PHALANGER DOCUMENTATION

USER'S GUIDE

1 CONTENTS

2 GETTING STARTED WITH PHALANGER

2.1 SPECIFICATIONS

2.1.1 REQUIREMENTS

Phalanger 2.1 is a Microsoft .NET Framework application. It requires Microsoft **.NET Framework 4.0** or a reasonably recent version of Mono. Microsoft Windows XP, Windows Vista, Windows 7, Windows Server 2003 and Windows Server 2008 (R2) are recommended operating systems for this application.

Integration with ASP.NET is the principal feature of Phalanger. This requires **Internet Information Services** (IIS) with **ASP.NET** installed (IIS6, IIS7). ASP.NET is installed automatically for IIS-enabled systems during the .NET Framework setup.

2.1.2 FEATURES

Phalanger is a complex solution giving web-application developers the ability to benefit from both the ease-of-use and effectiveness of the PHP language and the power and richness of the .NET platform. This solution enables developers to painlessly deploy and run existing PHP code on an ASP.NET web server and develop cross-platform extensions to such code taking profit from the best from both sides. Compatible with PHP 5.1 as well as with proposed features from the upcoming PHP 6, the object model in Phalanger enables to combine PHP objects with the .NET ones. It is possible to use and extend any .NET class in PHP and also consume classes written in PHP from an arbitrary .NET language (C#, VB.NET, C++/CLI …).

From another point of view, Phalanger provides .NET programmers with the giant amount of practical PHP functions and data structures - many of them reimplemented in the managed environment of the .NET Framework. The whole library of PHP functions and classes (including those implemented in the PHP extensions) is accessible to a .NET programmer regardless to his or her favorite programming language together with type information.

Not to omit the greatest, it must be stated that the compilation of PHP scripts gives yet more power to the existing PHP web applications inside the Phalanger environment. All the static (run-time immutable) code in the scripts gets parsed and compiled only once and all following accesses to a page benefit from the unleashed execution of the native compilation of the script. Yet the usage of Phalanger is not limited to web applications. The compiler supports output of standalone executables or dynamic link libraries enabling you to create managed PHP console or windows applications, or library modules reusable from any other .NET Framework application.

2.2 INTRODUCTION TO PHALANGER

Phalanger is a powerful solution for all the web-application developers who wish to get as much as possible from their PHP skills while staying on the Windows platform.

As a Phalanger-enabled developer you can easily produce and deploy powerful web applications in your favorite PHP language without the need for switching from your Visual Studio. You can benefit from the rapid development of PHP scripts combined with the richness of the function libraries inside the PHP extensions and the .NET Base Class Library, and from the power and robustness of .NET Framework.

With Phalanger you may reuse your PHP code in any .NET Framework application. Just compile your scripts as a Phalanger Library and reference it in your application. Read more about the Phalanger Console Applications and Libraries in 4.2.

In addition to PHP application development you may of course easily benefit of the whole Phalanger Class Library functionality in your .NET applications. The Class Library is designed to be easy to use whether from PHP scripts or .NET applications. You may even use the functionality provided by the PHP extensions in your managed code as simply as if you were using .NET Framework classes and methods. Refer to Using the Class Library and Extensions in 4.2 for more information.

In all the above scenarios Phalanger beats its opponents in the overall performance. In contrast to the original PHP interpreter, our module is compiling the scripts into the MSIL (Microsoft Intermediate Language). While the first page access is delayed due to compilation, all following accesses benefit of the fast execution of the native code which is always far more efficient than script interpretation. This gain is most important in case of common PHP-script libraries (for example PHPLIB, PEAR, Nuke etc.) which are complex yet immutable and thus their script files are compiled only once. On the other hand there may be some code in the PHP scripts which is not known at the time of compilation and has to be compiled at run-time. Although this procedure handicaps script compilation a little, it is known that PHP constructs imposing run-time compilation are used rather rarely and if, there is not much code to deal with.

Phalanger is the best choice for heavy load enterprise web-applications where you get the most of the robustness and the scalability of the underlying ASP.NET infrastructure, while keeping the ease of development and management of the PHP code. Thanks to the sophisticated caching mechanism of the ASP.NET applied on the Phalanger compilations, the effect of pre-compilation time penalty is soon compensated by the time-saving native execution on heavy traffic sites.

The principal goal of the Phalanger project is to enable full functionality of existing PHP scripts without any modification. The only condition is that these scripts do rely neither on special functionality provided by the UNIX platform or the Apache server nor on undocumented, obsoleted or even faulty functionality of the PHP interpreter. A code not requiring such special support from the platform may be deployed just by copying into a virtual directory and a few configuration settings.

2.3 SAMPLE APPLICATIONS

During the installation process, you can choose to configure IIS 7 for sample applications. This will also create App Pool named “PhalangerAppPool” with recommended settings for Phalanger web sites (enables 32bit mode, .NET v4.0, and integrated pipeline).

We added several sample applications to the installation package to make you familiar with the Phalanger features as quickly as possible. Those samples comprise of several PHP web pages, PHP console applications, a WinForm application, and applications written in C# using Phalanger to access extensions. To test the sample applications immediately after installation you can click on shortcuts created during installation in your Start menu (folder Phalanger, subfolder Samples). If you have chosen the " *Create IIS 7 web samples on your IIS 7 if installed.*" option during the installation, shortcuts to the sample web applications would also be added to the menu. Then if you click on a web page shortcut, a new window of your default browser will be opened and the respective web application will be started. Note that the first request to an application is always a little bit delayed but the following ones are as fast as the compiled code executes on the .NET Framework. A list of applications shipped with Phalanger follows.

- **A web application containing several elementary scripts**

Three scripts working with HTML forms, uploaded files, session and configuration are installed to a virtual path `localhost/Phalanger_SimpleScripts/` where they can be run from.

- **Interactive tester of the Phalanger features web application**

We developed an interactive tester which takes PHP scripts source codes and executes them. It shows the output and the source code of selected script. This application can be used for comfortable interactive testing of the Class Library functions, for example.

`localhost/Phalanger_Tests/`

- **A simple ASP.NET application that uses PHP as the code-behind language**

`localhost/Phalanger_ASP.NET/`

- **Simple PHP console applications**

Four console applications which illustrate the calls to the class library file-system functions, inclusions and usage of the PHP 5 classes, input from a console and finally a simple calculator. Applications are placed in:

```
{Install Directory}\Samples\ConsoleApplication
```

A shortcut to this location is added to the Start menu during installation.

- **A windows application that uses Windows Forms .NET libraries**

The application takes advantage of Windows Forms 2.0 (classes in the System.Windows.Forms namespace) to visualize the Mandelbrot set in a window.

- **A console application demonstrating the support of .NET 2.0 Generics and LINQ**

Phalanger 2.0 can produce, consume, and extend generic types! Also supported are generic methods, generic delegates, and a basic LINQ (Language INtegrated Query).

- **An example of how to use PHP extensions from the C# programming language**

Three C# console applications which present how the extensions *exif*, *ming* and *zlib* can be called from the C# programming language. A VS.NET solution containing all three C# projects is placed in `{Install Directory}\Samples\Extensions\Extensions.sln`

Exif extension sample

Uses the *exif* extension to extract EXIF headers from an image file. This extension requires the following line to be present in a configuration file:

```
<add assembly="php_exif.mng, Version=2.1.0.0, Culture=neutral, PublicKeyToken=4ef6ed87c53048a3" section="exif" />
```

Ming extension sample

Uses the *ming* extension to create a Flash movie with a rotating red square. The movie is saved as "`ming_test.swf`" in the current directory. This extension requires the following line to be present in a configuration file:

```
<add assembly="php_ming.mng, Version=2.1.0.0, Culture=neutral, PublicKeyToken=4ef6ed87c53048a3" section="ming" />
```

Zlib extension sample

Uses the *zlib* extension to create a .gz file and compress a string. This extension requires the following line to be present in a configuration file:

```
<add assembly="php_zlib.mng, Version=2.1.0.0, Culture=neutral, PublicKeyToken=4ef6ed87c53048a3" section="zlib" />
```

3 INSTALLATION

The Phalanger Core, Compiler, Class Library and tools are installed on your system. The installer lets you decide whether you want it to integrate Phalanger with IIS. If you choose to perform the integration, several virtual directories with sample applications will be created on your `localhost` IIS web site. If you do not let the installer perform the integration, you can set up your IIS manually

- See **Error! Reference source not found.**
- See IIS 7 & 7.5 (Integrated Pipeline) manual Web Site setup

The installer will copy all files to the chosen directory and place copies of several assemblies into the Global Assembly Cache (GAC). The Phalanger configuration section is automatically added to the MACHINE.CONFIG configuration file that is located in:

`$WinDir$\Microsoft.NET\Framework\v4.0.xxx\CONFIG\`

`$WinDir$\Microsoft.NET\Framework64\v4.0.xxx\CONFIG\` (on x64 systems)

Shortcuts to the PHALANGER COMMAND PROMPT, samples, and this guide are created in the **Phalanger 2.1** start menu folder.

Directory structure created in the chosen target directory:

Directory	Description
`\Bin`	Binaries of the core, compiler, class library and tools. The PHALANGER COMMAND PROMPT start menu shortcut appends this directory to the PATH environment variable.
`\Doc`	User reference documentation.
`\Dynamic`	Contains pre-generated dynamic wrappers of class libraries.
`\Extensions`	Contains a set of PHP extensions that have been tested with Phalanger and some of their dependencies.
`\Samples`	Non-web samples. Available from the PHALANGER start menu folder through shortcuts.
`\TypeDefs`	Contains type definition files used for generating managed wrappers of extensions.
`\WebRoot`	Web samples. If you've chosen to integrate Phalanger with IIS, these samples are accessible at `http://localhost/Phalanger_SimpleScripts`, `http://localhost/ Phalanger_ASP.NET` and `http://localhost/Phalanger_Tests`.
`\Wrappers`	Contains pre-generated managed wrappers of PHP extensions. The installer places all wrappers into GAC during installation.

Please note that not all the PHP extensions in the EXTENSIONS directory are enabled by default. Edit the configuration file to enable extensions (*machine.config*). To use a PHP extension that is not distributed with Phalanger, follow these steps:

1. Make sure that the extension is intended to be used with PHP 4. PHP 5 extensions are not supported.
2. Place the extension into the EXTENSIONS directory.
3. (Optional, advanced) Write a type definition file named `[extension_name].xml` and put it into the `TypeDefs` directory. Please see the Developer's Guide for details on type definition files.
4. Execute `Bin\extutil -w` to generate a managed wrapper of the extension.
5. Verify that a new file named `[extension_name].mng.dll` was created in the `Wrappers` directory. Add the wrapper assembly to GAC. Now you can use the extension.

The Phalanger project is a complex solution for PHP application development and deployment. Please note that Phalanger is a tool rather that a product and thus the following chapter describes the various situations Phalanger can be helpful to application developers. If you are interested in ready-to-use applications please refer to the Sample Applications in 2.3.

3.1 IIS 7 & 7.5 (INTEGRATED PIPELINE) MANUAL WEB SITE SETUP

1. open `Internet Information Services (IIS) Manager`
2. add new application (or map an application to an existing directory)
3. Manage the app, Advanced Settings - Set Default AppPool (or your own, Integrated pipeline)
4. (if you need native extensions) within the Application Pools, ensure your pool runs in 32bit
    a. right click on the pool, Advanced Settings
    b. Enable 32-Bit Applications = true
5. in your website's web.config, add following into *<configuration>* section

```
<system.webServer>
<handlers>
<add name="Phalanger" path="*.php" verb="*" type="PHP.Core.RequestHandler, PhpNetCore,
Version=2.1.0.0, Culture=neutral, PublicKeyToken=0a8e8c4c76728c71"/>
</handlers>
</system.webServer>
```

6. go to your application, open Default Document, add "index.php", "default.php"

4 CHAPTER 3 - USAGE OF PHALANGER

4.1 USING THE CLASS LIBRARY AND EXTENSIONS FROM C#

The Phalanger Core and the Class Library are offering many classes and methods which can other programs on the .NET platform benefit from. Besides, the managed wrappers of the PHP extensions allow the .NET programmers to use any function provided by a PHP extension. Although Phalanger was primarily developed to introduce the PHP language to .NET platform, the first straightforward deployment scenario is to reference its assemblies in other .NET programs. It suffices to place references to the Core or the Class Library assemblies to the list of referenced assemblies of your program and then you can use classes from the namespace *PHP.Library* (the Class Library) and *PHP.Core* (the Core). If you want to use managed wrappers just add a reference to those which you are interested in. There is a sample application on this scenario in the Samples (see **Section 1.3**).

4.2 CREATING PHALANGER CONSOLE APPLICATIONS AND LIBRARIES

If you are programming in the PHP language you can now use this language to write console applications or class libraries. This is the second possibility how to use Phalanger.

A console application is produced by the `phpc` command-line compiler if it receives the `/target:exe` option. A library of PHP functions and classes is generated if the `/target:dll` option is supplied. To display the list of command-line compiler's options type `phpc /help` on the command line. The shell window with predefined paths to the compiler can be opened via the shortcut "Phalanger Command Prompt" which is added to your Start menu.

Console applications and libraries comprise of a set of scripts which are compiled together into one multi-module assembly. There is no major difference between a console applications and a library. The only one is that the library is not executable. Each executable assembly should have a method called entry point by which its execution is started. For C# console applications the entry point is always the public static method Main() of the class chosen by the user in the C# project properties or supplied on the command line of `csc` C# compiler. In the case of Phalanger, the user chooses a script which is started as the first one. This can be set in the PHP project properties or by supplying an option /entrypoint to the `phpc` command-line compiler. See the 2.3 for samples.

If you distribute your Phalanger console application you should distribute Phalanger along with it. However, it suffices to copy a Phalanger.msi installation package since the others are not necessary to run the application.

As it is usual on the .NET platform, a console application whose executable is named *App.exe* reads its configuration from the file *App.exe.config* located in the same directory. The prototypic configuration file (named *Application.config*) for the console applications containing all supported configuration options along with their default values and respective PHP names can be found in the `{Install Directory}/Samples`. So you can copy it to the bin directory of you application, rename it and modify it. Note, that only those options whose values differ from the default ones need to be stated in that file. The others can be omitted.

4.3 DEPLOYING PHALANGER WEB APPLICATIONS

The most complex and also the most powerful features of Phalanger focus the world of web applications. A web application written in Phalanger has much in common with the applications written in the ASP.NET since Phalanger is integrated into the ASP.NET and exploits its great features such as the configuration management, session handling, file uploading, source code modification watching and much more. So, if you are familiar with ASP.NET, you will find Phalanger web applications similar to the ASP.NET ones.

A Phalanger web application comprises of script source files placed in a virtual directory on the IIS web server. This directory should be configured as a web application as it is described in 3. Some sample web applications are installed if you have chosen so in the Phalanger installation.

The application is configured by the *Web.config* files which may be placed in any subdirectory of the virtual directory. Configuration contained in the *Web.config* file is applied on all scripts in the directory where it is placed and in all subdirectories as it is usual in the ASP.NET web applications. Phalanger has many configuration options which can be set in the *Web.config* file. The prototypic *Web.config* file which contains all supported configuration options along with their default values and respective PHP names can be found in the directory `{Install Directory}/WebRoot/Samples/SimpleScripts`. If you are configuring your own application, you don't need to state all options in your configuration file but only those which values differ from the default ones.

As Phalanger handles requests, sessions and file uploading via ASP.NET, the following ASP.NET configuration options are also applicable (see ASP.NET documentation at MSDN for details):

- section system.web, node globalization
- section system.web, node sessionState
- section system.web, node httpRuntime

Since the source files of the web applications are placed on the server in the virtual directory, they can be modified while some requests are still pending. This is handled correctly by Phalanger (in the same way as the APS.NET does). So you can modify source files without caring about how they are recompiled because Phalanger does it for you automatically.

There are several sample web applications shipped with Phalanger as described in the Sample Applications in 2.3.

5 TROUBLESHOOTING

Problem: Global variables from a request or a session are reported to be undefined

Check whether the *RegisterGlobals* configuration option has its value set to "`true`". The default value is "`false`" thus there is no WEB.CONFIG file or the configuration option is missing there the global variables are not registered.

Problem: Wrong encoding is used in the Phalanger web application pages.

Check *Machine.config* configuration file for values set in *system.web* section, node *globalization*, attributes *requestEncoding* and *responseEncoding*. Values of these attributes are used to encode responses by the ASP.NET server. The Phalanger output is going through the ASP.NET buffers which perform the encoding.

Problem: A source code of a script is displayed instead of its output.

Check whether IIS is well configured, i.e. whether a virtual directory exists, whether a web application is created on that virtual directory and the application handler is configured. See 3.1.

Problem: After uninstalling Phalanger, the installation directory remains on disk and cannot be deleted.

The directory is locked because there are still some Phalanger applications running. Close all Phalanger applications and then try again. It might also be necessary to unload Phalanger web application from IIS or completely stop IIS.

Problem: Phalanger web pages don't work, IIS returns "500 Internal Error".

If you installed IIS after installing .NET Framework, ASP.NET must be registered with IIS manually. Run `cmd.exe`, switch to the `$WinDir$\Microsoft.NET\Framework\v4.0.xxx` directory and execute `aspnet_regiis -i`.

Problem: php4ts.dll could not be loaded

First ensure php4ts.dll is in global assembly cache. If you are running on x64 environment, and if you are using native extensions (php4 extensions), you must force the application to run in 32bit. (in case of website, see - IIS 7 & 7.5 (Integrated Pipeline) manual Web Site setup)