlinks are the easiest way: ExeOutput for PHP provides you with some pre-defined targets that you can use for hyperlinks in your pages. Here you have to use the `_heopenit` target.

HTML code: `<a target="_heopenit" href="[destination URL]">Your link</a>`

`_heopenit` tells ExeOutput for PHP to temporarily extract the file specified by [destination URL] (it must be the virtual path starting from the root without the protocol: `MyPath/MyFile.ext`) to the hard disk and runs the associated program to open the file. The file is deleted when the application is closed.

Example: `<a target="_heopenit" href="res/demo1.pdf">Open a PDF file</a></p>`

## 6 Accessing files with PHP

With PHP file functions, your application can access all files located on the end user's computer. You may want to know how you can access files compiled in your application from your PHP scripts.

### 6.1 Accessing compiled files from PHP

ExeOutput for PHP makes the PHP runtime believe that PHP scripts and other resource files are on the hard disk in a virtual subfolder named "Data" from the folder where the EXE is located.

The `$_SERVER['DOCUMENT_ROOT']` server variable contains the full path to the virtual "Data" folder used. It includes the trailing back slash.

For instance,

```
<?php print($_SERVER['DOCUMENT_ROOT']); ?>
```

Would return:

```
C:\my documents\phpsamples\generaldemo\Output\Data\
```

If the EXE file is located in

```
C:\my documents\phpsamples\generaldemo\Output\
```

If you try to locate the Data subfolder yourself, you are unable to find any, neither folder nor PHP scripts: they are actually virtual files.

Consequently, with PHP, you can open files like this:

```
<?php  
$handle = fopen($_SERVER['DOCUMENT_ROOT']."resource.txt", "r");  
?>
```

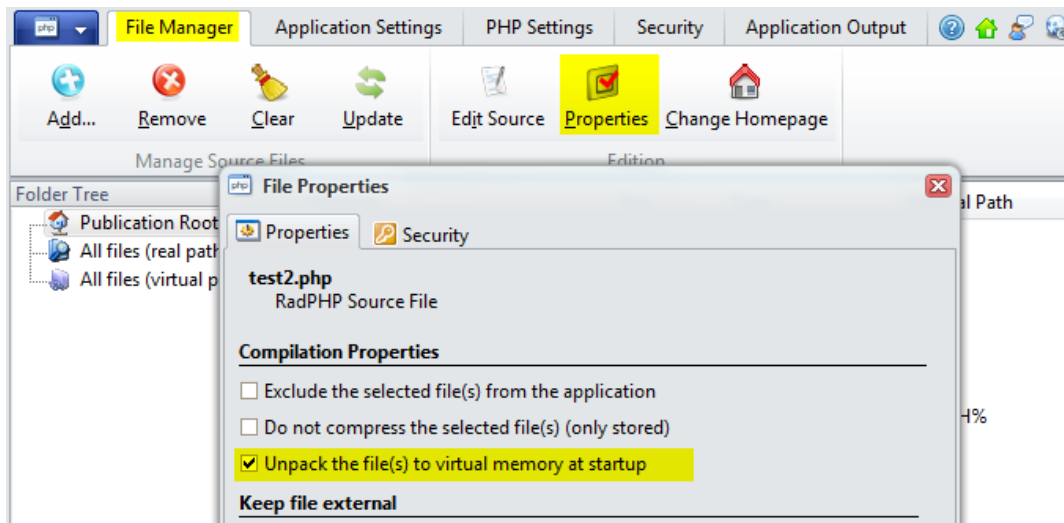
Warning: php's fopen cannot download and open files from URLs beginning with http://heserver/ or ghe://heserver/ as no real web server is used. Use real paths as explained above.

## 6.2 PHP's include, include\_once, require, require\_once

ExeOutput for PHP supports: include, require, include\_once and require\_once with simple or more complex paths. If PHP is unable to locate a file that you reference with one of the include commands, please contact us to report the problem. In the meanwhile, you can see the next paragraph for a workaround.

## 6.3 Force ExeOutput for PHP to make a compiled file available to PHP

Generally, ExeOutput for PHP will intercept all file requests made by the PHP runtime. If this is not the case, you may need to mark files in the File Manager: select your files, click Properties and turn on this option **“Unpack the file(s) to virtual memory at startup”**.



Another programmatic option: use the ExeOutput's built-in PHP function named `exeoutput_unpackvirtualfile`.

```
string exeoutput_unpackvirtualfile ( string $sourcepath , string $optionaldestpath )
```

where:

`$sourcepath` is the virtual source path to the compiled file you want to unpack to memory.

`$optionaldestpath` is the absolute path to the destination virtual file. If you leave it blank, the path from `$_SERVER['DOCUMENT_ROOT']` + the virtual path is used.

The function returns the absolute path to the virtual file (in UTF-8 format). This path can be used with php functions like `fopen`, `file_exists`, `file_get_contents`.

Example: the following script makes the `demo1.pdf` available and displays its full (virtual) path and size:

```
<?php $filename = exeoutput_unpackvirtualfile ('res\demo1.pdf', '');
echo $filename . ': ' . filesize($filename) . ' bytes';
?>
```

## 6.4 Get the path to the folder containing your EXE file

Use the following PHP code:

```
<?php
$pathtofolder = exo_getglobalvariable('HEPublicationPath', '');
?>
```

where:

`$pathtofolder` contains the full path to the folder that contains your application's .exe file (no filename). It will always include the path trailing backslash (e.g. C:\MyPath\).

## 7 Modifying and saving files with PHP

PHP provides you with several ways to modify and save files.

In order to correctly save files, you must ensure that the user has permissions to write into files in the location you want.

For instance, you can make portable applications with ExeOutput for PHP. A portable application is generally started from a USB stick and the folder that contains the EXE has write permissions. Your application will be able to save files in that folder.

On the contrary, if your application is installed in the Program Files directory, it is generally not allowed to save files in the application's folder.

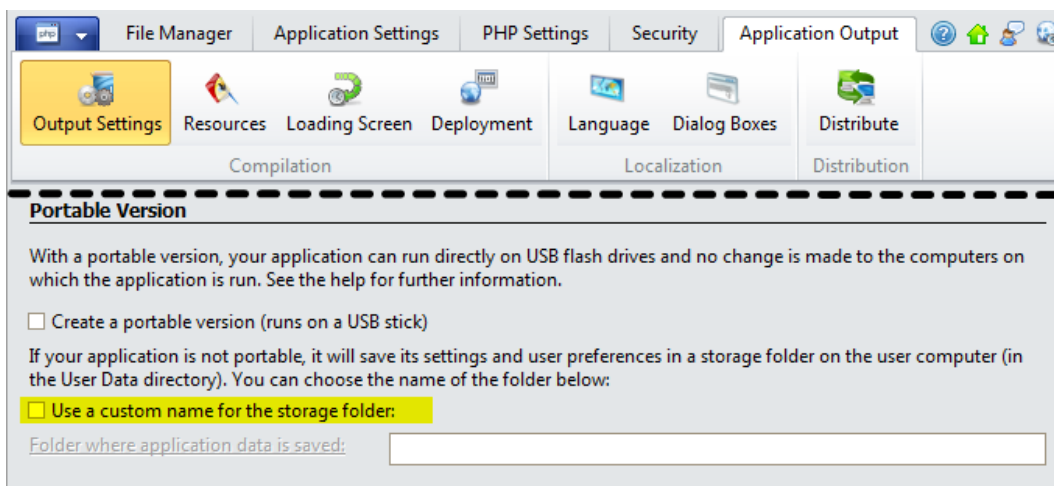
Thus, ExeOutput for PHP provides you with a storage folder dedicated to your application. The absolute path to this folder can be retrieved with the following php command:

```
<?php
$storagelocation = exo_getglobalvariable('HEPubStorageLocation', '');
echo $storagelocation;
?>
```

This PHP code would display

```
C:\Users\MyUser\AppData\Local\ExeOutput\UserApplication\{FA3B284B-6494-4033-B51D-7F2949AFB4FB}\
```

You may customize the name of the storage folder in ExeOutput going to the "Application Output -> Output Settings" page:



This folder is always available: you can use it to store files of your application.

## 8 Working with Databases

Applications compiled with ExeOutput for PHP can work with databases but it may require some additional work as ExeOutput for PHP does not provide a real server.

Since ExeOutput for PHP creates applications for the desktop, we recommend you to work with a local database solution (a database related to the application and with one user). The database should be stored in regular files locally.

### 8.1 Using SQLite

SQLite is an embedded database library that implements a large subset of the SQL 92 standard. Its claim to fame is the combination of both the database engine and the interface within a single library, as well as the ability to store all the data in a single file.

The SQLite library is not included by default. It's a PHP extension that may be enabled using the "PHP Settings -> PHP Extensions" page in ExeOutput. Enable these two extensions:

- php\_pdo\_sqlite.dll
- php\_sqlite3.dll if you want to work with SQLite 3

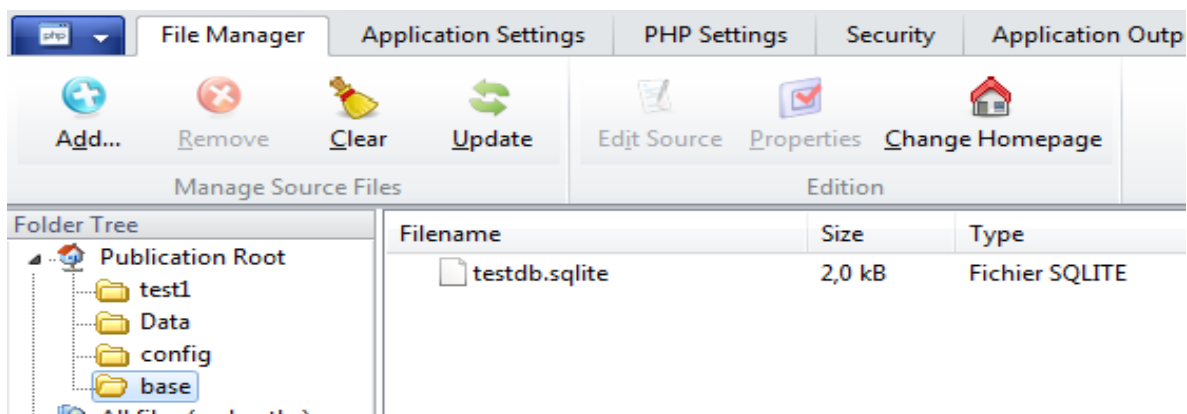
or

- php\_pdo\_sqlite.dll
- php\_sqlite.dll if you want to work with previous SQLite versions.

**Warning:** do not add php\_sqlite3.dll and php\_sqlite.dll in the same project.

If you want to use a read-only database, you can compile its storage files inside the EXE and access files as explained above.

For instance, we have a read-only SQLite 3 database file whose relative path is "base\testdb.sqlite".



To open the database file with SQLite and PDO, we would use this code:

```
<?php
$storagelocation = $_SERVER['DOCUMENT_ROOT'];
$dbname = $storagelocation.'base\\testdb.sqlite';
$db = new PDO('sqlite:'. $dbname);
$result = $db->query('SELECT * FROM Dogs');
// close the database connection
$db = NULL;
...
?>
```

Otherwise, if your database is going to be updated by the user and you want to keep changes on the disk, it must be kept outside the EXE in the storage folder (see the previous paragraph about the storage folder).

For instance, we want to create a SQLite 3 database file and update it. The following PHP code can be used:

```
<?php
// Gets the location of the database we want to create:
$storagelocation = exo_getglobalvariable('HEPubStorageLocation', '');
$dbname = $storagelocation.'testdogs.sqlite';

//open the database
$db = new PDO('sqlite:'.$dbname);

//create the database
$db->exec("CREATE TABLE Dogs (Id INTEGER PRIMARY KEY, Breed TEXT, Name TEXT, Age
INTEGER)");

//insert some data...
$db->exec("INSERT INTO Dogs (Breed, Name, Age) VALUES ('Labrador', 'Tank', 2);".
"INSERT INTO Dogs (Breed, Name, Age) VALUES ('Husky', 'Glacier', 7); " .
"INSERT INTO Dogs (Breed, Name, Age) VALUES ('Golden-Doodle', 'Ellie', 4);");

// close the database connection
$db = NULL;

?>
```

See the working sample in the General Demonstration. We also have a SQLite editor available as a sample at <http://www.exeoutput.com/samples.php>

## 8.2 Using MySQL

Your application may access databases on a remote server, such as SQL servers through intranets or the Internet. Your application would work like a client. In that case, engines like MySQL, PostgreSQL can be used.

MySQL requires a server where data should be stored. ExeOutput for PHP does not provide a real server on which MySQL could be set up.

Moreover, MySQL is licensed under the GPL: for commercial use (like for our compiler), you have to pay a license fee: see <http://www.mysql.com/about/legal/licensing/oem/>.

If you want to work with MySQL despite it being not suitable for local applications, you need a remote server or a WAMP package such as Server2Go.

We provide you with a sample that aims to show you how to use MySQL with your applications compiled with ExeOutput for PHP. Thanks to the use of Server2Go, a portable WAMP package, your applications made with ExeOutput for PHP can work with MySQL databases locally and can be easily deployed, without requiring complex installations.

You can get the sample at <http://www.exeoutput.com/mysql.php>

The MySQL php extension is not included by default. You must enable it using the "PHP Settings -> PHP Extensions" page in ExeOutput. Enable these two extensions:

- php\_pdo\_mysql.dll
- php\_mysql

Warning: avoid using database login info directly in PHP code compiled with ExeOutput for PHP. See below about security.

## 9 Sessions, cookies and global variables

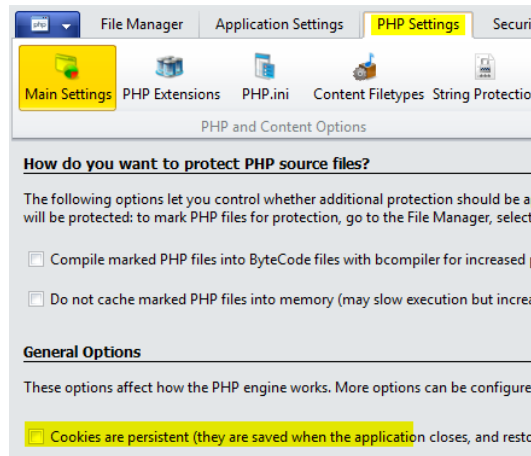
---

### 9.1 Session and cookies

Using sessions and cookies is possible with ExeOutput for PHP. We have detailed working samples and PHP code available in the General Demonstration.

ExeOutput for PHP handles server side cookies set by PHP functions. However, ExeOutput for PHP ignores cookie expiration dates: a cookie will remain valid until it is removed.

About persistent cookies: cookies remain valid until the application is closed. ExeOutput for PHP lets you save and restore cookies for the next time the application is run. If you wish to do so, enable the "Cookies are persistent" option as shown below:



To remove a cookie, set its value to an empty string.

## 9.2 Global Variables

ExeOutput for PHP allows you to work with global variables that can be both used by PHP, HEScript and JavaScript scripts in order to share data and store values in the application. They are better and more secure than cookies. If you were using cookies in your JavaScript or PHP code, we recommend you to switch to global variables.

You can manage global variables using:

- the SetGlobalVar and GetGlobalVar functions in HEScript.
- the SetGlobalVariable and GetGlobalVariable methods in JavaScript.
- the `exo_setglobalvariable` and `exo_getglobalvariable` functions in PHP.

ExeOutput for PHP defines several global variables at startup. You may use these variables in your PHP, JavaScript or HEScript code.

All pre-defined global variables are listed in the documentation. Some useful ones are:

<b>HEPublicationFile</b>	The full path to the application's .exe file (including filename)
<b>HEPublicationPath</b>	The full path to the folder that contains the application's .exe file (no filename). It will always include the path trailing backslash (e.g. C:\MyPath\).
<b>HEPubStorageLocation</b>	The path where the application stores its data.
<b>HEPubTempPath</b>	The path to a temporary location where the application stores its external files temporarily. This will change each time.

PHP code example:

```
<?php
// Gets the location of the database we want to create:
$storagelocation = exo_getglobalvariable('HEPubStorageLocation', '');
// Full path to the database file
$dbname = $storagelocation.'testdogs.sqlite';
?>
```

## 10 Customizing your application with HEScript

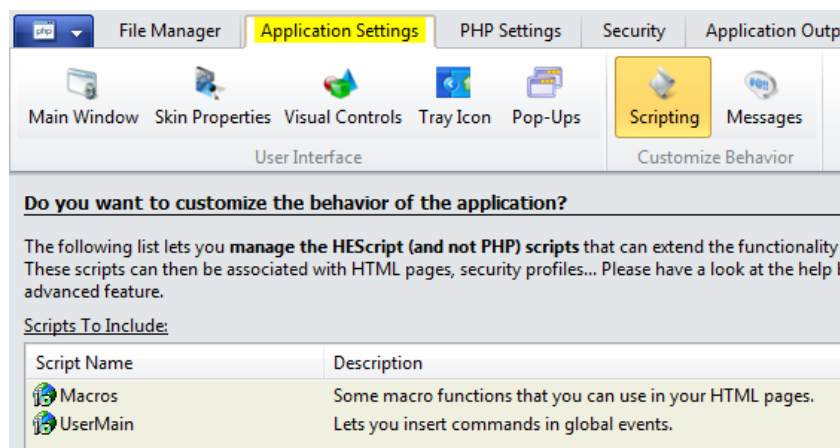
### 10.1 What is HEScript

Applications compiled with ExeOutput for PHP have a built-in script engine and they are actually driven by a bunch of internal scripts. The scripting language used by ExeOutput for PHP is called HEScript. It is based on the Object Pascal language syntax (similar to Embarcadero® Delphi) with some minor changes.

You can therefore extend the functionality of applications by writing and calling your own script functions in HEScript, in addition to PHP code itself. Of course, PHP and HEScript can be combined together.

Contrary to JavaScript or PHP, you cannot write HEScript functions directly into HTML pages because HEScript scripts are compiled into p-code when the application is being built.

You can edit HEScript scripts in ExeOutput for PHP by going to "Application Settings => Scripting":



Some scripts such as UserMain contains pre-defined events, i.e. functions that are called by ExeOutput for PHP at appropriate times.

ExeOutput for PHP provides you with a lot of built-in functions that can be called from HEScript scripts: see the documentation (Script Function Reference topic).

## 10.2 About the UserMain script

The UserMain script is always created when a new project is started. It contains a lot of global events which allow you to insert special commands.

For instance, the OnBeforeNavigate function event is triggered just before the application is going to display a webpage. Set Result to True to stop the operation.

The OnPubLoaded event is called when the application is started and before the homepage is displayed. Set Result to True if you want to exit immediately without any warning.

The OnStartMainWindow event is triggered when the main window is going to be displayed (just before the homepage is shown).

Example: if you want to display a message at startup, you could modify OnStartMainWindow:

1. Double-click on UserMain.
2. Locate the OnStartMainWindow procedure and replace it with:

```
procedure OnStartMainWindow;
```

```
begin
```

```
    // When the main window is going to be displayed (just before the homepage is shown).
```

```
    MessageBox("Welcome to my program", "Hello", MB_OK);
```

```
end;
```



```
25 Result := False;
26 end;
27
28 procedure OnPubBeingClosed;
29 begin
30 // When the application is going to be closed.
31 end;
32
33 procedure OnDisplayWindow(WindowName: String);
34 begin
35 // When a window is going to be displayed (including main and secondary windows)
36 end;
37
38 procedure OnStartMainWindow;
39 begin
40 // When the main window is going to be displayed (just before the homepage is displayed)
41 MessageBox: "Welcome to my program", "Hello", MB_OK;
42 end;
43
44 procedure OnCloseWindow(WindowName: String);
45 begin
46 // When a window is closed by the user.
47 end;
48
49 function OnTimer(TimerName: String): Boolean;
50 begin
51 // Occurs when a specified amount of time, determined by StartTimer HEScript is reached.
```

3. Click Save Script to save your changes.

## 10.3 About the Macros script

This optional script contains some pre-defined macro functions that you can call from your PHP and HTML webpages like any other functions.

## 10.4 Calling HEScript from PHP, JavaScript and with links

### 10.4.1 From PHP

ExeOutput provides you with two built-in php functions to call HEScript functions:

- `exo_runhescriptcom ( string $script )`  
where `$script` is the reference to the script's name and function to call.
- `string exo_return_hescriptcom ( string $script , string $defaultvalue )`  
where `$script` is the reference to the script's name and function to call.  
`$defaultvalue` is the value returned if the script function is not found.

Syntax for \$script: script name + dot + procedure/function name, i.e.

[scriptname].[functionprocedurename] | parameters

Example:

```
<?php
echo exo_return_hescriptcom("UserMain.ReturnDate", "Error");
?>
```

**NB:** if your procedure/function uses parameters, you can pass them using a | separator. See the documentation for further information.

### 10.4.2 From hyperlinks

You can call HEScript procedures and functions independently via HTML links.

You need to use the hescript:// prefix instead of http:// to inform the runtime module that it should execute an HEScript procedure/function.

The hescript:// prefix is followed by the script name + dot + procedure/function name, i.e. hescript://[scriptname].[functionprocedurename] | parameters

Example: hescript://Macros.MacroExit will close the application.

In HTML code:

```
<a href="hescript://Macros.MacroExit">Click here to quit this program</a>
```

In PHP:

```
<?php
exo_runhescriptcom ("Macros.MacroExit");
?>
```

**NB:** if your procedure/function uses parameters, you can pass them using a | separator. See the documentation for further information.

### 10.4.3 From JavaScript

You may finally use HEScript together with JavaScript. The Internet Explorer JavaScript model has a special function window.external that allows JavaScript scripts to communicate with HEScript scripts.

JavaScript syntax:

```
window.external.runHEScriptCom('[scriptname].[functionprocedurename]|parameters');
```

This exactly works like the `hescript://` method for HTML links; you just have to replace it by `window.external.runHEScriptCom('.....');`

Example: `window.external.runHEScriptCom('hescript://Macros.MacroExit');`

## 11 Security of your applications

### 11.1 Security concerns

During the compilation stage, ExeOutput for PHP **compresses and encodes source files** into the final executable file.

To start your application, end users have to launch the .exe file: it is not possible to unpack a compiled application using a file archiver like Zip/Unzip tools. Moreover, files such as PHP, HTML pages, images, JavaScript... are never unpacked to the hard disk: therefore, the average end user cannot extract and copy them.

Note: ensure to make a backup of your source files because once an application is compiled, you cannot extract source files from it anymore.

Applications built with ExeOutput for PHP feature several additional **security options**.

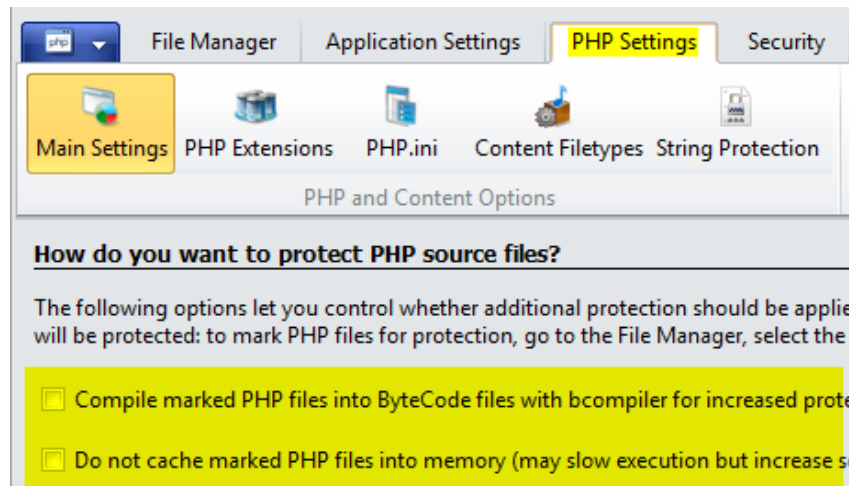
**WARNING**: it is strongly recommended not to include private passwords, database login information or security-sensitive information in applications compiled with ExeOutput for PHP and released to public. Use encryption, remote server authentication, or at least, use the string protection feature as explained below.

Since PHP scripts must be unpacked to memory in order to be interpreted by the PHP runtime, it may be possible for a skilled hacker to extract portions of compiled PHP files. To make this task even more complicate and time-consuming, ExeOutput for PHP includes some security measures such as debugger detection and internal software protection.

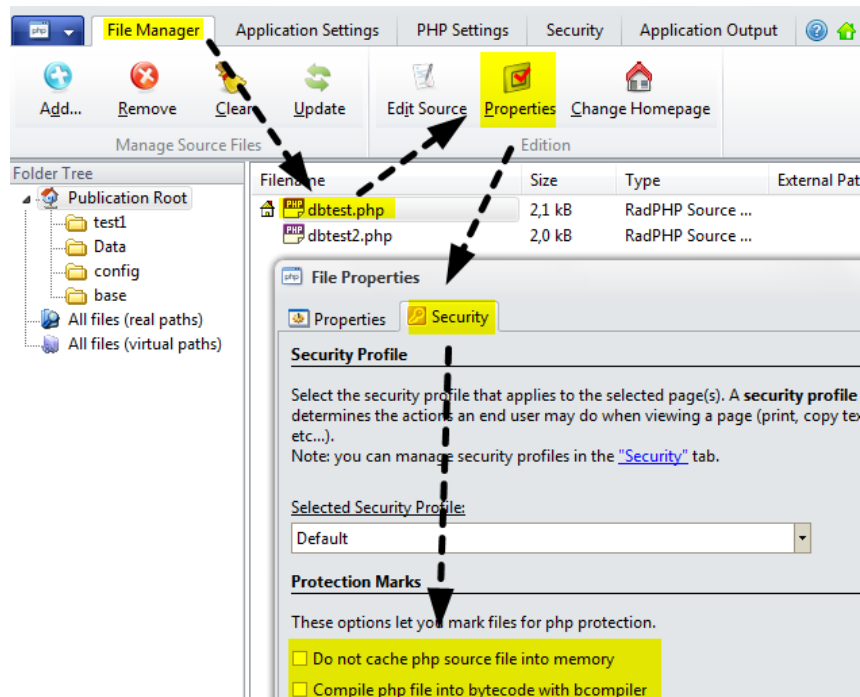
## 11.2 PHP script encoding and no caching options

ExeOutput for PHP provides you with **additional security options for sensitive PHP scripts**.

To enable these options, go to PHP Settings -> Main Settings.



These **options are global**. Since they should not be applied to all PHP scripts, **you must mark the PHP scripts** that should be protected thanks to the Protection Marks - available in File Properties (Security tab) in the File Manager:



### 11.2.1 Encoding with bcompiler

The bcompiler is a php extension that enables you to encode PHP scripts in php bytecode, making the source code difficult to reverse engineer. Further information is available at <http://www.php.net/manual/en/intro.bcompiler.php>

When a PHP file is marked to be encoded with bcompiler, ExeOutput for PHP calls bcompiler internal functions to encode the php source file into a bytecode file. Then, it is the bytecode file that is compiled into the final EXE. Nothing else is required.

#### Notes:

- bcompiler may not work with all PHP files. If ExeOutput for PHP fails to properly make conversions, the error is logged in the compilation log and ExeOutput for PHP compiles the original php source file.
- php scripts that use require, include may not properly work with bcompiler.

### 11.2.2 Do not cache PHP files into memory

PHP scripts are unpacked to memory in order to be interpreted by the PHP runtime. Since some PHP scripts may be required several times (includes for example), ExeOutput for PHP will cache them in memory. This means that they remain in memory until the cache is full or the application is closed. The application is thus more responsive, since the decompression step is skipped.

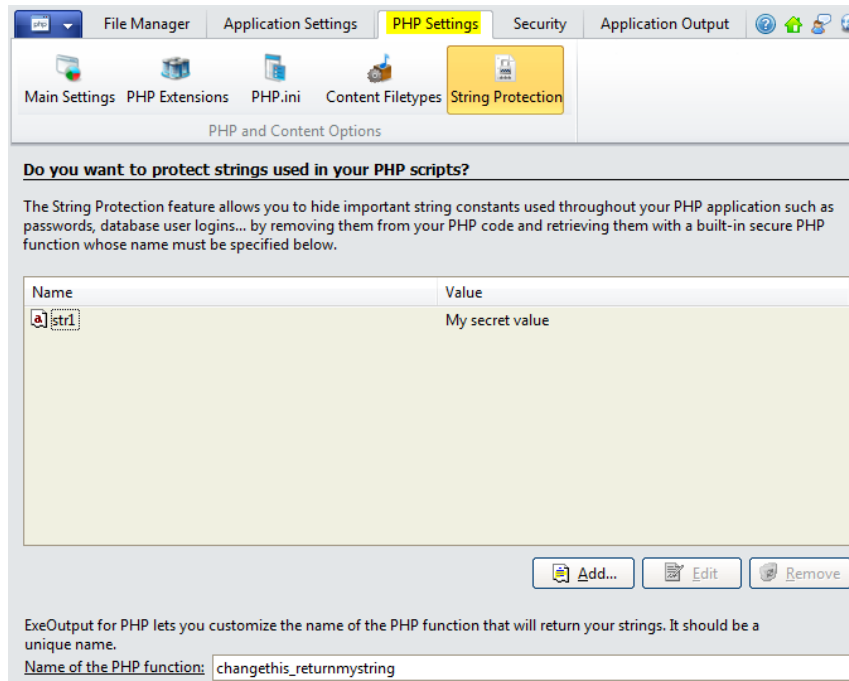
This option lets you **decide which PHP scripts should not be kept in memory after execution**. Note that in this case, ExeOutput for PHP will always have to decompress non-cached PHP scripts each time they are requested by the PHP runtime.

## 11.3 Securing your sensitive strings

**WARNING:** it is strongly recommended not to use private passwords, database login info or security sensitive information in applications compiled with ExeOutput for PHP and released to public. Even if there are security measures, remember that any computer program might be studied with reverse engineering and cracked. There is no 100% security option.

The String Protection feature allows you to hide string constants used throughout your application by replacing them in your PHP code by a call to a custom PHP function. Thus, strings do not appear in plain text in your PHP code.

To use this feature, you define the strings you want to protect in the PHP Settings -> String Protection page.



Each string gets a unique identifier. You also give a name to the PHP function that will be used to retrieve the string at runtime. After that, replace the string with a call to the PHP function.

The prototype of the php function is this:

```
string customname ( string $stringid );
```

For instance, we have a password "My secret value" whose identifier is "str1". The PHP function name is changethis\_returnmystring.

Instead of using this php code that contains the password in plain text:

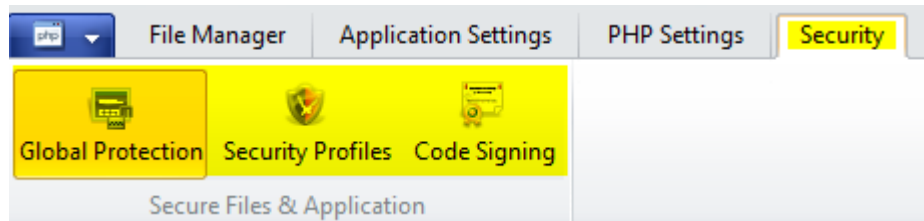
```
<?php
$pass = "My secret value";
echo ("The password is: ".$pass);
?>
```

we use the following code:

```
<?php
$pass = changethis_returnmystring("str1");
echo ("The password is: ".$pass);
?>
```

## 11.4 Global Password, Expiration Date

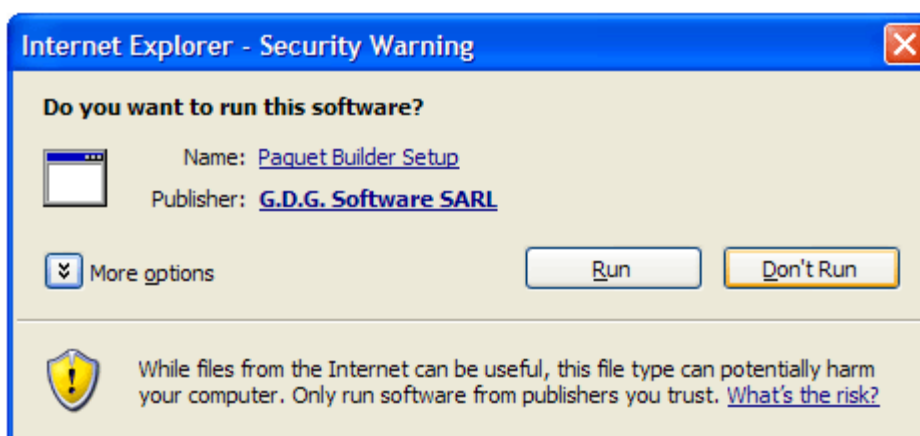
If you wish to restrict the access to your application, you can password protect it, add an expiration date, and use security profiles...



We **recommend you to digitally sign your application** (this is called code signing). Thus, you ensure end users that the code within your application has not been altered by a malicious code. Code signing is based on *Microsoft Authenticode® technology*.

With ExeOutput for PHP, it is easy to **sign your compiled application .exe files** as ExeOutput for PHP calls the necessary programs itself.


If you digitally sign your software, end users are generally presented with a **digital certificate** when your application is downloaded from the web to their system:



The code signing certificate is delivered by a **Certificate Authority (CA)**. A CA is a third party trusted by the industry, akin to a notary who handles electronic IDs. Verisign or Comodo are two examples of CA.

## 12 Links to Support and Documentation

---

This guide is not as complete as the documentation shipped with ExeOutput for PHP. To access the documentation from ExeOutput for PHP, press F1 or click .

If you have any questions or problems with ExeOutput for PHP, you can contact us with:

- The user forum:

<http://www.gdgsoft.info>

- our E-Mail address:

[info@exeoutput.com](mailto:info@exeoutput.com)

Follow us on Twitter:

<https://twitter.com/#!/gdgsoft>

Our other products are available at:

<http://www.gdgsoft.com>