The PHP Company

Zend Server Community Edition 5.x
Reference Manual

By Zend Technologies

www.zend.com

This is the Installation Guide for Zend Server Community Edition, Version 5.0.

Zend Server Community Edition Installation Guide, issued February 2010.
DN: ZCE-IG-210210-5.0-001

Table of Contents

Overview

Overview Zend Server is a complete, enterprise-ready Web Application Server for running and managing PHP applications that require a high level of reliability, performance and security.

What is Included in Zend Server:

Business-grade PHP	An up-to-date, tested and supported PHP stack ensures high reliability, enhances security and increases staff productivity.
Deployment with confidence	A complete and consistent environment used in development, testing and production eliminates many of the problems encountered during deployment.
Rapid response to problems	Advanced application monitoring and diagnostics enable early problem detection and quick root cause analysis.
Top application performance	Built-in optimization and acceleration ensures high performance and low resource utilization.

Code Tracing: Solve Problems Faster Than Ever!

Finding the root cause of problems, especially when they occur in the production environment, is a time-sink for developers and system administrators. Zend Server 5.0 applies the concept of a black box flight recorder to PHP. It can record live application execution in testing or production, helping you quickly pinpoint bugs and performance bottlenecks in your code.

Job Queue: Offload Execution of Long-running PHP Scripts

Web applications generally follow the synchronous communication paradigm, however some tasks are better suited to asynchronous execution. Long-running report generation, order processing, database cleanup, and pulling of RSS feeds are some examples of jobs that can be executed asynchronously. Zend Server 5.0 incorporates *Job Queue*, providing full support for creating, executing and managing jobs to optimize application performance and reduce server load.

A Web Application Server for Your Application

If you're developing or running a business-critical PHP application on a couple of servers, Zend Server is the right solution for you. In cases where your application runs on a large number of servers, or if you require session clustering or a job queue, Zend Platform Enterprise Solution could suit your need.

Enhance PHP Application Reliability and Security

Tracking, installing, configuring and testing dozens of PHP libraries and drivers is a time sink for developers, testers and administrators. The rapid updates and code changes in today's fast-paced Web application arena further aggravate the challenges of maintaining reliable and secure PHP runtime environments.

Zend Server customers have access to Zend's technical support, and receive online software updates, hot fixes and security patches, to ensure they run the most reliable, secure, and up-to-date version of PHP. Read about the Service Level Agreement (SLA) Zend provides to its customers.

Ensure Successful Deployments

Many of the problems encountered during application deployment or in production occur because different PHP versions and configurations are used in development, testing and production. Zend Server enables you to deploy your PHP applications with confidence, ensuring every member of your team uses the same, highly reliable environment consistently through each stage of the application life cycle. If you ship your PHP application to a remote customer, Zend Server's unattended installer facilitates fast and trouble-free deployments.

Detect Problems Before the Phone Rings

When things go wrong with your application, you want to know about it as soon as possible, and resolve the problem before end-users are impacted. Zend Server enables you to take a proactive approach when it comes to ensuring the best user experience by monitoring PHP application execution and alerting you to critical problems such as:

- Slow PHP script execution
- PHP errors
- Errors in specific function calls
- Excess memory usage
- Errors in called Java code
- And more…

Quickly Pinpoint Root Cause of Problem

Knowing that a problem occurred is an essential first step, yet what really counts is how fast you can isolate its root cause and deliver a solution. Zend Server slashes root cause analysis time by capturing application execution data, such as variable values, call stack and environment information, for every detected incident. Developers can further analyze captured data in Zend Studio, thereby eliminating the time-consuming task of reproducing production problems in a lab.

Boost Application Performance

A high-quality user experience is expected from business-critical Web applications, even during peak loads, yet deploying more hardware to increase performance may prove to be costly. Zend Server provides multiple capabilities for improving application response times and minimizing resource utilization.

- Code Acceleration – PHP bytecode caching increases performance with no application changes
- Full page caching – A URL-based HTML output cache that does not require any application changes
- Partial page caching – A set of functions that allow developers to cache data in shared memory or to disk

Zend Server Community Edition (CE)

The community edition of Zend Server is a free, simple PHP Web Application Server environment that is ideal for running non-critical PHP applications or just for experimenting with PHP.
Zend Server Community Edition is a fast and reliable PHP application stack. It is completely free, and you can use it in development, testing and production.

Boost Performance of your PHP Applications

Zend Server Community Edition provides multiple capabilities for improving application response times and minimizing resource utilization:

- PHP bytecode caching (Zend Optimizer+) - increases performance with no application changes
- Data caching - a set of functions that allow developers to cache data in shared memory or to disk

Use a Reliable PHP Stack in Development and Production

Zend Server Community Edition is a pre-integrated PHP application stack that's been tested by Zend to ensure the highest levels of reliability. You can use it to run your application in production, during development and testing, ensuring a consistent environment throughout the application lifecycle.
If at any point you require technical support, software updates, security patches, application monitoring or extra performance, you can simply upgrade to Zend Server, the commercial version of Zend Server Community Edition.

Get Up and Running with a Full PHP Stack in Minutes

Eliminate wasted time spent on putting together your PHP stack piece by piece. Zend Server Community Edition includes everything you need, whether you're using Windows, Linux or Mac OS X. The simple, native installers will set you up in minutes with:

- Bytecode accelerator (Optimizer+)
- Zend Data Cache
- A certified PHP distribution
- Zend Framework
- Apache (or IIS integration)
- MySQL (on Windows and Mac OS X)
- Out-of-the-box connectivity to all common databases
- Java code connectivity
- Web-based PHP administrator console

About

Zend Server Community Edition includes a tested and certified version of PHP and a set of tools to set up and optimize your environment.
These tools are presented in an improved Administration Interface designed to provide all the tools and technology necessary to support PHP environments.

Special attention has been given to creating consistency across operating systems to ensure interoperability and facilitate the needs of diverse environments that use Linux, and Windows and Mac operating systems.

The PHP versions are PHP 5.2 and PHP 5.3, which have been tested and optimized for development use. Commonly used extensions and Zend Framework are included with the PHP to provide a one-stop shop for all the resources that developers need to create PHP Web applications.

A complementary set of tools is provided with Zend Server Community Edition to optimize and extend your PHP capabilities. The tools included in Zend Server Community Edition are described in detail in the Components Section. Instructions on how to work with each component are provided in the Tasks section, where each possible task is described in detail from start to end.

To get started with Zend Server Community Edition, click here.

Installation Directories

Not all users decide to install their software in the same location. To reflect this actuality, all paths in this document have been replaced with the following prefix: <install_path>. This represents the location of the installed files. If you used the default settings, the location should be as follows:

- Windows: C:\Program Files\Zend\ZendServer
- Windows 64 bit C:\Program Files (x86)\Zend\ZendServer
- DEB/RPM: /usr/local/zend
- Tarball: /usr/local/zend
- Mac: /usr/local/zend

Password Management

After completing the Installation process and opening Zend Server Community Edition, a password definition page is displayed for first time users. This page only appears once to define the Administration Interface's login password.

For security reasons, Zend Server Community Edition cannot restore your password for you. However, you can reset your password if you have access to the application's files and Administrator privileges.

The following procedure describes how to reset a lost password from **outside** the Administration Interface.

To reset your password:

In Windows:

1. In the Start menu locate the Zend Server Community Edition section and select **Zend | Change Password**. Your password is reset.
2. The next time you log in to the Administration Interface, you will be prompted to set a new password.

Other operating systems:

1. From the command line, run *gui_passwd.sh* that is located in: *<install_path>/bin*
2. You will be prompted to enter a new password.

Correct completion of this procedure in Windows: Zend Server Community Edition displays the password definition page.

Correct completion of this procedure in other operating systems: You can log in with the new password.

If you are unable to change your password, refer to the Support Center for further information.

The following procedure describes how to change your password from **inside** the Zend Server Community Edition Administration Interface.

To change your password from inside the Administration Interface:

1. In the Administration Interface, go to **Administration | Password** and License.
2. Enter your current password and enter your new password in the next two fields.
3. Click "Change Password" to apply changes.

Correct completion of this procedure results in Zend Server Community Edition requiring you to log in with the new password the next time you access the Administration interface.

Support

Zend Technologies provides a wide range of resources for obtaining additional information and support, such as the Zend Support Center, the Zend Newsletter, and the Zend Developer Zone.

Zend Support Center

The Zend Support Center is a portal for information on all Zend Product related issues.
From the Zend Support Center you can access:

Zend Forums

Hosted user forums for the Zend product user community. See what other users have to say and get answers directly from the Zend Support team. Visit: http://forums.zend.com

Zend Support Knowledge Base

The Zend Knowledge Base contains an extensive range of articles on commonly encountered problems, troubleshooting, tips and work-arounds.
Search the Knowledge Base for any Zend product related issue at https://www.zend.com/en/support/knowledgebase.php.

Online Documentation

The Zend Product Online Documentation Center can be easily browsed and searched as a resource for accessing the most to date information on using all Zend Products. Visit: http://www.zend.com/en/resources/zend-documentation/

Open a Support Ticket (Only Available in Zend Server)

If you did not find the answer to your question in any of the Support resources, you can open a ticket with the Zend Support team, who will answer each ticket on an individual basis. This can be done through https://www.zend.com/en/helpdesk/newticket.php.

In Zend Server CE, the Community Edition, all Support is administered via the Forum.

Zend PHP Email Updates

Sign up for Zend PHP email updates for the hottest updates, special promotions and useful developer information.
To sign up, log in to your Zend account at https://www.zend.com/en/user/myzend, enter your email address and click Subscribe.

Zend Developer Zone Resource Center

The Zend Developer Zone is the leading resource center for PHP developers, where you can learn about PHP and meet the experts.

The Zend Developer Zone features the following:

- The PHP 5 Info Center
- Articles and Tutorials
- PHP and PEAR News Weeklies
- Worldwide Job Classifieds

Visit: http://devzone.zend.com

Feedback

Send feedback, questions and comments on the Online Help and Documentation to: documentation@zend.com.

Concepts

General Layout

Zend Server Community Edition's Administration Interface is the main area for configuring and managing your development environment.
The Administration Interface is accessed through your browser by entering the link that is provided at the end of the installation process. Login is done through the Password administration page that appears when you access the Administration Interface for the first time.
Click here for more about configuring your password.

The Administration Interface is comprised of the following tabs:

- **Monitor** - The Monitor tab is the main area for system information and it includes Dashboard | Server Info | PHP Info | Logs
- **Setup** - The Setup tab is the main area for configuring your PHP and it includes Components | Extensions | Directives | Debugger

In addition to the main Administration Interface, Zend Server Community Edition comes with a tray utility called the Zend Controller that provides quick access to:

Monitor Tab

Dashboard

The Dashboard page is accessed from **Monitor | Dashboard** and is the default page after logging in to the Administration Interface.

The Dashboard page is a summary of information and quick links. The information in this page is divided into Recent Events, Tasks and a System Overview.

Server Info

The Server Info page is accessed from **Monitor | Server Info.**

The Server Info page displays the details of your environment. The information displayed in this page is as follows:

- **Zend Server** - Product version.
- **PHP** - PHP version and the path to your PHP configuration file (php.ini). This information can also be accessed from the Administration Interface, on the PHP Info page.
- **Web Server** - Your Web server's IP, type and the operating system used to run the Web server.
- **Zend Framework** - Release version and directory location in your computer.
- **Zend Data Cache** - Release version and status.
- **Zend Debugger** - Release version and status.
- **Zend Guard Loader -** Release version and status.
- **Zend Java Bridge -** Release version and status.
- **Zend Optimizer+** - The status of the Optimizer+ component used for opcode caching and optimizations.

If your PHP application is business-critical, you probably want to make sure that your PHP runtime environment is up to date. Zend Server Updater ensures that you have the latest versions of PHP, Zend Server Components and Extensions. This feature is available only in the commercial version of Zend Server.

PHP Info

The PHP Info page is accessed from **Monitor | PHP Info.**

The PHP Info screen is a read-only page that outputs a large amount of information about the current state of PHP. It is an easily accessible representation of information contained in the php.ini file, including information about PHP compilation options and extensions, the PHP version, server information and environment, PHP environment, OS version information, paths, master and local values of configuration options, HTTP headers and the PHP License.

Note:

The values displayed in the PHP Info page may differ from the system-wide settings displayed further down the page in the "Local View" column of the Configuration section. To see the system-wide settings, view information listed in the "Master Value" column.

If your PHP application is business-critical, you probably want to make sure that your PHP runtime environment is up to date. Zend Server Updater ensures that you have the latest versions of PHP, Zend Server Components and Extensions. This feature is available only in the commercial version of Zend Server.

Changing PHP Info

The Administration Interface allows easy changing of PHP info through the Setup tab. Any changes made in the Extensions, Components and Directives pages will be automatically updated in your php.ini file and will be reflected in the PHP Info page.

Note:

Configuration changes will only take effect once you PHP has been restarted by clicking

More information about the PHP Info display can be found in the PHP Manual, accessed by going to "PHP Options and Information" - External Link.

Logs

The Logs page is accessed from **Monitor | Logs**.

The Logs page is a means for developers to view log information directly from the Administration Interface. This information can be used to investigate unwanted activity in your environment in terms of errors and application behavior.
The logs displayed in this page consist of the system logs, as determined by the type of Web server you use:

- Apache servers include three logs - PHP Error log, Apache Error log and Server Access log - all of which reference the installation locations (except for the PHP Error log, which comes from the error_log directive).
- IIS servers include the PHP Error log.

Power users can edit the XML file to include additional logs. For more information on adding logs to the Logs page, see Working with Logs.

From this page you can:

- View Logs
- Filter Logs
- Navigate inside a log
- Add Logs

Setup Tab

Components

The Components page is accessed from **Server Setup | Components.**

The Components page provides a convenient way to view and configure the components installed in your environment.

From this page, when applicable you can for each rule:

- **Turn On/Off -** See table below for component specific information.
- **Clear -** Empties cache information.
- **Configure Directives -** Clicking this link directs you to a pre-filtered view of the directives (in **Server Setup | Directives**) that belong to the component.
- **View Description -** at the end of each row of the table is a small icon that displays a tooltip that describes the component.

Additional actions for Specific rules:

- **Zend Debugger | Allowed Clients** - Clicking this link directs you to **Server Setup| Debugger** where you can define the IP addresses that can or are prohibited to connect.
- **Zend *Job Queue* | Queue Setup** - Clicking this link directs you to **Server Setup | Job Queue** where you can define global Job Queue settings.
- **Zend Monitor | Monitoring Rules** - Clicking this link directs you to **Rule Management | Monitoring** where you can define and activate monitor rule settings.
- **Zend Page Cache | Caching Rules** - Clicking this link directs you to **Rule Management | Caching** where you can create and edit cache rules.

Note:

The following message appears when an option was not installed: "This component is not installed, for instructions see the Installation Guide". For Windows see Windows Installation, for DEB see DEB Installation and for RPM see RPM Installation.

The following components can be turned On/Off and configured as follows:

Component	Status	Comments
Zend Data Cache	**On** - Activates the Data Cache: Scripts that include the Data Cache API can run. **Off** - Disables the Data Cache: Scripts that include the Data Cache API **cannot** run.	This component stores information and therefore has an additional action for clearing information.
Zend Optimizer+	**On** - PHP is optimized. **Off** - PHP **is not** optimized.	This component stores information and therefore has an additional action for clearing information.
Zend Java Bridge	**On** - The Java Bridge runs: Scripts containing the Java Bridge API can run. **Off** - The Java Bridge stops running: Scripts containing the Java Bridge API **cannot** run.	This component can be restarted.
Zend Debugger	**On** - Activates the Debugger for local and remote debugging with Zend Studio. **Off** - Disables the Debugger and does not permit debugging from Zend Studio.	The Debugger requires that you enter a list of IP addresses to allow, deny or permit remote debugging through a firewall. therefore it has an additional option for adding "Allowed Clients"
Zend Guard Loader	**On** - Scripts encoded with Zend Guard run. **Off** - Scripts encoded with Zend Guard **cannot** run.	

Note:

For more information on adding additional components, see the Installation Guide.

The On/Off Status is used to configure your php.ini according to the components you want to load. If you intend to use functions related to a component in your code, verify that the extension is enabled and that the status is set to On.

Hovering the curser over the Information icon displays a brief component description.

If your PHP application is business-critical, you probably want to make sure that your PHP runtime environment is up to date. Zend Server Updater ensures that you have the latest versions of PHP, Zend Server Components and Extensions. This feature is available only in the commercial version of Zend Server.

Extensions

The PHP Extensions page is accessed from **Server Setup | Extensions.**

The PHP Extensions page provides a convenient way to view and configure extensions.
Use this page to control and configure extensions that are loaded in your environment.
To find out how to add more extensions to this list, see Adding Extensions and UNIX: Compiling PHP Extensions for Zend Server.

PHP extensions are sets of instructions that add functionality to your PHP. Extensions can also be employed to recycle frequently used code. You can place a set of functions into one extension and instruct your projects to utilize the extension. Another use for PHP extensions is to improve efficiency and/or speed. Some processor intensive functions can be better coded as an extension, rather than as straight PHP code.

If your PHP application is business-critical, you may wish be alerted to database access failures. Zend Server can monitor your application in production, alert you to failures or performance degradation, and provide you with diagnostic information for rapid root cause determination.
This feature is available only in the commercial version of Zend Server.

The Extensions page is list of the extensions included with the Zend Server Community Edition installation and extensions added to the php.ini by the user. Use the Extensions page to view the status of all your extensions and to quickly and easily load and unload extensions.
You can also configure directives associated with certain extensions. Extensions with directives that can be configured have a Configure link next to them.
Clicking the link opens the PHP Directives page, filtered to the exact directives associated with the particular extension. Click the All option in the PHP directives page to see a complete list of directives.

From this page, when applicable for each extension you can:

- **Turn Off** - The extension is not running on the machine and code that includes the Extension's functions works.
- **Turn On**- The extension is running on the machine.
- **Built in**- This applies to extensions that have dependencies, or were complied with PHP. Built-in extensions cannot be removed and thus do not have an On/Off option.
- **Directives -** Clicking this link directs you to a pre-filtered view of the directives (in **Server Setup | Directives**) that belong to the extension.
- **View Description -** at the end of each row of the table is a small icon that displays a tooltip that describes the component.

Directives

The PHP Directives Info page is accessed from **Server Setup | Directives.**

The PHP Directives page allows you to easily edit your PHP configurations from the Administration Interface. From here, you can view and configure commonly used directives. The available directives are grouped by category in expandable lists. Clicking the arrow next to the category name expands the list to expose the different options. Where relevant, input fields are added, to change a directive's value. The initial display shows the most commonly used Directives. Click "**All**" for the full list of directives or use the "**Search**" component to locate a specific directive or use *ext:<extension_name>* to find directives by extension. You can also use the **Popular** option to view commonly used directives such as directives that define directories and languages.

Debugger

The Debugger page is accessed from **Server Setup | Debugger.**

The Debugger page is used to enable remote PHP debugging and profiling of Web applications using the Zend Debugger component.
This component enables developers using the Zend IDE to connect to a remote server to analyze (debug and profile) and fix code.
Event information collected by the Monitor component can be further diagnosed with Zend Studio, provided that the machine running Zend Studio is registered as an "allowed host" and it does not appear in the "denied hosts" list. Special attention to this should be given when specifying IP ranges to make sure that necessary IPs are not included in that range. By default, your local IP (127.0.0.1) is registered as an "allowed host" by default.

The Zend Debugger page allows you to configure the hosts for the following debug options:

- Hosts allowed to initiate debugging and profiling sessions.
- Hosts denied the permission to initiate debugging and profiling sessions.

Administration Tab

Password

The Password page is accessed from **Administration | Password.**


To change or reset your password follow the instructions in Password Management.

Updating your License

You are not required to enter a license to use Zend Server Community Edition. However, you must have a valid license to use the complete edition of Zend Server Community Edition.

How do I get a license?

If you do not currently have a valid license, go to the licensing page to find out how to get a license: http://www.zend.com/en/products/server/license

I already have a license, what do I do?

If you have already purchased a license, you should have received a confirmation e-mail that includes your Order number and License key.

To enter a License:

1. Go to **Administration | Password and License**
2. In the "Update License" area, enter the Order number and License key that were included in your confirmation email.
3. Click 

to apply the changes.
4. Click 

.

Zend Server Community Edition will start to run in a fully functional mode.

License Expiration

Before your license expires, a notification is displayed at the bottom of the Administration Interface, telling you how long you have left until your license expires and where to go to renew your license.

Once a license expires, Zend Server Community Edition reverts to the Community Edition mode until a new license is entered. During this time, all the licensed features are unavailable. However, their settings are kept and will be restored, along with the functionality, when a new license is entered.

Zend Controller

The Zend Controller is accessed from the system tray by clicking on the Zend Icon  , or from the command line by running <install_path>/bin/zendcontroller.
Windows users can load the Zend Controller by going to <install_path>\bin and clicking *Zend Controller.exe.*
The Zend Controller is a system tray utility that provides quick access to frequently performed tasks and useful information.

Adding the Zend Controller to the Start Menu/System Tray/Taskbar

The Zend Controller resides in the System Tray/Taskbar. The Zend Controller may behave differently in each environment: In some systems, the Zend Controller may run as soon as the computer is started and in others, it doesn't. The following instructions are included to let you define the Controller's behavior according to your preferences:

- GNOME - View the instructions online at: http://www.ubuntugeek.com/howto-add-entries-in-gnome-menu.html
- KDE - view the KDE online documentation at: http://docs.kde.org/development/en/kdebase-workspace/kmenuedit/quickstart.html
- Windows Vista and XP and 2008:

    1. Right-click **Start** and select Properties.
    2. Click the **Start** Menu tab and click the radio button next to Classic Start menu.
    3. Click the **Customize...** button and then the **Add...** button.
    4. Click the **Browse...** button and locate the .exe file. The default location is <install_dir>\bin\ZendController.exe.
    5. Highlight the program and click **OK**. Then click **Next**.
    6. Highlight the folder in which you want the application to appear or click **New Folder...** to create a new folder. Click **Next**.
    7. Select a name for the shortcut and click **Finish**.
       Note: In Windows XP, 2003, Vista and 2008, you may need administrative rights to make changes to the Start menu, depending on the existing user profiles and privileges.

- Mac OS X

    1. Go into the System Preferences.
    2. Click on Accounts, and select your account.

3. Click on Startup Items.
4. Click the '+' sign next to the Zend Controller file. The next time the system is restarted, the Zend Controller runs at startup.

Tasks

Working with Zend Server Community Edition

The following text describes how to work with Zend Server Community Edition . Each of the tasks in this section describes a different procedure that can be used to facilitate your PHP development process.

The following table lists the different tasks, their descriptions and the expected outcome of each task:

Task	Description	Outcome
Getting Started	Review all the post installation tasks before working with Zend Server Community Edition .	Access the Administration Interface.
Working with Extensions	How to enable and disable extensions.	The environment is customized to suit your requirements.
Working with Logs	How to view and add logs.	View and define which logs are displayed.
Working with Components	How to enable and disable components (Debugger, Data Cache Guard Loader, Java Bridge).	The environment is customized to suit your requirements.
Working with Directives	How to enable and disable directives.	The environment is customized to suit your requirements.
Working with Optimizer+	How to use the Optimizer+.	Improve performance by running the Optimizer+.
Working with Zend Guard Loader	How to use the Guard Loader component.	Run code encoded with Zend Guard.
Working with Java Bridge	How to use the Java Bridge.	Extend your PHP code to reach out to Java functionality in runtime.
Working with the Debugger	How to configure the Debugger to debug and profile code running with Zend Server Community Edition.	Use the local and remote debugging features in Zend Studio for Eclipse.

Task	Description	Outcome
Working with Local Debugging	How to configure the Debugger to debug and profile code running with Zend Server Community Edition.	Use the local debugging feature in Zend Studio for Eclipse.
Working with Data Cache	How to use the Data Cache API.	Implement the Data Cache API functions into your PHP code.

Getting Started with Zend Server Community Edition

Zend Server Community Edition is a tool that requires a minimal amount of actual interaction with the Administration Interface. Once your environment is setup, apart from occasionally logging in to view your system settings or your php.ini, there are not many day-to-day activities that require the Administration Interface.

The first point of reference for working with Zend Server Community Edition is what to do after installation.

What to do After Installing Zend Server Community Edition

The following section describes the tasks that should be performed after installing Zend Server Community Edition for the first time.

These tasks cover all the different installation types (DEB, RPM,and Windows). Each task is accompanied by a description of its purpose and the expected results.

Run the Administration Interface

Purpose: To verify the installation and that the Administration Interface is accessible.

Result: the Administration Interface opens in a browser.

The Administration Interface is a Web interface that runs through a browser.

This procedure describes how to view the Administration Interface.

To view the Administration Interface:

1. To run Zend Server Community Edition locally, open a browser and enter the following URL:
   For Windows: http://localhost/ZendServer;
   For Linux/Mac: http://localhost:10081/ZendServer or https://localhost:10082/ZendServer

   If you are using a remote connection, replace localhost with your Host Name or IP.
2. The Zend Server Community Edition login screen opens and prompts you to set a password.
   This screen only appears once and is not displayed again after your password is set.

The next time you log in to Zend Server Community Edition, you are prompted for the password you set the first time you opened Zend Server Community Edition.

Configure Your Password

Purpose: To ensure that you can access the Administration Interface.

Result: Your password is created.

When you first run Zend Server Community Edition, the registration screen is displayed. Define your Zend Server Community Edition login password in this screen.

To view the different password management options, click Password Management.

Check Apache

Purpose: To verify that Apache is running.

Result: System confirmation.

This procedure describes how to check if the Apache Web server is running.

To check if the Apache server is running:

DEB, RPM : from the command line, run ps -ef | grep -E 'apache2|httpd'.

Windows: In the system tray, hover over the Apache Monitor icon to view the Apache status. If necessary, click to open a dialog with the Stop, Start and Restart options.

A notification with the Apache server status is displayed.

Note:

Every time the Apache is restarted, the following message is displayed: "httpd: Could not reliably determine the server's fully qualified domain name, using 127.0.0.1 for ServerName".

To resolve this situation, add a line to the Apache configuration file, as follows:

Open the file <install_path>/apache2/conf/httpd.conf and add the following line, placing your server's Host name in the brackets: ServerName [server name]

Check IIS

Purpose: To verify that the bundled Apache is installed and running.

Result: System confirmation.

This procedure describes how to check if the IIS server is running.

To check if the IIS server is running:

Use Microsoft: http://support.microsoft.com/kb/314771 [^]

Look for the presence of the following registry key:

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\InetStp

-or-

Issue the following command in cmd :

Iisreset /status

If the following message is received, then IIS is not running:

“'iisreset' is not recognized as an internal or external command, operable program or batch file.” ---&61664; not installed

If the following messages are received, then IIS is running:

“Status for Windows Process Activation Service (WAS) : Running"

"Status for World Wide Web Publishing Service (W3SVC) : Running” ---&61664; installed

Run a Test on Your Web Server

Purpose: To verify that the installed Web server is running properly.

Result: The "Hello World" message is displayed in your browser.

This procedure describes how to run a test PHP script.

To run a simple test script:

1. Create a file called hello.php
2. Enter the following code into the file:

```
<?php
echo "Hello World";
?>
```

The "Hello World" message is displayed when the code runs in a browser.

1. Save the file in your Apache document root directory. Only files in this directory are serviced by the Web server. For information about the document root directory, see Deploying Code with Zend Server.
2. Open a browser and enter the following URL: http://localhost:<port number>/hello.php.
   Replace <port number> with the port you are using. The default values are port 80 for Windows DEB and RPM and port 10088 for the other operating systems unless you manually changed the port assignment.

Your browser displays the "Hello World" message.

Configure Debugger Access Control

Purpose: To enable PHP debugging using Zend Studio and Zend Server Community Edition.

Result: You are able to debug your PHP code and view the results in Zend Studio.

Before working with the Debugger, configure the allowed hosts in **Server Setup | Debugger.**

Note:

By default, Zend Server Community Edition comes with a permissive setting that allows all standard private IP addresses (for example 10.*.*.*) to access the Debugger. For security reasons, if you do not have an immediate need for permissive access, remove these ranges from the Allowed Hosts: 10.*.*.* / 192.168.*.* / 172.16.*.*.

Additional setup information can be found in the Installation Guide, in Package Setup and Control Scripts.

Configuring Zend Server Community Edition

This section refers to the actual configuration workflow for using Zend Server Community Edition. Here, we describe the general workflow. Each component also has a separate section describing how to work with the component in detail.

The Zend Server Community Edition's Administration Interface is the main control center for configuring your PHP and Zend Server Community Edition components. After installing Zend Server Community Edition, use the Administration Interface to configure your PHP by performing the following actions:

1. In **Server Setup | Extensions**, define the extensions that should be "turned on" or "turned off". If you are planning to use functions related to an extension in your code, verify that the extension is turned on. If your extension has additional directives that are used to configure the extension's behavior, a configure link is included in the Directives column. Clicking this link leads you to the directives, pre-sorted to display the relevant directives.
2. The Directives page is accessed by clicking **Server Setup | Directives**. Here, you find all the directives relating to the extensions and components loaded in your PHP. If you cannot find a directive in the directives page, look in **Server Setup | Extensions** or **Server Setup |** Components to check that the extension or component is "turned on". See Adding Extensions for instructions on how to manually add an extension.
3. In **Server Setup |** Components, define the Zend Server Community Edition components that should be "turned on" or "turned off". If you are planning to use functions related to Zend Server Community Edition components in your code (such as the Optimizer+, Data Cache, Debugger, Guard Loader or Java Bridge), verify that the extensions are "turned on". If your Zend Server Community Edition component has additional directives used for configuring the component's behavior, a configure link is included in the Directives column. Clicking this link leads you to the relevant directive in the Directives page .
4. In **Server Setup | Debugger,** define which hosts are allowed to connect to the server to use the Zend Debugger for debugging and which hosts are not allowed.

Restart PHP Message

The Restart PHP message appears whenever a change is made to settings in your php.ini file. in order to apply the settings click the "Restart PHP" button. The changes will be applied to php.ini file on which Zend Server is running.

Working with Extensions

The Extensions page provides a convenient way to view and configure PHP extensions.
Use this page to control and configure the extensions that are loaded in your environment.

Changing Extension Status

To change an extension's status:

1. Go to **Server Setup | Extensions**.
2. Select an extension. In the actions column, click Turn off or Turn on:

   - Built-in extensions do not have the Turn on or Turn off option.
   - After changing an extension's status, a message appears to prompt you to click the Restart Server button at the bottom of the screen 
   - You can turn more than one extension on (or off) before you click Restart Server . All the changes that are made prior to restarting the server are applied after the restart.
   - If you navigate to other tabs, the changes you make are saved and applied when the server is restarted.

Changes are updated in the Server Info page and in your php.ini file. Changes are also applied when the server is manually restarted.

Restart PHP Message

The Restart PHP message appears whenever a change is made to settings in your php.ini file. in order to apply the settings click the "Restart PHP" button. The changes will be applied to php.ini file on which Zend Server is running.

Configuring Directives Associated with Extensions

To configure a directive associated with an extension:

1. Go to **Server Setup | Extensions**.
2. If the Extension has directives that can be configured, a link appears in the directives column.
   Clicking the link opens the Directives page, with the relevant directives already filtered.
3. Configure the directive as required.
   You can configure multiple directives before you save and apply your changes.
4. Click the Save Changes 

button at the top right corner of the screen to save your changes. To discard changes, navigate away from the screen without clicking the Save Changes button.

Changes are updated in the Extension Configuration screen and in the php.ini file the next time the server is restarted.

Note:

Directives of extensions that are turned off can also be configured through the Extensions page. Added extensions that are not part of the original Zend Server Community Edition list of extensions cannot be configured on the Extensions page.

Working with Logs

The Logs page is a log viewer for developers to view log information directly from the Administration Interface.
From this page you can:

- View a Log
- Filter log information
- Navigate inside a log
- Activate 'Auto Refresh'
- Advanced
- Log File Permissions

Advanced users can also add logs to the list of logs to display in the "Log View" list.

View a Log

This procedure describes how to view a log file.

To view a log file:

1. Go to Monitor | Logs.
2. Select a log from the View Log list.
3. The log information is displayed in the main display area.

Use the Show option 

(located below the main display) to determine how many lines to display. To use this option, enter a number between 5 and 200 and click **Go** to apply the setting.

Filter Log Information

This procedure describes how to filter a log file to fine tune the information to display specific results.

To filter a log file:

1. Select a log to display.
2. Go to the Filter area and enter the text to use for the filter: You can use any text.
3. Click **Refresh** or **Find**.

   The results are displayed in the main display area.

To run another query, change the text in the Filter area and click **Refresh**. There is no need to display the complete log again.

Navigate Inside a Log

This procedure describes the different navigation options available for navigating inside a selected log file.

Start - displays the first X lines of the log file.
Prev - shows the previous X lines of the log file.
Next - Shows the Next X lines of the log file.
End - displays the last X lines of the log file

'X' represents the number of lines that you specified in the Show option 

. The default value is 20.

Activate 'Auto refresh'

The following procedure describes how to activate and deactivate the Auto refresh option. The Auto refresh option sets the log information to display the most recent log entries in the last lines of the log that is currently being viewed. Therefore, as the log changes over time, the content in the view is always current. This feature provides an easy way to view errors in "almost real-time". (Because the refresh rate is in seconds, there is at least a 3-5 second display lag, which is why the Auto refresh feature is not considered true real-time logging.)

To activate Auto refresh:

1. Select a log to display.
2. Click the Auto refresh check box to automatically refresh the log information.

As long as the log is displayed, the information is refreshed. Each time you choose another log or exit the page, the settings are reset.

Advanced - Add logs to the list of logs in the "Log View" list.

It is possible to add and display other logs that are specific to your environment in the Log Tail page.

To add these other logs requires that you view and access backend application files which, in normal circumstances, should not be changed. For this reason, we request that you perform this task only if you clearly understand the instructions. If for some reason the system does not load or malfunctions, please re-install Zend Server Community Edition .

Power users may edit the XML file in <install_path>/gui/application/data/logfiles.xml to add as many logs as they may have.

To add log files to the list:

1. Open the file <install_path>/gui/application/data/logfiles.xml.
2. Add the name and location (full path) of the log files in the same format as the existing files and save.
3. Restart your PHP.

Working with Components

The Components page provides a convenient way to view and configure the Zend Components installed in your environment.
Use this page to control and configure components loaded in your environment.

Changing Component Status

To change a component's status:

1. Go to **Server Setup | Components**.
2. Select a component and click the link in the Actions column to turn the component on or off.
3. After changing the component's status, a message appears, prompting you to click the Restart Server button at the bottom of the screen 

.

   - More than one component can be loaded or unloaded before you click Restart Server . All the changes made prior to restarting the PHP are applied when the server restarts.
   - Even if you navigate to other tabs, the changes are kept and are applied when the server restarts.

Changes are updated in the Components page and in your php.ini file. Changes are also applied when you manually restart your Web Server.

Configuring Directives Associated with Components

To configure a directive associated with a component:

1. Go to **Server Setup | Components**.
2. If the component has directives that can be configured, a link appears in the directives column.
   Clicking the link opens the Directives page with the relevant directives already filtered.
3. Configure the directive as required.
   You can configure multiple directives before you save your changes.
4. Click the Save Changes 

button to save your changes. To discard changes, leave the screen without clicking Save Changes.

Changes will be updated in the Components page and in your php.ini file the next time the server restarts.

Note:

Directives of both loaded and unloaded components can be configured through the Components page.

Actions

Actions are additional activities that can be applied to a certain component when necessary.

The actions are as follows:

-  Clear - Clears all cached information (Data Caching and Optimizer+ bytecode caching).
- **Manage** - Directs the user to an additional page inside the Administration Interface to manage and fine-tune a component. The basic definitions that are defined by directives are set by clicking Configure.
-  Restart - Server-based components can be restarted using this action (for example the Java Bridge).

Adding New Components

The installation process determines which components are installed in your environment. Depending on your operating system, you can choose to customize your installation (Windows) or to work with a basic set of components that you can add to later on (DEB, RPM).
We provide all Zend components with loader binary when ZAMP is installed, however in examples like php.ini its entry is commented upon and therefore is not loaded.
In this case no additional installation is required but only configuration change.

For installation specific instructions on how to add additional components, see Choosing Which Distribution to Install and click on your installation type for instructions.

Working with Directives

This tab is accessed from **Server Setup| Directives**

The initial display shows the most commonly used directives. Click "**All**" for the full list of directives or use the "**Search**" component to locate a specific directive.

Users are also directed to this page from the Extensions and Components pages when they click "Configure" for an extension or a component that has directives which can be configured.

To configure directives:

1. Expand one of the lists, use the Search/All or the popular options to locate the relevant directive.
2. Configure the directive as required.
   You can configure multiple directives before saving.
3. Click the Save Changes  button at the top right corner of the screen to save all the changes made or leave the page without saving to discard the changes
4. As soon as changes are made to this page, a prompt to Restart Server is displayed.
5. Click  .

The changes are updated in the Directives page and in your php.ini file.

Working with Optimizer+

The Optimizer+ runs out-of-the-box (by default, after installation). Optimizer+ allows you to gain a performance boost by reducing code compilation time. When PHP code is compiled for the first time, it is saved in the server's memory. Each time the code is called, the pre-compiled version is used instead of waiting for the code to compile, which causes a delay each time the code is used.

Note:

Using the Optimizer+ should not be confused with caching. The Optimizer+ saves a compiled script to the server 's memory, while Caching saves the script's output to the server's memory.

The general recommendation is to always keep the Optimizer+ set to 'On' to boost Web application performance.

If your PHP application is business-critical, you may wish be alerted to any performance slowdowns. Zend Server can monitor your application in production, alert you to performance issues or errors, and provide you with diagnostic information for rapid root cause determination.
This feature is available only in the commercial version of Zend Server.

When Not to use Optimizer+ (Blacklist)?

There are some instances where it is preferable not to store PHP byte-code for certain PHP files. To do so, you can make a list (a blacklist) of file names that you want the Optimizer+ to ignore or increase the Optimizer+ resource allocation.

Files and directives should be blacklisted under the following conditions:

- Directories that contain files that are larger than the allocated memory defined in: *zend_optimizerplus.memory_consumption* or or that contain more files than the allocated quantity of files, as defined in *zend_optimizerplus.max_accelerated_files*.
- Large files that have high memory consumption - If you have exhausted all your allocated memory, select the largest and slowest scripts blacklist them.
- Files that have long execution times (makes the compilation save irrelevant).
- Code that is modified on the fly (e.g., auto-generated template files).

Increasing Optimizer+ Resource Allocation

The following procedure describes how to change Optimizer+ resource allocation. This procedure is used as an alternative to blacklisting files and should be tried first, before adding a file to a blacklist (unless the file meets one of the criteria above). Optimizer+ settings can be changed to increase allocated memory and the maximum quantity of files. This alternative depends on the amount of memory available to allocate to the Accelerator.

Memory allocation can only be increased when the Optimizer+ is set to 'On'.

To increase the Optimizer+ memory allocation:

1. Go to **Server Setup | Components** and verify that the "Zend Optimizer+" component is set to 'On'.
2. Click the "Configure" link in the directives column to display the list of Optimizer+ directives.
3. Locate the directive: *zend_optimizerplus.memory_consumption* and increase the value according to your system's memory allocation abilities.

To increase the quantity of files:

1. Go to **Server Setup | Components** and verify that the "Zend Optimizer+" component is set to 'On'.
2. Click the "Configure" link in the directives column to display the list of Optimizer+ directives.
3. Locate the directive: *zend_optimizerplus.max_accelerated_files* and increase the value according to your system's memory allocation abilities.

If the memory fills up quickly (especially if there are only a few files), increase the memory allocation or blacklist the file. Files which exceed the allocated memory or file quantity are not accelerated.

Blacklisting Files

If none of the alternatives (described above) are suitable, or if the file meets one of the criteria for blacklisting a file, use the following procedure to create a blacklist file that contains the file names of the files you do not want to be byte-code cached by Optimizer+.

To create a blacklist file:

1. Create a .txt file using a text editor.
2. Write a list of the file names to blacklist (i.e., ignored by the Optimizer+). List each file name in a new line.
3. In **Server Setup | Components,** verify that the "Zend Optimizer+" component is set to 'On'.
4. Click the "Configure" link in the directives column to display the list of Optimizer+ directives.
5. Locate the directive: *zend_optimizerplus.blacklist_filename* and specify the full path to the file location.

The files in the blacklist are now ignored by Optimizer+.

Optimizer+ Duplicate Functions Fix

In situations where certain functions were (or were not) defined, some PHP code produces different opcodes, depending on the circumstances. This causes a discrepancy for the Optimizer+ in the situation where the Optimizer+ caches one version, and a sequence of events arises that requires a different function. If the discrepancy is not addressed, the script stops working and raises a "duplicate functions" error.

To maintain proper performance in these and similar situations, activate the *zend_optimizerplus.dups_fix* parameter. This parameter shuts down the Optimizer+ duplicate function check to prevent these errors from occurring.

This parameter can be defined in **Server Setup | Directives** by searching for *zend_optimizerplus.dups_fix*.

Working with Zend Guard Loader

The Zend Guard Loader is a PHP extension that is used to run code that was encoded or obfuscated using Zend Guard. If you chose to install this component, it is set to run by default, out-of-the-box.

To locate your installation package and verify if the component was installed by default or needs to be installed, see the Installation Guide, Choosing Which Distribution to Install.

PHP code that was either encoded or obfuscated using the Zend Guard, or which is license restricted will only work if the

Zend Guard Loader component is set to 'On'.

The Zend Guard Loader component can be set to 'On' or 'Off'' from Server Setup | Components**.**

Note:

If you do not require the Zend Guard component for optimal performance, either do not install it, or set this component to 'Off'.

Working with Java Bridge

The Java Bridge is only active when the Java Bridge component is installed and activated (see the Installation Guide). The component's status and settings can be viewed and configured in the Administration Interface, from **Server Setup | Components**.

Note:

The Java Bridge requires that you have Sun Microsystems JRE 1.4 (or later) or IBM Java 1.4.2 (or later) installed on your computer. During or after installing (depending on the installation type), you are prompted to direct the installer to the JRE location. Therefore, you should already have JRE installed. 64-bit JRE is not supported.

More information about JREs and the latest updates can be obtained from the SUN Microsystems Website.

Configuration

This procedure describes how to configure the target Java runtime environment.

Configuring the runtime environment:

Use the following command to run JavaMW:

```
java com.zend.javamw.JavaServer
```

For correct execution, the classpath should include the javamw.jar file in the directory where JavaMW is installed.

Example:

```
and Mac <install_dir>/bin/javamw.jar
<install_dir>\bin\javamw.jar
```

Testing the Bridge Connection

The following code sample shows how you can, as an initial step, test the connection between your PHP and Java environments to ensure that the Java Bridge is defined properly and communicates with the correct Java. This code demonstrates the interaction between a PHP application and Java objects that occurs in the Java Bridge implementation.

To test the Java Bridge connection:

Create a new PHP script to create a Java object, as in the example below:

```
<?php
// create Java object
  $formatter = new Java("java.text.SimpleDateFormat",
                        "EEEE, MMMM dd, yyyy 'at' h:mm:ss a
zzzz");
  // Print date through the object
  print $formatter->format(new Java("java.util.Date"))."\n";
  // You can also access Java system classes
  $system = new Java("java.lang.System");
  print $system."\n"; // will use toString in PHP5
  print "Java version=".$system->getProperty("java.version")."
<br>\n";
  print "Java vendor=".$system->getProperty("java.vendor")."
<p>\n\n";
  print "OS=".$system->getProperty("os.name")." ".
              $system->getProperty("os.version")." on ".
              $system->getProperty("os.arch")." <br>\n"; ?>
```

If the Java Bridge is correctly installed and running, you should receive the following response:

```
Friday, June 13, 2008 at 8:45:57 PM U.S Daylight Time class
java.lang.System Java version=1.6.0_06 Java vendor=Sun
Microsystems Inc.
OS=Linux 2.6.25.3-18.fc9.i686 on i386
```

This output shows the date, Java version, vendor and operating system and indicates that the connection is complete.

If you receive an error message instead of the expected output information, one of the following problems may have occurred:

1. The Java Bridge is not installed
2. The Java Bridge extension is not running (**Server Setup | Components**)

3. The Java Bridge Server needs to be restarted (**Server Setup | Components**)
4. The requested .jar file does not appear in the environment's classpath.

Once the connection is established, you can start using the API to call Java objects from your PHP.

Before using the Java Bridge API

Before you start incorporating the Java Bridge API in your code, you must be aware that when you call Java from PHP, you must use Java coding standards to call the correct objects, because the Java Bridge does not perform dynamic data conversion. You must perform the type conversion in your PHP code.

For example,

Example:

If you call a Java method that looks like this:

```
public void doSomething(int i);
```

Using what you would expect to work in PHP:

```
$var = "1"
$javaObject->doSomething($var);
```

The Java Bridge throws an exception. To avoid this, use the following line of code to convert the parameter from a string to a numeric value before the Java Bridge passes it:

```
$javaObject->doSomething($var + 0);
```

For more information, see the API, or Java Bridge Use Cases.

Debugger

Working with Local Debugging

Local debugging occurs when your entire environment (Zend Studio for Eclipse, Debugger and Zend Server Community Edition) is located on a single machine.

When working with an IDE such as Zend Studio for Eclipse, your project files are, in most cases, placed in a location that you have defined. To run the files on the Web Server, you must first move the files to the Web Server's document management directory called "htdocs".

Working with the Debugger

The Debugger API that is included in Zend Server Community Edition is a remote debugging tool for developers who work with Zend Studio. If the Debugger Component is not set to "On" in the Components page, you are not able to run remote debug sessions using Zend Studio. For more information on turning the Debugger Component to "On", see Working with Components.
From the Zend Server Community Edition perspective, other than defining allowed hosts and denied hosts, no additional interaction is required.

The following procedure describes how to define allowed hosts for debugging. Users define allowed hosts to create a list of IP addresses (of computers that run Zend Studio) that have permission to debug the PHP code that runs on the server.

To define allowed hosts for debugging:

1. In the Administration Interface go to **Server Setup | Debugger**.
2. In the "Allowed Zend Studio Clients for Debugging" section, enter a valid IP address or enter a range by entering the beginning of an IP address and adding '0' instead of the rest of the number. To make sure you are using Wildcards (*) to specify a range of IPs select the pattern you want from the drop-down list.
3. From the drop-down list, select an option according to the type of IP address you entered. Click 'Exact IP address only' for a single IP, or one of the other options to represent a range of hosts.
4. Click 

to add the Host.
5. The changes are applied after you restart the Server 

The IP or range of IPs is allowed to connect to the server to debug PHP code with Zend Studio .
To remove a specific IP from the list, click "Remove".
Important Note:
If your machine has several IP addresses (for example if you are using a wireless network connection on a laptop) verify that you have defined all the possible IP addresses as "Allowed Hosts for Debugging" or that the IP you want to use is first in the list of IPs in Zend Studio for Eclipse. (In **Window | Preferences | PHP | Debug | Installed Debuggers,** verify that Zend Debugger is selected and click **Configure** in the Client Host/IP field.)

The following procedure describes how to define denied hosts for debugging. Users define denied hosts to create a list of IP addresses (of computers that run Zend Studio) that do **not** have permission to debug the PHP code that runs on this server.

To define denied hosts for debugging:

1. In the Administration Interface go to **Server Setup | Debugger**.
2. In the "Denied Zend Studio Clients for Debugging" section, enter a valid IP address or use Wildcards (*) to specify a range of IPs.
3. From the drop-down list, select an option according to the type of IP address you entered. Click 'Exact IP address only' for a single IP, or one of the other options to represent a range of hosts.
4. Click 

to add the host.
5. The changes are applied after you restart the Server 

.

The IP or range of IPs is denied permission to connect to the server to debug PHP code with Zend Studio.

To remove a specific IP from the list, click "Remove".

Note:

Do not add the same IP address to both the Allowed and Denied host lists. Pay attention when you specify a range of IP addresses: If you deny a range of addresses that includes an IP that was specified in the Allowed hosts, the host is not allowed to create a debug session.

Wildcards (Net Mask)

Wildcards use the asterisk (*) to define a string of IP addresses and to specify a range of IPs that are either allowed or denied hosts. This option makes it possible to specify a range of IPs from 0-255, according to the selected number of wildcards. For example, if you use the Net Mask option to deny the IPs 10.1.3. *, all the IP addresses beginning with 10.1.3. are denied access to the Studio Server (i.e., integration with Studio is not permitted for these IP addresses).

Remote Debugging Through a Firewall?

Remote debugging is the process of creating a connection between two machines: For example, the machine on which the Debugger (Zend Studio) resides and the machine on which the Zend Server Community Edition resides. When these machines are on the same local network or there are no security devices that limit remote connections, no additional action is required. However, if one or both of the machines are behind a firewall, the communication required to run the debug process is not allowed. To allow debugging and still maintain a secure environment, you need to use firewall tunneling. For more information on how to setup firewall tunneling, see Working with Firewall Tunneling.

Working with Zend Controller

Initial Setup

The following procedure describes how to configure the Zend Controller's settings to communicate with Zend Server Community Edition. This procedure should be completed before using the Zend Controller.

To Set up the Zend Controller:

1. Open the Zend Controller menu (right-click in Windows or Unix, Ctrl-Click in Mac).
2. In the Zend Controller's menu, click to open the Settings dialog.
3. Make sure the following settings are correct:

- **Hostname** - unique name or IP number of the server on which Zend Server Community Edition is running. Can be a remote server on the same LAN.
- **Port** - The default ports are:
  - Windows: 80 for HTTP
  - Unix: 10081 for HTTP and 10082 for HTTPS
  If you changed the port of the Web server that runs Zend Server Community Edition during the installation, change this value too.
- **Password** - The password is automatically configured when you set your Administration Interface password.
- **Connection Scheme** - Your preferred method of connecting the Control Panel with Zend Server Community Edition for communication purposes, where HTTPS is a secured connection protocol.

Once the Zend Controller is properly configured, you can use it to change the status of the following components; Data Cache, Debugger, Optimizer+ and Java Bridge. You can also access the Administration Interface directly by clicking one of the following Zend Controller buttons: Configure Zend Debugger, Zend Extension Configuration and PHP Info.
Other Zend Controller features include Multi-Source search and Benchmarking.

Using the Zend Controller Benchmark Tool

The Zend Controller Benchmark tool is a simple benchmark that developers can use to run performance tests on the URLs (Web pages) they develop. The main purpose of this tool is to identify the performance gain that is achieved when using Zend Server Community Edition's Optimizer+ and Data Caching components. This can be done by turning the different Zend Server Community Edition components on and off and running the benchmark.
The Zend Controller Benchmark tool does not replace standard benchmarking utilities. It is intended to provide a quick and easy way to measure performance without having to run elaborate and resource-expensive performance tests.

How it Works

The Benchmark tool checks HTTP request response times and lists them in a bar chart that displays when the test was started and the average amount of 'requests per second' received for the duration of the test (user defined, in seconds). These tests can be run once, without one of the performance-related components (Data Cache and Optimizer+), and then again (with each or all components turned on) to see the effect each component has on performance.
Before running a test, make sure the URL you enter is the exact URL and does not rely on redirection: Using a redirecting URL causes the test to fail.

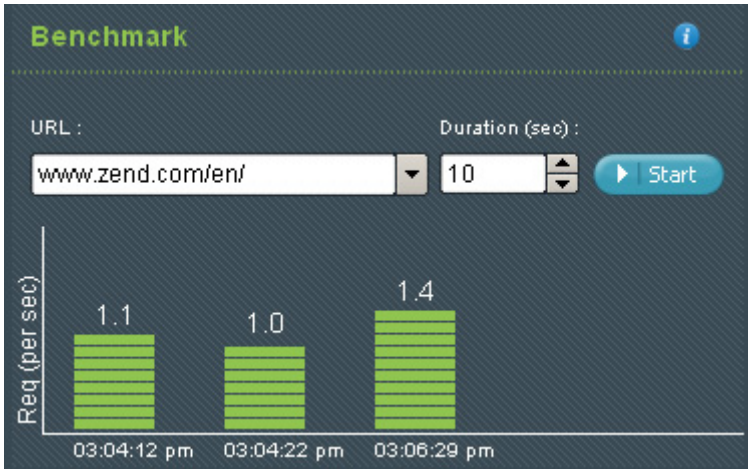
To run a Benchmark:

1. Open the Zend Controller
2. In the Benchmark section, enter a URL.
3. In the Duration section, define the amount of seconds to run the test.
   If you are comparing how different Zend Server Community Edition components affect performance, make sure you run the tests at approximately the same time, to avoid large fluctuations in traffic volume and ensure that the traffic conditions are similar for each test.
4. Click **Go** to start running the test.
   Clicking **Abort** terminates the test without collecting test information.

The results are displayed in a bar chart. The Benchmark tool displays up to five test results. If there are more than five results, the tool displays the five most recent results.

Understanding Results

Once you have the results, the most important consideration is to determine what constitutes a good value.



When testing the effect Zend Server Community Edition components have on performance, the more requests per second, the faster the code.

Another consideration is the size of the page: Large pages take longer to load and should be checked during both high and low traffic to determine if the page is performing well.

Cache

Working with Data Cache

The Data Cache API is used the same way as any other API: By inserting the API functions into your PHP code. The Data Cache component uses an API to cache partial PHP outputs using memory or disk.

You can further enhance the performance of your application by caching Web pages that don't require frequent change.
This feature is available only in the commercial version of Zend Server.

The Data Cache API includes the following functionality:

- Storing variables to the Cache
- Fetching variables from the Cache
- Deleting variables from the Cache
- Clearing the Cache
- Disk/memory (SHM) storage
- Caching using namespaces
- Cache folder depth configuration

Disk/Shared-Memory Caching

This feature provides options to determine where to store cached variables. Memory caching improves server responsiveness and increases performance - primarily in environments that run high-traffic applications that can benefit from off loading activity directed toward their hard disk. Disk caching is more suitable for smaller applications and ensures the cached content is available after the machine is restarted.

SHM/disk storage is implemented by using the appropriate API functions and configuring the Data Cache directives.

Note:

Memory option error messages have been created to notify you if the store operation fails or you run out of allocated memory.

The following example shows the different storage options:

Example:

```
A simple key with no namespace stored on disk
if (zend_disk_cache_store("hello1", 1) === false){
    echo "error2\n";    exit();
}
Shared memory:
if (zend_shm_cache_store("hello1", 1) === false){
    echo "error2\n";    exit();
}
Store with namespace on disk
if (zend_disk_cache_store("ns1::hello1", 1) === false){
    echo "error2\n";    exit();
}
Shared memory:
if (zend_shm_cache_store("ns1::hello1", 1) === false){
    echo "error2\n";    exit();
}

Store with namespace on disk with limited lifetime (3)
if (zend_disk_cache_store("ns3::test_ttl", 2, 3) === false){
    echo "error12\n";    exit();
}
Shared memory:
if (zend_shm_cache_store("ns3::test_ttl", 2, 3) === false){
    echo "error12\n";    exit();
}
```

'namespace' Support

Using namespaces for caching provides the ability to define a key that can serve as an identifier to delete select items from the cache, rather than unnecessarily removing shared instances. '*namespace'* support is intended for environments that run large applications that are separated into modules. Applying a '*namespace*' to each module provides the identification necessary to pinpoint all the cached items that belong to a given module and remove only those specific items.

This does not mean that you must use the '*namespaces*' to clear the cache: The entire cache can be cleared by using the '*output_cache_remove*' function.

Setting the cached 'namespace':

The cache *'namespace'* is set by adding it as a prefix to the cache with '::' as the separator.

Example:

This example shows how to manipulate variable caching using a *'namespace'*
zend_disk_cache_store("my_namespace::my_key",$data) is fetched with
zend_disk_cache_fetch("my_namespace::my_key");
zend_shm_cache_clear("my_namespace") clears all the keys that start with
"*my_namespace::*"

Cache Folder Depth Configuration

Defining the Cache folder depth is intended for environments that use a large number of keys. By definition, cached content is separated into different directories by key, to prevent performance degradation caused by accessing files that contain large amounts of content. This option is only available with disk caching. Increase the cache folder depth according to the quantity of content that requires caching (small amount = 0, large quantities = 2).

Note:

A single directory may include several keys, depending on the quantity of cached content.

The cache folder depth is defined by the directive zend_cache.disk.dir_levels. The value of the directive configures how the cached files are stored. The accepted values for this directive are 0, 1 or 2, where:

0 = one directory containing all the cache files

1 = a separate directory under the cache directory

2 = an additional sub directory for cached content under the cache directory

phpMyAdmin

Working with phpMyAdmin to Manage MySQL

phpMyAdmin is a tool written in PHP which is intended to handle the administration of MySQL over the Web. Currently, it can create and drop databases, create/drop/alter tables, delete/edit/add fields, execute any SQL statement, manage keys on fields, manage privileges, export data into various formats and is available in 55 languages.

The Zend Server Community Edition Installer includes this component as part of the installation process in Windows and Zend Server Community Edition. Download the Linux and Mac version from http://www.phpmyadmin.net: They are available as RPM and DEB packages from your distribution's repository. See the Installation Guide for additional operating system and Installer-specific information.

The following types of Installations are available:

- Linux
- Mac OS X
- Windows

Working with MySQL Server: Linux

This procedure is relevant for users who manually downloaded and installed phpMyAdmin.

This procedure describes how Unix users with root privileges can use the phpMyAdmin tool to set up their environment to work with a MySQL server.

Before following these instructions, verify that your MySQL server is installed and running. If you do not have an Internet connection, make sure you have access to the phpAyAdmin installation package.

To extract and install phpMyAdmin:

1. Download the package from http://www.phpmyadmin.net.
2. Extract the package with the command *tar -xzvf phpMyAdmin-2.11.7-all-languages-utf-8-only.tar.gz .*
3. Move the extracted directory to <install_path>/zend/gui/lighttpd/htdocs/phpMyAdmin with the following command:
   mv <extracted dir> <install_path>/zend/gui/lighttpd/htdocs/phpMyAdmin .
4. Change your directory using the following command: *cd <install_path>/zend/gui/lighttpd/htdocs/phpMyAdmin/*
5. Create a directory called config under the phpMyAdmin directory with the following command: *mkdir config .*
6. Open the phpMyAdmin Web Interface by following the link: https://localhost:10082/phpMyAdmin/scripts/setup.php .
   If you are using a different port or connecting from a remote server, replace the port number <10082> with the appropriate port number or replace <localhost> with the IP address of the remote computer.
7. Once the phpMyAdmin setup page is open, you can start configuring it to manage your MySQL Server.

To configure phpMyAdmin to work with an existing MySQL server:

1. In the phpMyAdmin setup page, click **Add** to add a MySQL server.
2. In the Add section, configure the following parameters:
   - **Server Host Name**: localhost for local servers. If you are not using a local server, enter your machine's IP address.
   - **Port socket path**.
   Most users will not have to change any settings.
3. In the Authentication Type drop-down, change the type to **http**.

4. Click **Add** to add the new server and fold the display.
   A message stating that a new server was added is displayed.
5. Go to Configuration and click **Save** to create a configuration file.
6. Take the configuration file and move it to <Missing>.
   Your server has now been added and can be configured with phpMyAdmin.

Further information on using phpMyAdmin can be found in the online documentation at: https://localhost:10082/phpMyAdmin/Documentation.html.

Note:

To log in to your phpMyAdmin server, you must use your existing MySQL server user name and password (usually "root" for administrators).

Working with MySQL Server: Mac OS X

The Zend Server Community Edition Mac package includes MySQL and phpMyAdmin. This enables the files to be installed seamlessly and to ensure a smooth configuration process.

File Locations

- mySQL binaries (such as 'mysql') reside in:
  <install_path>/mysql/bin/
- mySQL tables and database reside in:
  <install_path>/mysql/data/
- Configuration files, in particular, my.cnf reside at:
  <install_path>/mysql/data/

Default Port and Socket

Since, by default the 'Skip-networking' option is enabled, the MySQL server does not listen on a TCP/IP port at all; All interactions with 'mysqld' must be made via Unix sockets. The socket file resides at <install_path>/mysql/tmp/mysql.sock.

Starting and Stopping

Generally, zendctl.sh is used to start and stopZend Server Community Edition modules. To start and stop the MySQL server use:
<install_path>/bin/zendctl.sh stop-mysql
<install_path>/bin/zendctl.sh start-mysql

Password

Default user is: zend, and password is left blank

Change the password, either at the config file 'my.cnf', or using the phpMyAdmin interface. To access the phpMyAdmin interface go to the Dashboard and follow the 'Open phpMyAdmin' link.

phpMyAdmin Note:

phpMyAdmin access is by default allowed only from the localhost. To open phpMyAdmin interface to remote user comment out the following lines from <install_path>/gui/lighttpd/etc/lighttpd.conf:

```
138 #    $HTTP["remoteip"] !~ "127.0.0.1" {
139 #        $HTTP["url"] =~ "^/phpmyadmin/" {
140 #            url.access-deny = ( "" )
141 #            server.errorfile-prefix = "/usr/local/zend/gui/lighttpd/share/lighttpd-custom-errors/errorcode-"
142 #        }
143 #    }
```

Working with MySQL Server: Windows

If you already have phpMyAdmin

When you install Zend Server Community Edition , you can use the custom installation type and choose not to install phpMyAdmin.
If you decide to install phpMyAdmin, a separate version is installed and the existing phpMyAdmin configurations are retained. The default location is <install_dir>\phpMyAdmin. The default authentication is user: root; and without a password.
A link to this phpMyAdmin installation is added in the Zend Server Community Edition dashboard.

If you already have MySQL

If you have a local installation of MySQL, it will be automatically detected during the installation process.
If you want to set phpMyAdmin to a remote MySQL server (running on a separate machine), see the PHPMyAdmin online documentation.

Apache Note:
When running phpMyAdmin on Apache, the URL is case sensitive.

If you don't have anything (phpMyAdmin or MySQL)

When you install Zend Server Community Edition , you can use the full or custom installation types to choose to install phpMyAdmin and MySQL.
Both phpMyAdmin and MySQL are installed on your local machine under the default location <install_dir>\phpMyAdmin
and <install_dir>\MySQL.
A link to this phpMyAdmin installation is added in the Zend Server Community Edition Dashboard.

Reference Information

This section contains reference information for PHP developers. Here you will find information about using the Java Bridge, the extensions included in this release and other system-related information.

The list of extensions provides an overview of all the extensions that are included and their status (On, Off, Disabled). A description of what each status means can be found in the PHP Extension List.

Components

Zend Server Community Edition is comprised of several components that each contribute important functionality to facilitate the development process.

The components are:

- Debugger - The Zend Debugger communicates with the Zend (PHP) Engine to retrieve runtime information and present it in Zend Studio for root cause analysis.
- Optimizer+ - The Zend Optimizer+ component speeds up PHP execution via opcode caching and optimization.
- Guard Loader - The Zend Guard Loader is used in order to run PHP scripts that are encoded with Zend Guard.
- Data Cache - The Zend Data Cache component provides a set of PHP functions to improve performance, by storing data in the cache.
- Java Bridge - The Zend Java Bridge component makes it possible to use Java classes and code from within PHP.
- Zend Framework - An open source framework for developing Web applications and Web services with PHP.

Click on a link to view a full description of the components architecture. To see how to work with a component, select a topic that begins with "Working with..." from the Tasks section. For a short description of each component and where it is installed, see the Installed Components section in the Installation Guide.

Debugger

The Zend Debugger component enables remote debugging of PHP scripts with Zend Studio. The Zend Debugger communicates with the Zend (PHP) Engine to retrieve runtime information and present it in Zend Studio for root cause analysis purposes.

Note:

If your machine has multiple IP addresses, make sure you define all the IPs as allowed hosts in Zend Server Community Edition.

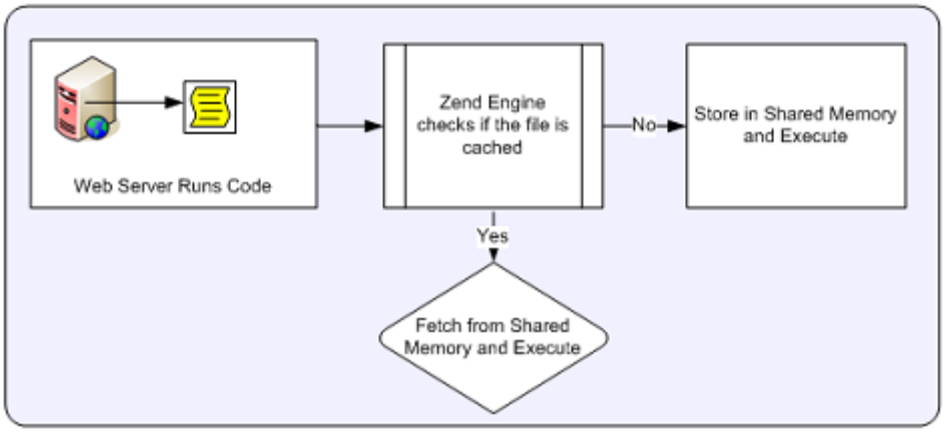
The Zend Debugger API communicates with the Zend (PHP) engine to reveal PHP runtime information such as variables, call stack and environment information. This information is then displayed and set up in Zend Studio to enable server side debugging, profiling and code coverage.

Optimizer+

The Zend Optimizer+ component speeds up PHP execution through opcode caching and optimization.

The Zend Optimizer+ improves PHP performance by storing precompiled script bytecode in the shared memory. This eliminates the stages of reading code from the disk and compiling it on future access. For further performance improvement, the stored bytecode is optimized for faster execution. This component works out-of-the-box and therefore does not require any configuration or changes to your code.

The Zend Optimizer+ speeds up PHP execution and increases server performance, resulting in better Web application performance.



This component is intended for PHP developers who run complex PHP applications and can benefit from bytecode caching (which is especially helpful for working with Zend Framework).

Note:

The Optimizer+ works exclusively with Apache or FastCGI environments (no CLI or CGI support).

Guard Loader

The Zend Guard Loader runs PHP scripts that are encoded with Zend Guard.

The Zend Guard Loader is a PHP extension that runs outputs created by Zend Guard, which provides an easy way to encode, obfuscate and license PHP code via an Eclipse-based interface or from the command line.

The Guard Loader extension must be installed on each Web server that runs files that were encoded with, or use, Zend Guard licenses.

Note:

You can also use the Zend Optimizer that also includes the Guard Loader extension for code written in PHP 5.2. The Zend Optimizer is available as a free download from www.zend.com.

The Zend Guard Loader translates encoded files to a format that can be parsed by the Zend Engine. This runtime process uses the Zend engine as a trigger to start the Zend Guard Loader component.

Zend Guard

Zend Guard is a separate product available from Zend that provides an easy way to encode, obfuscate and license PHP code via an Eclipse-based interface or from the command line.
To view the API, click Zend Guard Loader.
For additional information on using Zend Guard, see the Zend Guard User Guide, available online from http://files.zend.com/help/Zend-Guard/zend-guard.htm

Data Cache

The Zend Data Cache component provides a set of PHP functions to improve performance by storing data in the cache.

The Zend Data Cache is used to cache different types of data (e.g., strings, arrays and objects), as well as script output or script output elements for various durations. Items can be stored in shared memory (SHM) or to disk. Namespaces are supported, to group cached objects for easy management.

Data Caching is primarily used when it is impractical or impossible to cache the entire page output, such as when sections of the script are fully dynamic, or when the conditions for caching the script are too numerous. An example of this kind of usage is when some of the output is a form: The data may include credit card numbers, addresses and other kinds of information that should not be cached, for security reasons. For more information, see Working with the Data Cache.

The Data Cache API includes the following functionality :

- Storing variables to the cache
- Fetching variables to the cache
- Deleting variables from the cache
- Clearing the cache
- Disk/memory (SHM) storage
- Caching using namespaces
- Cache folder depth configuration

Java Bridge

The Zend Java Bridge provides PHP developers with a way to use existing Java code and build PHP applications that use Java code.
The Java Bridge integrates Java code in PHP by connecting the PHP object system with the Java Bridge object system.

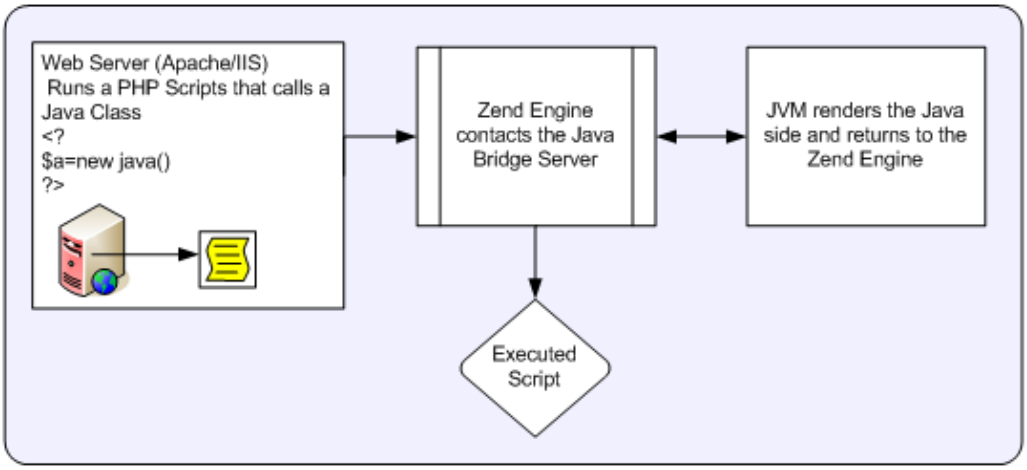
Note:

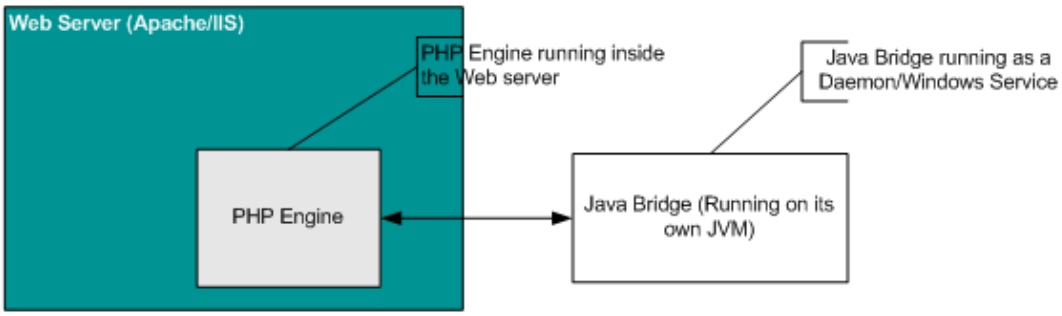
The Java Bridge requires that you have SUN Microsystems JRE 1.4 (or later) or IBM's Java 1.4.2 (or later) installed on your computer.
During (or after) installing, (depending on the installation type, you are prompted to direct the installer to the JRE location. You should, therefore, already have JRE installed. 64-bit JRE is not supported.
More information about JRE and the latest updates can be obtained from SUN Microsystems's website.

The Java Bridge PHP extension adds functions that allow you to instantiate new Java classes from inside your PHP script. Once a Java class is instantiated, the Java Bridge gets a message from the Zend Engine to execute the Java code. The Java Bridge executes the script and returns the results to the Zend Engine.

Zend Server Community Edition includes the Java Bridge PHP Extension and the ability to restart the Java Bridge and configure the Java Bridge settings (from **Server Setup | Components**).
The Java Bridge is an optional component that is installed differently, depending on the operating system (WIN, UNIX , MAC) and the installation method format (EXE, DEB, RPM , Tarball). Once the extension is installed and its status is On, PHP code can use the Java Bridge API to call Java objects.

The process of calling Java objects in PHP is described in the following diagram:





Advantages

The Zend Java Bridge provides the following advantages:

- J2EE application servers can be extended to include the advantages that PHP offers (relative to other Web-enablement languages), such as reduced development time, reduced time-to-market, lower TCO (Total Cost of Ownership), etc.
- PHP-centric companies can take advantage of J2EE services that are not present in scripting languages.
- The PHP/Java Bridge provides the ability to interact with plain Java objects.
- The Java Bridge operates without the overhead of a JVM for each Apache process.
- The Java Bridge consumes a set amount of memory that is disproportionately small relative to the amount of activity that it handles.

Zend Framework

Zend Framework is a high quality, open source framework for developing Web applications and Web services with PHP.
Built in the true PHP spirit, the Zend Framework delivers ease-of-use and powerful functionality. It provides solutions for building modern, robust and secure websites.

Zend Framework Resources

All the developer resources can be found at: http://framework.zend.com/

Why Zend Framework

(Taken from: http://framework.zend.com/whyzf/overview)
Extending the art and spirit of PHP, Zend Framework is based on simplicity: Object-oriented best practices, corporate friendly licensing and a rigorously tested agile code base. Zend Framework is focused on building more secure, reliable and modern Web 2.0 applications and Web services, and consuming widely available APIs from leading vendors like Google, Amazon, Yahoo!, and Flickr, as well as API providers and cataloguers like StrikeIron and ProgrammableWeb.

Expanding on these core themes, we have implemented Zend Framework to embody extreme simplicity and productivity, the latest Web 2.0 features, simple corporate-friendly licensing and an agile, well-tested code base that your enterprise can depend upon.

Extreme Simplicity & Productivity

We designed Zend Framework with simplicity in mind. To provide a lightweight, loosely-coupled component library simplified to provide 4/5s of the functionality everyone needs and that lets you customize the other 20% to meet your specific business needs. By focusing on the most commonly needed functionality, we retain the simplified spirit of PHP programming, while dramatically lowering the learning curve - and your training costs – so developers get up-to-speed quickly. We do this with:

Extensible and well-tested code base

Flexible architecture

No configuration files necessary to get going

Frameworks and best practices mean reduced training costs and quicker time-to-market – important factors in adoption decisions. Built so you can pick and choose just the pieces you need to turbocharge your web applications – all your developers know where to find their PHP / Zend Framework code, which speeds new development and reduces maintenance costs.

Latest Web Development Features

AJAX support through JSON – meet the ease-of-use requirements your users have come to expect
Search – a native PHP edition of the industry-standard Lucene search engine
Syndication – the data formats and easy access to them your Web 2.0 applications need
Web Services – Zend Framework aims to be the premier place to consume and publish web services
High-quality, object-oriented PHP 5 class library – attention to best practices like design patterns, unit testing and loose coupling

Friendly & Simple Licensing, Safe for the Enterprise

Based on the simple and safe new BSD license, with Zend Framework's License, you can rest assured that your code is compliant, unimpeachable and protected as you see fit. We also require all contributors to the open source Zend Framework to complete and sign a Contributor License Agreement (CLA) - which is based on the standard open-source Apache license — to protect your intellectual property (that is, your added-value) built on Zend Framework.

Fully Tested – Extend Safely and Easily

Thoroughly Tested. Enterprise-ready and built with agile methods, Zend Framework has been unit-tested from the start, with stringent code coverage requirements to ensure that all code contributed has not only been thoroughly unit-tested, but also remains stable and easy for you to extend, re-test with your extensions and further maintain.

Zend Controller

The Zend Controller runs parallel to the Administration Interface, to provide easy access to useful developer tools and information.
The Zend Controller is a small utility that you can use to remotely access the Administration Interface for tasks such as turning components on and off. The Zend Controller also provides developer resources, including the Benchmark Tool and a search area that lists sites targeted for PHP developer use.

API Reference

The API reference includes reference information for working with the API's. Each page includes a description of the component along with the functions for interacting with the component and the directives for configuring the component's behavior as follows:

- Zend Debugger Directives
- Zend Debugger API
- Zend Optimizer+ Directives
- Zend Optimizer+ API
- Zend Guard Loader Directives
- Zend Guard Loader API
- Zend Data Cache Directives
- Zend Data Cache API
- Zend Java Bridge Directives
- Zend Java Bridge API
- Zend Java Bridge Class
- Zend Extension Manager
- Zend Utils

Zend Debugger - Configuration Directives

Configuration Directives Summary

Directive	Type	Modification Scope	Description
zend_debugger.allow_hosts	string	PHP_INI_SYSTEM	Specifies the hosts that are allowed to connect (hostmask list) with Zend Debugger when running a remote debug session with Zend Studio
zend_debugger.deny_hosts	string	PHP_INI_SYSTEM	Specifies the hosts that are not allowed to connect (hostmask list) with the Zend Debugger when running a remote debug session with Zend Studio
zend_debugger.allow_tunnel	string	PHP_INI_SYSTEM	A list of hosts (hostmask list) that can use the machine on which Zend Server is installed to create a communication tunnel for remote debgging with Zend Studio. This is done to solve firewall connectivity limitations
zend_debugger.max_msg_size	integer	PHP_INI_SYSTEM	The maximum message size accepted by the Zend Debugger for protocol network messages
zend_debugger.httpd_uid	integer	PHP_INI_SYSTEM	The user ID of the httpd process that runs the Zend Debugger (only for tunneling)
zend_debugger.tunnel_min_port	integer	PHP_INI_SYSTEM	A range of ports that the communication tunnel can use. This defines the minimum value for the range
zend_debugger.tunnel_max_port	integer	PHP_INI_SYSTEM	A range of ports that the communication tunnel can use. This defines the maximum value for the range
zend_debugger.expose_remotely	integer	PHP_INI_SYSTEM	Define which clients know that the Zend Debugger is installed: 0 - Never. The presence of the Zend Debugger is not detected by other clients 1 - Always. All clients can detect the Zend Debugger 2 - Allowed Hosts. Only clients listed in zend_debugger.allow_hosts can detect the Zend Debugger Any other value makes the Zend Debugger undetectable (same as "Never")
zend_debugger.passive_mode_timeout	integer	PHP_INI_SYSTEM	The Debugger's timeout period (in seconds) to wait for a response from the client (Zend Studio)
zend_debugger.xdebug_compatible_coverage	boolean	PHP_INI_SYSTEM	Directive in order to mock up xdebug coverage

Directive	Type	Modification Scope	Description
zend_debugger.use_fast_timestamp	boolean	PHP_INI_ALL	Enables fast time sampling which is dependent on CPU cycles and frequency, otherwise, the directive uses operating system timing (which may be less accurate)

Configuration Directive Details

zend_debugger.allow_hosts

Specifies the hosts that are allowed to connect (hostmask list) with Zend Debugger when running a remote debug session with Zend Studio

Type: string

Default Value: 127.0.0.1/32,10.0.0.0/8,192.168.0.0/16,172.16.0.0/12

Available since version 3.6

zend_debugger.deny_hosts

Specifies the hosts that are not allowed to connect (hostmask list) with the Zend Debugger when running a remote debug session with Zend Studio

Type: string

Available since version 3.6

zend_debugger.allow_tunnel

A list of hosts (hostmask list) that can use the machine on which Zend Server is installed to create a communication tunnel for remote debgging with Zend Studio. This is done to solve firewall connectivity limitations

Type: string

Available since version 3.6

zend_debugger.max_msg_size

The maximum message size accepted by the Zend Debugger for protocol network messages

Type: integer

Default Value: 2097152

Available since version 3.6

zend_debugger.httpd_uid

The user ID of the httpd process that runs the Zend Debugger (only for tunneling)

Type: integer

Default Value: -1

Available since version 3.6

zend_debugger.tunnel_min_port

A range of ports that the communication tunnel can use. This defines the minimum value for the range

Type: integer

Default Value: 1024

Available since version 3.6

zend_debugger.tunnel_max_port

A range of ports that the communication tunnel can use. This defines the maximum value for the range

Type: integer

Default Value: 65535

Available since version 3.6

zend_debugger.expose_remotely

Define which clients know that the Zend Debugger is installed:
0 - Never. The presence of the Zend Debugger is not detected by other clients
1 - Always. All clients can detect the Zend Debugger
2 - Allowed Hosts. Only clients listed in zend_debugger.allow_hosts can detect the Zend Debugger.
Any other value makes the Zend Debugger undetectable (same as "Never")
Type: integer
Default Value: 2
Available since version 3.6

zend_debugger.passive_mode_timeout

The Debugger's timeout period (in seconds) to wait for a response from the client (Zend Studio)
Type: integer
Units: seconds
Default Value: 20
Available since version 3.6

zend_debugger.xdebug_compatible_coverage

Directive in order to mock up xdebug coverage
Type: boolean
Default Value: 0
Available since version 4.0

zend_debugger.use_fast_timestamp

Enables fast time sampling which is dependent on CPU cycles and frequency, otherwise, the directive uses operating system timing (which may be less accurate)

Type: boolean

Default Value: 1

Available since version 4.0

Zend Debugger - PHP API

- Zend Debugger functions
  - debugger_start_debug - Triggers a debug session from within a script
  - debugger_connect - Initiates a tunnel connection

PHP Functions

debugger_start_debug

Triggers a debug session from within a script

Available since version 3.6

Description

```
void debugger_start_debug (void)
```

debugger_connect

Initiates a tunnel connection

Available since version 3.6

Description

```
boolean debugger_connect (void)
```

Return Value

TRUE the connection is established or FALSE could not connect

Zend Optimizer+ - Configuration Directives

Configuration Directives Summary

Directive	Type	Modification Scope	Description
zend_optimizerplus.enable	boolean	PHP_INI_SYSTEM	Optimizer+ On/Off switch. When set to Off, code is not optimized.
zend_optimizerplus.use_cwd	boolean	PHP_INI_SYSTEM	If set to On, use the current directory as a part of the script key
zend_optimizerplus.validate_timestamps	boolean	PHP_INI_ALL	If enabled, the Optimizer+ checks the file timestamps and updates the cache accordingly.
zend_optimizerplus.revalidate_freq	integer	PHP_INI_ALL	How often to check file timestamps for changes to the shared memory storage allocation.
zend_optimizerplus.revalidate_path	boolean	PHP_INI_ALL	Enables or disables file search in include_path optimization
zend_optimizerplus.inherited_hack	boolean	PHP_INI_SYSTEM	Enable this hack as a workaround for "can't redeclare class" errors
zend_optimizerplus.dups_fix	boolean	PHP_INI_ALL	Enable this hack as a workaround for "duplicate definition" errors
zend_optimizerplus.log_verbosity_level	integer	PHP_INI_SYSTEM	The verbosity of the Optimizer+ log
zend_optimizerplus.memory_consumption	integer	PHP_INI_SYSTEM	The Optimizer+ shared memory storage size. The amount of memory for storing precompiled PHP code in Mbytes.
zend_optimizerplus.max_accelerated_files	integer	PHP_INI_SYSTEM	The maximum number of keys (scripts) in the Optimizer+ hash table
zend_optimizerplus.max_wasted_percentage	integer	PHP_INI_SYSTEM	The maximum percentage of "wasted" memory until a restart is scheduled
zend_optimizerplus.consistency_checks	integer	PHP_INI_ALL	Check the cache checksum each N requests
zend_optimizerplus.force_restart_timeout	integer	PHP_INI_SYSTEM	How long to wait (in seconds) for a scheduled restart to begin if the cache is not being accessed
zend_optimizerplus.blacklist_filename	string	PHP_INI_SYSTEM	The location of the Optimizer+ blacklist file
zend_optimizerplus.save_comments	boolean	PHP_INI_SYSTEM	If disabled, all PHPDoc comments are dropped from the code to reduce the size of the optimized code.
zend_optimizerplus.fast_shutdown	boolean	PHP_INI_SYSTEM	If enabled, a fast shutdown sequence is used for the accelerated code
zend_optimizerplus.optimization_level	integer	PHP_INI_SYSTEM	A bitmask, where each bit enables or disables the appropriate Optimizer+ passes

Directive	Type	Modification Scope	Description
zend_optimizerplus.enable_slow_optimizations	boolean	PHP_INI_SYSTEM	Enables or disables the optimization passes that may take significant time, based on an internal runtime calculation

External Configuration File: Optimizer+ blacklist file

The Optimizer+ blacklist file is a text file that holds the names of files that should not be accelerated. The file format is to add each filename to a new line. The filename may be a full path or just a file prefix (i.e., /var/www/x blacklists all the files and directories in /var/www that start with 'x'). Files are usually triggered by one of the following three reasons:

1) Directories that contain auto generated code, like Smarty or ZFW cache.

2) Code that does not work well when accelerated, due to some delayed compile time evaluation.

3) Code that triggers an Optimizer+ bug.

Configuration Directive Details

zend_optimizerplus.enable

Optimizer+ On/Off switch. When set to Off, code is not optimized.

Type: boolean

Default Value: 1

Available since version 4.0

zend_optimizerplus.use_cwd

When this directive is enabled, the Optimizer+ appends the current working directory to the script key, thus eliminating possible collisions between files with the same name (basename). Disabling the directive improves performance, but may break existing applications.

Type: boolean

Default Value: 1

zend_optimizerplus.validate_timestamps

When disabled, you must reset the Optimizer+ manually or restart the webserver for changes to the file system to take effect.
The frequency of the check is controlled by the directive "zend_optimizerplus.revalidate_freq"
Type: boolean
Default Value: 1
Available since version 4.0

zend_optimizerplus.revalidate_freq

How often to check file timestamps for changes to the shared memory storage allocation.
Type: integer
Units: seconds
Default Value: 2
Available since version 4.0

zend_optimizerplus.revalidate_path

If the file search is disabled and a cached file is found that uses the same include_path, the file is not searched again. Thus, if a file with the same name appears somewhere else in include_path, it won't be found. Enable this directive if this optimization has an effect on your applications. The default for this directive is disabled, which means that optimization is active.
Type: boolean
Default Value: 0
Available since version 4.0

zend_optimizerplus.inherited_hack

The Optimizer+ stores the places where DECLARE_CLASS opcodes use inheritance (These are the only opcodes that can be executed by PHP, but which may not be executed because the parent class is missing due to optimization). When the file is loaded, Optimizer+ tries to bind the inherited classes by using the current environment. The problem with this scenario is that, while the DECLARE_CLASS opcode may not be needed for the current script, if the script requires that the opcode at least be defined, it may not run. The default for this directive is disabled, which means that optimization is active.
Type: boolean
Default Value: 1
Available since version 4.0

zend_optimizerplus.dups_fix

Enable this hack as a workaround for "duplicate definition" errors
Type: boolean
Default Value: 0
Available since version 4.0

zend_optimizerplus.log_verbosity_level

All Optimizer+ errors go to the Web server log.
By default, only fatal errors (level 0) or errors (level 1) are logged. You can also enable warnings (level 2), info messages (level 3) or debug messesges (level 4).
For "debug" binaries, the default log verbosity level is 4, not 1.
Type: integer
Default Value: 1

zend_optimizerplus.memory_consumption

The Optimizer+ shared memory storage size. The amount of memory for storing precompiled PHP code in Mbytes.

Type: integer

Units: MBytes

Default Value: 64

Available since version 4.0

zend_optimizerplus.max_accelerated_files

The number is actually the first one in the following set of prime numbers that is bigger than the one supplied: { 223, 463, 983, 1979, 3907, 7963, 16229, 32531, 65407, 130987 }. Only numbers between 200 and 100000 are allowed.

Type: integer

Default Value: 2000

Available since version 4.0

zend_optimizerplus.max_wasted_percentage

The maximum percentage of "wasted" memory until a restart is scheduled

Type: integer

Units: %

Default Value: 5

Available since version 4.0

zend_optimizerplus.consistency_checks

The default value of "0" means that the checks are disabled. Because calculating the checksum impairs performance, this directive should be enabled only as part of a debugging process.
Type: integer
Default Value: 0
Available since version 4.0

zend_optimizerplus.force_restart_timeout

The Optimizer+ uses this directive to identify a situation where there may be a problem with a process. After this time period has passed, the Optimizer+ assumes that something has happened and starts killing the processes that still hold the locks that are preventing a restart. If the log level is 3 or above, a "killed locker" error is recorded in the Apache logs when this happens.
Type: integer
Units: seconds
Default Value: 180
Available since version 4.0

zend_optimizerplus.blacklist_filename

For additional information, see "Extermal Configuration File", above
Type: string
Available since version 4.0

zend_optimizerplus.save_comments

If disabled, all PHPDoc comments are dropped from the code to reduce the size of the optimized code.

Type: boolean

Default Value: 1

Available since version 4.0

zend_optimizerplus.fast_shutdown

The fast shutdown sequence doesn't free each allocated block, but lets the Zend Engine Memory Manager do the work.

Type: boolean

Default Value: 0

Available since version 4.0

zend_optimizerplus.optimization_level

A bitmask, where each bit enables or disables the appropriate Optimizer+ passes

Type: integer

Default Value: 0xffffffbbf

Available since version 4.0

zend_optimizerplus.enable_slow_optimizations

Enables or disables the optimization passes that may take significant time, based on an internal runtime calculation

Type: boolean

Default Value: 1

Available since version 4.0

Zend Optimizer+ - PHP API

- Zend Optimizer+ functions
    - accelerator_reset - Resets the contents of the Optimizer+ shared memory storage.
      Note: This is not an immediate action. The shared memory storage is reset when a request arrives while the shared memory storage is not being used by a script.

PHP Functions

accelerator_reset

Resets the contents of the Optimizer+ shared memory storage.
Note: This is not an immediate action. The shared memory storage is reset when a request arrives while the shared memory storage is not being used by a script.
Available since version 3.6

Description

```
boolean accelerator_reset (void)
```

Return Value

Returns TRUE unless the Optimizer+ is disabled.

Zend Guard Loader - Configuration Directives

Configuration Directives Summary

Directive	Type	Modification Scope	Description
zend_loader.enable	boolean	PHP_INI_SYSTEM	Enables loading encoded scripts. The default value is On
zend_loader.disable_licensing	boolean	PHP_INI_SYSTEM	Disable license checks (for performance reasons)
zend_loader.obfuscation_level_support	integer	PHP_INI_SYSTEM	The Obfuscation level supported by Zend Guard Loader. The levels are detailed in the official Zend Guard Documentation. 0 - no obfuscation is enabled
zend_loader.license_path	string	PHP_INI_SYSTEM	Path to where licensed Zend products should look for the product license. For more information on how to create a license file, see the Zend Guard User Guide

Configuration Directive Details

zend_loader.enable

If you do not plan to use the Zend Guard Loader to load encoded files, you can slightly improve performance by adding the zend_loader.enable = 0.

This disables the transparent auto-loading mechanism that is built into the Zend Guard Loader

Type: boolean

Default Value: 1

Available since version 4.0

zend_loader.disable_licensing

If you do not need to use any licensing features, you can disable the Zend Guard Loader license request. Setting this option lowers Guard Loader memory usage and slightly enhances performance

Type: boolean

Default Value: 0

Available since version 4.0

zend_loader.obfuscation_level_support

The Obfuscation level supported by Zend Guard Loader. The levels are detailed in the official Zend Guard Documentation. 0 - no obfuscation is enabled

Type: integer

Default Value: 3

Available since version 4.0

zend_loader.license_path

Path to where licensed Zend products should look for the product license. For more information on how to create a license file, see the Zend Guard User Guide

Type: string

Available since version 4.0

Zend Guard Loader - PHP API

- Zend Guard Loader functions
  - zend_loader_enabled - Checks the Zend Optimizer+ configuration to verify that it is configured to load encoded files
  - zend_loader_file_encoded - Returns TRUE if the current file was encoded with Zend Guard or FALSE otherwise. If FALSE, consider disabling the Guard Loader
  - zend_loader_file_licensed - Compares the signature of the running file against the signatures of the license files that are loaded into the License Registry by the php.ini file. If a valid license file exists, the values of the license file are read into an array. If a valid license does not exist or is not specified in the php.ini, it is not entered in the PHP server's license registry. If a valid license that matches the product and signature cannot be found in the license directory, an array is not created. For information on the proper installation of a license file, as well as the php.ini directive, see the Zend Guard User Guide
  - zend_loader_current_file - Obtains the full path to the file that is currently running. In other words, the path of the file calling this API function is evaluated only at run time and not during encoding
  - zend_loader_install_license - Dynamically loads a license for applications encoded with Zend Guard.
  - zend_obfuscate_function_name - Obfuscate and return the given function name with the internal obfuscation function
  - zend_current_obfuscation_level - Returns the current obfuscation level support (set by zend_optimizer.obfuscation_level_support) to get information on the product that is currently running.
  - zend_runtime_obfuscate - Start runtime-obfuscation support to allow limited mixing of obfuscated and un-obfuscated code
  - zend_obfuscate_class_name - Obfuscate and return the given class name with the internal obfuscation function
  - zend_get_id - Returns an array of Zend (host) IDs in your system. If all_ids is TRUE, then all IDs are returned, otherwise only IDs considered "primary" are returned
  - zend_loader_version - Returns Zend Guard Loader version

PHP Functions

zend_loader_enabled

Checks the Zend Optimizer+ configuration to verify that it is configured to load encoded files

Available since version 4.0

Description

```
boolean zend_loader_enabled (void)
```

Return Value

Returns TRUE if the Guard Loader is configured to load encoded files. Returns FALSE if the Guard Loader is not configured to load encoded files.

zend_loader_file_encoded

Returns TRUE if the current file was encoded with Zend Guard or FALSE otherwise. If FALSE, consider disabling the Guard Loader

Available since version 4.0

Description

```
boolean zend_loader_file_encoded (void)
```

Return Value

TRUE if Zend-encoded, FALSE otherwise

zend_loader_file_licensed

Compares the signature of the running file against the signatures of the license files that are loaded into the License Registry by the php.ini file. If a valid license file exists, the values of the license file are read into an array. If a valid license does not exist or is not specified in the php.ini, it is not entered in the PHP server's license registry. If a valid license that matches the product and signature cannot be found in the license directory, an array is not created. For information on the proper installation of a license file, as well as the php.ini directive, see the Zend Guard User Guide
Available since version 4.0

Description

```
array zend_loader_file_licensed (void)
```

Return Value

Returns an array or FALSE.
If an array is returned, a valid license for the product exists in the location indicated in the php.ini file.

zend_loader_current_file

Obtains the full path to the file that is currently running. In other words, the path of the file calling this API function is evaluated only at run time and not during encoding

Available since version 4.0

Description

```
string zend_loader_current_file (void)
```

Return Value

Returns a string containing the full path of the file that is currently running

zend_loader_install_license

Dynamically loads a license for applications encoded with Zend Guard.

Available since version 4.0

Description

```
boolean zend_loader_install_license (string $license_file [ , boolean $overwrite = 0 ])
```

Parameters

license_file
: Name of the license file

overwrite
: Controls if the function overwrites old licenses for the same product
0=Do not overwrite

1=Overwrite .
The default value is 0

Return Value

TRUE if the license was loaded successfully, FALSE otherwise

zend_obfuscate_function_name

Obfuscate and return the given function name with the internal obfuscation function

Available since version 4.0

Description

```
string zend_obfuscate_function_name (string $function_name)
```

Parameters

function_name
Name of the function to obfuscate

Return Value

Returns the obfuscated form of the given string.

zend_current_obfuscation_level

Returns the current obfuscation level support (set by zend_optimizer.obfuscation_level_support) to get information on the product that is currently running.
Available since version 4.0

Description

```
int zend_current_obfuscation_level (void)
```

Return Value

Current obfuscation level

zend_runtime_obfuscate

Start runtime-obfuscation support to allow limited mixing of obfuscated and un-obfuscated code
Available since version 4.0

Description

```
boolean zend_runtime_obfuscate (void)
```

Return Value

TRUE if succeeds, FALSE otherwise

zend_obfuscate_class_name

Obfuscate and return the given class name with the internal obfuscation function

Available since version 4.0

Description

```
string zend_obfuscate_class_name (string $class_name)
```

Parameters

class_name
: Name of the class to obfuscate

Return Value

Returns the obfuscated form of the given string

zend_get_id

Returns an array of Zend (host) IDs in your system. If all_ids is TRUE, then all IDs are returned, otherwise only IDs considered "primary" are returned

Available since version 4.0

Description

```
array zend_get_id ([ boolean $all_ids = false ])
```

Parameters

all_ids

If all_ids is TRUE, returns all IDs, otherwise returns only IDs that are considered "primary". The default value is false

Return Value

Array of host IDs

zend_loader_version

Returns Zend Guard Loader version

Available since version 4.0

Description

```
string zend_loader_version (void)
```

Return Value

Zend Guard Loader version

Zend Data Cache - Configuration Directives

Configuration Directives Summary

Directive	Type	Modification Scope	Description
zend_datacache.shm.max_segment_size	integer	PHP_INI_SYSTEM	The maximal size of a shared memory segment
zend_datacache.shm.memory_cache_size	integer	PHP_INI_SYSTEM	Amount of shared memory to be used by the cache
zend_datacache.disk.save_path	string	PHP_INI_SYSTEM	The path for storing cached content to the disk
zend_datacache.disk.dir_level	integer	PHP_INI_SYSTEM	Directory depth, for storing keys
zend_datacache.enable	boolean	PHP_INI_SYSTEM	Enables the Data Cache. The Data Cache cannot work without this directive. The Data Cache can be turned on or off from the Administration Interface
zend_datacache.apc_compatibility	boolean	PHP_INI_SYSTEM	When enabled, the Data Cache extension registers APC compatibility methods

Configuration Directive Details

zend_datacache.shm.max_segment_size

The maximal size of a shared memory segment

Type: integer

Units: MBytes

Default Values:

- Windows, Linux i386, Linux x86-64, Linux AMD64: 32
- Mac OS X, Solaris, FreeBSD i386, FreeBSD x86-64, AIX/PPC: 2

Available since version 4.0

zend_datacache.shm.memory_cache_size

Amount of shared memory to be used by the cache

Type: integer

Units: MBytes

Default Values:

- Windows, Linux i386, Linux x86-64, Linux AMD64: 32
- Mac OS X, Solaris, FreeBSD i386, FreeBSD x86-64, AIX/PPC: 2

Available since version 4.0

zend_datacache.disk.save_path

The path for storing cached content to the disk

Type: string

Default Value: datacache

Available since version 4.0

zend_datacache.disk.dir_level

Directory depth, for storing keys

Type: integer

Default Value: 2

Available since version 4.0

zend_datacache.enable

Enables the Data Cache. The Data Cache cannot work without this directive. The Data Cache can be turned on or off from the Administration Interface

Type: boolean

Default Value: 1

Available since version 4.0

zend_datacache.apc_compatibility

When enabled, the Data Cache extension registers APC compatibility methods

Type: boolean

Default Value: 1

Available since version 4.0

Zend Data Cache - PHP API

- Zend Data Cache functions
  - zend_shm_cache_store - Stores a variable identified by key into the cache. If a namespace is provided, the key is stored under that namespace. Identical keys can exist under different namespaces
  - zend_disk_cache_store - Stores a variable identified by a key into the cache. If a namespace is provided, the key is stored under that namespace. Identical keys can exist under different namespaces
  - zend_shm_cache_fetch - Fetches data from the cache. The key can be prefixed with a namespace to indicate searching within the specified namespace only. If a namespace is not provided, the Data Cache searches for the key in the global namespace
  - zend_disk_cache_fetch - Fetches data from the cache. The key can be prefixed with a namespace to indicate searching within the specified namespace only. If a namespace is not provided, the Data Cache searches for the key in the global namespace
  - zend_shm_cache_delete - Finds and deletes an entry from the cache, using a key to identify it. The key can be prefixed with a namespace to indicate that the key can be deleted within that namespace only. If a namespace is not provided, the Data Cache searches for the key in the global namespace
  - zend_disk_cache_delete - Finds and deletes an entry from the cache, using a key to identify it. The key can be prefixed with a namespace to indicate that the key can be deleted within that namespace only. If a namespace is not provided, the Data Cache searches for the key in the global namespace
  - zend_shm_cache_clear - Deletes all entries from all namespaces in the cache, if a 'namespace' is provided, only the entries in that namespace are deleted
  - zend_disk_cache_clear - Deletes all entries from all namespaces in the cache, if a 'namespace' is provided, only the entries in that namespace are deleted

PHP Functions

zend_shm_cache_store

Stores a variable identified by key into the cache. If a namespace is provided, the key is stored under that namespace. Identical keys can exist under different namespaces

Available since version 4.0

Description

```
boolean zend_shm_cache_store (string $key, mixed $value [ , int $ttl = 0 ])
```

Parameters

key
: The data's key. Optional: prefix with a [namespace::]

value
: Any PHP object that can be serialized

ttl
: - Time to live (TTL), in seconds. The Data Cache keeps an object in the cache as long as the TTL is not expired. Once the TTL is expired, the object is removed from the cache. The default value is 0

Return Value

FALSE if cache storing fails, TRUE otherwise

zend_disk_cache_store

Stores a variable identified by a key into the cache. If a namespace is provided, the key is stored under that namespace. Identical keys can exist under different namespaces

Available since version 4.0

Description

```
boolean zend_disk_cache_store (string $key, mixed $value [ , int $ttl = 0 ])
```

Parameters

key
: The data key. Optional: prefix with a namespace

value
: Any PHP object that can be serialized.

ttl
: - Time to live, in seconds. The Data Cache keeps objects in the cache as long as the TTL is not expired. Once the TTL is expired, the object is removed from the cache. The default value is 0

Return Value

FALSE if cache storing fails, TRUE otherwise

zend_shm_cache_fetch

Fetches data from the cache. The key can be prefixed with a namespace to indicate searching within the specified namespace only. If a namespace is not provided, the Data Cache searches for the key in the global namespace
Available since version 4.0

Description

```
mixed zend_shm_cache_fetch (mixed $key)
```

Parameters

key
: The data key or an array of data keys. Optional for key's name: prefix with a namespace

Return Value

FALSE if no data that matches the key is found, else it returns the stored data, If an array of keys is given, then an array which its keys are the original keys and the values are the corresponding stored data values

zend_disk_cache_fetch

Fetches data from the cache. The key can be prefixed with a namespace to indicate searching within the specified namespace only. If a namespace is not provided, the Data Cache searches for the key in the global namespace
Available since version 4.0

Description

```
mixed zend_disk_cache_fetch (mixed $key)
```

Parameters

key
: The data key or an array of data keys. Optional for key's name: prefix with a namespace

Return Value

FALSE if no data that matches the key is found, else it returns the stored data, If an array of keys is given, then an array which its keys are the original keys and the values are the corresponding stored data values

zend_shm_cache_delete

Finds and deletes an entry from the cache, using a key to identify it. The key can be prefixed with a namespace to indicate that the key can be deleted within that namespace only. If a namespace is not provided, the Data Cache searches for the key in the global namespace
Available since version 4.0

Description

```
boolean zend_shm_cache_delete (mixed $key)
```

Parameters

key
: The data key or an array of data keys. Optional for key's name: prefix with a namespace

Return Value

TRUE on success, FALSE on failure.

zend_disk_cache_delete

Finds and deletes an entry from the cache, using a key to identify it. The key can be prefixed with a namespace to indicate that the key can be deleted within that namespace only. If a namespace is not provided, the Data Cache searches for the key in the global namespace
Available since version 4.0

Description

```
boolean zend_disk_cache_delete (string $key)
```

Parameters

key
: The data key or an array of data keys. Optional for key's name: prefix with a namespace

Return Value

TRUE on success, FALSE on failure or when entry doesn't exist.

zend_shm_cache_clear

Deletes all entries from all namespaces in the cache, if a 'namespace' is provided, only the entries in that namespace are deleted
Available since version 4.0

Description

```
boolean zend_shm_cache_clear (string $namespace)
```

Parameters

namespace
: The data key. Optional: prefix with a namespace

Return Value

If the namespace does not exist or there are no items to clear, the function will return TRUE. The function will return FALSE only in case of error.

zend_disk_cache_clear

Deletes all entries from all namespaces in the cache, if a 'namespace' is provided, only the entries in that namespace are deleted
Available since version 4.0

Description

```
boolean zend_disk_cache_clear (string $namespace)
```

Parameters

namespace
: The data key. Optional: prefix with a namespace

Return Value

If the namespace does not exist or there are no items to clear, the function will return TRUE. The function will return FALSE only in case of error.

Zend Java Bridge - Configuration Directives

Configuration Directives Summary

Directive	Type	Modification Scope	Description
zend_jbridge.server_port	integer	PHP_INI_SYSTEM	The TCP port on which the server is listening
zend_jbridge.ints_are_longs	boolean	PHP_INI_ALL	Converts PHP integers into java.lang.Long integers, primarily for 64-bit machines
zend_jbridge.encoding	string	PHP_INI_ALL	Sets the encoding type that is passed from PHP to Java
zend_jbridge.use_java_objects	boolean	PHP_INI_ALL	Uses basic Java objects and does not attempt to convert them to primitives

Configuration Directive Details

zend_jbridge.server_port

Default is 10001. Must be the same as the server's zend.javamw.port

Type: integer

Default Value: 10001

Available since version 4.0

zend_jbridge.ints_are_longs

Translates PHP integer values to java.lang.Long integers (64-bit) instead of java.lang.Integer integers (32-bit). The default setting is off

Type: boolean

Default Value: 0

Available since version 4.0

zend_jbridge.encoding

Sets the encoding type that is passed from PHP to Java

Type: string

Default Value: UTF-8

Available since version 4.0

zend_jbridge.use_java_objects

When set to 0, preserves the current implementation (which converts basic Java objects to primitives (e.g., java.long.Short to short).

When set to 1 for the Java Bridge, returns Java objects and does not convert them to primitives

Type: boolean

Default Value: 0

Available since version 4.0

Zend Java Bridge - PHP API

- JavaException - The JavaException class
  - JavaException::getCause - Get the Java exception that led to this exception
- Zend Java Bridge functions
  - java - Creates a Java object
  - java_last_exception_get - Returns a Java exception object for the last exception that occurred in the script: only the last exception is stored by the Java Bridge
  - java_last_exception_clear - Clears the last Java exception object record from the Java Bridge storage
  - java_set_ignore_case - Sets the case sensitivity for Java calls when there are mixed cases in your PHP script
  - java_throw_exceptions - Controls if exceptions are thrown on Java exception. When an exception is thrown by a Java application, this function controls if the exception caught by the PHP code will continue to be thrown or not (if not, it is stored in the Java Bridge's internal memory)
  - java_set_encoding - Sets encoding for strings received by Java from the PHP code to verify that the encoding is the same in the PHP and Java code
  - java_require - Includes an additional CLASSPATH/JAR in a PHP script context
  - java_reload - Reloads Jar files that were dynamically loaded - on demand

PHP Functions

java

Creates a Java object

Available since version 3.6

Description

```
object java (string $class_name [ , ... ])
```

Parameters

class_name
: Class name to create

...
: Additional arguments are treated as constructor parameters

Return Value

The Java object that was created, NULL otherwise

java_last_exception_get

Returns a Java exception object for the last exception that occurred in the script: only the last exception is stored by the Java Bridge
Available since version 3.6

Description

```
object java_last_exception_get (void)
```

Return Value

Java exception object, if there was an exception, NULL otherwise

java_last_exception_clear

Clears the last Java exception object record from the Java Bridge storage
Available since version 3.6

Description

```
void java_last_exception_clear (void)
```

java_set_ignore_case

Sets the case sensitivity for Java calls when there are mixed cases in your PHP script

Available since version 3.6

Description

```
void java_set_ignore_case (boolean $ignore)
```

Parameters

ignore
: If set, the Java attribute and method names are resolved, regardless of case

java_throw_exceptions

Controls if exceptions are thrown on Java exception. When an exception is thrown by a Java application, this function controls if the exception caught by the PHP code will continue to be thrown or not (if not, it is stored in the Java Bridge's internal memory)

Available since version 3.6

Description

```
void java_throw_exceptions (int $throw)
```

Parameters

throw
: If true, a PHP exception is thrown when a Java exception happens. If set to FALSE, use java_last_exception_get() to check for exceptions

java_set_encoding

Sets encoding for strings received by Java from the PHP code to verify that the encoding is the same in the PHP and Java code

Available since version 3.6

Description

```
void java_set_encoding ([ string $encoding = UTF-8 ])
```

Parameters

encoding
: Default encoding type is UTF-8. The default value is UTF-8

java_require

Includes an additional CLASSPATH/JAR in a PHP script context

Available since version 3.6

Description

```
void java_require (string $path)
```

Parameters

path
: URL pointing to the location of the Jar file. This function accepts the following protocols:
https://, http://, file://, ftp://
It can also be a local path: E.g., c:\

java_reload

Reloads Jar files that were dynamically loaded - on demand

Available since version 3.6

Description

```
void java_reload (string $new_jarpath)
```

Parameters

new_jarpath
: The path to the Jar files

The JavaException Class

JavaException is a PHP class that inherits from the default PHP5 class "Exception"

Available since: 3.6

Class Prototype

```
class JavaException {
/* Methods */
public object getCause (void)
}
```

Class Methods

JavaException::getCause

Get the Java exception that led to this exception

Available since version 3.6

Description

```
public object JavaException::getCause (void)
```

Return Value

A Java exception object, if there was an exception, NULL otherwise

Zend Extension Manager - Configuration Directives

Configuration Directives Summary

Directive	Type	Modification Scope	Description
zend_extension_manager.log_verbosity_level	integer	PHP_INI_SYSTEM	Log message verbosity level.
zend_extension_manager.load_order_file	string	PHP_INI_SYSTEM	The path to the location of the load file. The load file contains the information about the extensions' loading order
zend_extension_manager.activate_signal_handlers	boolean	PHP_INI_SYSTEM	UNIX only: Activates SIGSEGV and SIGABRT signal handlers.
zend_extension_manager.wait_for_debugger	boolean	PHP_INI_SYSTEM	UNIX only: Automatically pauses the process received by SIGSEGV and SIGABRT.

External Configuration File: load order file

The load order file specifies the extension loading order. This file contains plain text with a single 'extensionId' per line. The extension IDs are not necessarily found in php.ini files. The nonexistent IDs are not considered, but may be reserved for future use (i.e., system upgrade)
This file should not be modified.

Configuration Directive Details

zend_extension_manager.log_verbosity_level

The default level is usually set to 1, which includes very important information messages, errors and warnings.
Switch the level to 2 to see the notices. Higher levels (up to 5) are reserved for debug purposes only.

IMPORTANT: The ZEM is absolutely required to load any Zend extension from the Zend product line. There is other way to load Zend extensions besides using the ZEM.
Type: integer
Default Value: 1
Available since version 3.6

zend_extension_manager.load_order_file

The file should be in plain text. Each line should list only one extensionId. The order of lines (extensionIds) determines the order of loading the appropriate extensions. If a particular extensionId is not managed in any INI file, the ID is skipped.
Type: string
Default Value: zem_order
Available since version 3.6

zend_extension_manager.activate_signal_handlers

If enabled, the stack trace is printed when the signal is received. This directive can be combined with 'zend_extension_manager.wait_for_debugger'.

Type: boolean

Default Value: false

Available since version 3.6

zend_extension_manager.wait_for_debugger

If enabled, the process is paused when the signal is received, so that 'gdb' can be easily attached. 'zend_extension_manager.activate_signal_handlers' must be enabled.

Type: boolean

Default Value: false

Available since version 3.6

Zend Utils - Configuration Directives

Configuration Directives Summary

Directive	Type	Modification Scope	Description
zend_utils.log_verbosity_level	integer	PHP_INI_SYSTEM	Sets the log verbosity level of Zend Utilis logs [0-5]
zend_utils.use_graceful_restart	boolean	PHP_INI_SYSTEM	Restart API uses a graceful restart
zend_utils.aix_restart_cmd	string	PHP_INI_SYSTEM	Command to restart web server on aix

Configuration Directive Details

zend_utils.log_verbosity_level

Verbosity_level for most components that have a log (except Zend Debugger, Zend Guard Loader and Optimizer+):

0-ZERROR (is always be displayed - indicates an error that can't be recovered)

1-ZWARNING (displays a warning - indicates a warning (recoverable error) that the application can still run with)

2-ZNOTICE (displays a notice - indicates that something wrong has happened)

3-ZDBG1 (debug purposes only - high priority messages)

4-ZDBG2 (debug purposes only - medium priority messages)

5-ZDBG3 (debug purposes only - low priority messages)

Type: integer

Default Value: 2

Available since version 4.0

zend_utils.use_graceful_restart

Restart API uses a graceful restart

Type: boolean

Default Value: 0

Available since version 4.0

zend_utils.aix_restart_cmd

Command to restart web server on aix

Type: string

Default Value: CALL PGM(ZENDSVR/ZCCPHPR001)

Available since version 4.0

Zend Download Server - Configuration Directives

Configuration Directives Summary

Directive	Type	Modification Scope	Description
zend_dserver.enable	boolean	PHP_INI_SYSTEM	Enables or disables the Zend Download Server (ZDS) component. This can also be done in Zend Server, from Server Setup | Components. When turned to 'On', the ZDS passes downloads to a dedicated process. When turned to 'Off', all downloads are handled by the Apache server
zend_dserver.mime_types_file	string	PHP_INI_SYSTEM	The full path to the MIME type file.
zend_dserver.log_file	string	PHP_INI_SYSTEM	The location of the Zend Download Server (ZDS) log file
zend_dserver.log_verbosity	integer	PHP_INI_SYSTEM	Log's Verbosity Level
zend_dserver.min_file_size	integer	PHP_INI_SYSTEM	The minimal file size that can be served via a ZDS process. Smaller files are served via Apache
zend_dserver.nice	integer	PHP_INI_SYSTEM	The ZDS process priority level. The lower the number, the higher the priority the process is given.
zend_dserver.disable_byterange	boolean	PHP_INI_SYSTEM	Disables handling byte-range requests. All requests return an entire file
zend_dserver.etag_params	string	PHP_INI_SYSTEM	The file attributes that are taken as part of an etag.
zend_dserver.mmap_chunk	integer	PHP_INI_SYSTEM	The size of the data chunks that are read from the file into the socket.

External Configuration File: mime_types

The mime_types file is an external list of file extensions that should be sent through the Zend Download Server.

Configuration Directive Details

zend_dserver.enable

Enables or disables the Zend Download Server (ZDS) component. This can also be done in Zend Server, from Server Setup | Components. When turned to 'On', the ZDS passes downloads to a dedicated process. When turned to 'Off', all downloads are handled by the Apache server

Type: boolean

Default Value: 1

zend_dserver.mime_types_file

The full path to the MIME type file.

Type: string

Default Value: zend_mime_types.ini

Available since version

zend_dserver.log_file

The location of the Zend Download Server (ZDS) log file

Type: string

Default Value: ZDS.log

Available since version

zend_dserver.log_verbosity

The extension's log verbosity level.

1 - Fatal errors (ZDS goes down)

2 - Errors (The currect request bails out

3 - Warnings (not good, but continue)

4 - Notice (important info)

5 - Info (verbosity information)

Type: integer

Default Value: 2

Available since version

zend_dserver.min_file_size

The minimal file size that can be served via a ZDS process. Smaller files are served via Apache

Type: integer

Units: KBytes

Default Value: 64

Available since version

zend_dserver.nice

The ZDS process priority level. The lower the number, the higher the priority the process is given.

Type: integer

Default Value: 10

Available since version

zend_dserver.disable_byterange

Disables handling byte-range requests. All requests return an entire file

Type: boolean

Default Value: 0

Available since version

zend_dserver.etag_params

The file attributes that are taken as part of an etag.

Type: string

Available since version

zend_dserver.mmap_chunk

The size of the data chunks that are read from the file into the socket.

Type: integer

Units: KBytes

Default Value: 256

Available since version

Adding Extensions

Zend Server Community Edition users can benefit from extension management capabilities for third party extensions as well as for Zend Extensions. This enables users to load and unload all extensions directly from the Zend Server Community Edition Extensions page.

Important: The newly added extensions will be visible in the Administration Interface's Extensions page however, the directive configuration option will not be active and directives belonging to the extension have to be configured directly from the php.ini file.

Disclaimer:

Zend Technologies does not provide support for third party products, including extensions. Therefore, if an issue for support arises, please remove all third party extensions by commenting out the reference to them in your php.ini before referring to the Support Center - http://www.zend.com/en/support-center/.

There are two types of extensions: PHP extensions and Zend extensions. The extension provider should supply information regarding the extension type (Zend or PHP). Make sure to also check the provider's documentation for possible compatibility issues, PHP version compatibility and any other additional configurations that may be required.

To add Zend extensions:

1. Download the extension

   Note: - AIX Unix/Linux extensions end with the .so suffix.
2. Place the extension in your extensions directory.

   To locate the extensions directory, open the Administration Interface to **Monitor | PHP Info** and check the value for the directive extension_dir=.

   By default, your extensions directory is located in:

   <install_path>/zend/lib/php_extensions
3. Add the following line to your php.ini:

   zend_extension=<full_path_to_extension_so_file>
4. Restart your server.
5. **To restart your server**:

   Click Restart Server 

   in the Administration Interface.

   Ensure that the extension is properly loaded by checking the output of PHPInfo in the Administration Interface.

Note:

If you try to load a PHP extension as a Zend extension, in Linux you may receive the following error message in your server's error log: "<extension_name> doesn't appear to be a valid Zend extension."

If this occurs, remove it and add it as a PHP extension, following the instructions under "To Add PHP Extensions", below.

To add PHP extensions

1. Download the third party extension. Many third party extensions can be found at http://pecl.php.net.
   Extensions are obtained directly from external web repositories.
2. Place the PHP extension in your extensions directory.
   To locate the extensions directory, open your php.ini and check the value for the directive extension_dir=.
   By default, your extensions directory is located in:
   <install_path>/lib/php_extensions
3. Add the following line to your php.ini:
   extension=<my_extension_name>.so
   Ensure that you replace <my_extension_name> with your extension's name.
4. Restart your Web server.
   Ensure that the extension is properly loaded by checking the Administration Interface: See **Monitor | PHP Info** for the output of PHP Info.

The extensions appear in your Administration Interface under the Extensions tab and you can use the Administration Interface to load and unload the extension.

Adding Extensions for Windows

The following procedure describes how to download compiled extensions for Wndows DLL files.

Windows Note:

When downloading extensions for Windows from PECL, make sure to download the non thread-safe (NTS) version **ONLY**.

To download extensions:

1. Go to: http://www.php.net/downloads.php.
2. In the Windows binaries section, select: "**PECL <current ZendServer PHP version> Non-thread-safe Win32 binaries**" (64-bit users can use this too).
3. Click the package to start a download process. Follow the download instructions and extract the ZIP file.
4. Select the .dll you want.
5. To add the extension, go to the extension directory, *<install_path>*\ZendServer\lib\phpext, and add the .dll file there.
6. Go to your php.ini file and add the following line: extension=<extension_name>.dll.
7. To verify that the extension was loaded properly, go to **Setup | Extensions** and locate the extension from the list.
   When loading new extensions, also examine the log files.

For more information on these extensions, go to http://pecl4win.php.net/ .

Note: The extensions in this site are thread-safe and therefore should not be downloaded for use with Zend Server Community Edition .

Note:

Some extensions need directives to change the Extension's default configurations. These directives should be added added to your php.ini file manually. There is no way to predict which directives extensions may have: For each third party extension you want to add, make sure to go to the project's source site to check for additional information related to the extension.

Compiling Extensions

Under Unix/Linux operating systems you can also create and compile your own extensions using the phpize command.

Disclaimer:

External extensions are not supported by Zend. If you encounter a problem, remove any additional extensions before contacting Zend Support.

Building PHP extensions from source requires basic UNIX skills as well as several build tools, among others:

- An ANSI C compiler
- flex: Version 2.5.4
- bison: Version 1.28 (recommended), 1.35, or 1.75
- Any specific components or libraries required by the extension being built (such as gd, pdf libs, etc.)

To compile extensions from source:

1. Download and extract the extension's source.
2. Switch to the extension source directory (by default located in <install_path>/Zend/ZendServer/lib/phpext) and run the following commands:

   ```
   cd <your_extension_directory>
   <install_path>/bin/phpize
   ```

   Ensure that you replace <your_extension_directory> with your extension directory's name.
3. Run the ./configure command to prepare the source for compilation. You will need to include the "php-config" and "enable-shared" flags as follows:

   ```
   ./configure --with-php-config=<install_path>/bin/php-config\
               --enable-shared
   ```

   Note:

   Some extensions will need additional configuration flags. It is therefore advised to run "./configure --help" and review the possible flags before compiling.
4. Compile and install the extension binaries by running the following commands:

   ```
   make
   make install
   ```

   Make install should install the new .so extension binary in Zend Server Community Edition's extension directory.

5. Add the following line to your php.ini to load your new extension:

   ```
   extension=<my_extension_name>.so
   ```

   Replace <my_extension_name> with your extension's binary name.
6. Restart your Web server.
7. Ensure that the extension is properly loaded by checking the output of PHP Info. This can be viewed in the Zend Server Community Edition PHP Info page.

The extension appears in your Administration Interface under the Extensions page and you can use the Administration Interface to load and unload the extension.

UNIX: Compiling PHP Extensions

This procedure describes how to compile a PHP extension. Zend Server Community Edition includes over 77 extensions however there still may be a PHP extension that you want to compile by yourself.

Requirements:

- **PHP Tools**:
  - PECL (PHP Extension Community Library): PECL is a repository for PHP extensions, providing a directory of all known extensions and hosting facilities for download and development of PHP extensions. - It is also a tool supplied in the form of a small shell script with PHP code behind it to retrieve extensions from the aforementioned repository.
  - phpize: a shell script to generate a configure script for PHP extensions
- **Build Tools**:

  While PHP can be built using many different tool chains, this article will focus on using the GNU tool chain. The main tools where PHP is concerned are:
  - autoconf: automatic configure script builder. This is called by the phpize script.
  - automake: a tool for generating GNU Standards-compliant Makefiles
  - libtool: Generic library support script. Libtool hides the complexity of generating special library types (such as shared libraries) behind a consistent (sort of :)) interface.
  - GNU make: a GNU tool for controlling the generation of executables and other non-source files of a program from the program's source files
  - GCC: PHP extensions are typically written in C. Hence, in order for them to compile, you would need a C compiler. While GCC now stands for GNU compiler Collection and is no longer just a GNU C Compiler, for our purposes we only need the C part of the collection. GNU's elf-binutils package: The programs in this package are used to assemble, link and manipulate binary and object files.

Install the following packages:
Users of distributions with package managers (mainly Debian, Ubuntu, RHEL, CentOS and Fedora Core and many others) should install the following packages from their distribution's repository: gcc, make, autoconf, automake and libtool. Some of these tools depend on each other, for instance the libtool package depends on the gcc package, but no damage can be done from specifying all of them.

Note:
Users who utilize distributions that do not have package managers (Linux from scratch anyone?), can compile these tools themselves or obtain pre-compiled binaries for them quite easily.

Additionally, you can compile a PHP extension from the main PHP source (as opposed to PECL). This requires installing a package from the Zend Server Community Edition repository called *php-5.2-source-zend-server* or *php-5.3-source-zend-server*, depending on your Zend Server Community Edition's major PHP version. This package includes full PHP sources as patched, for security or optimization concerns, by the Zend development team. This ensures that you are using the exact same source code we used when building Zend Server Community Edition.

Scenario 1: compile a PECL extension called Newt

Newt is a PHP extension for RedHat's Newt (New Terminal) library, a terminal-based window and widget library for writing applications with user friendly interfaces.
Being what it is, this extension requires the existence of the Newt library development files. If you are using Debian or Ubuntu you should install a package called libnewt-dev. On RedHat based distributions the package name is newt-devel. Make sure these are installed before continuing.
NOTE: Other extensions will have other dependencies. For example, the Mcrypt extension will require the Mcrypt development package.
NOTE: Since PECL will attempt to write the extension onto /usr/local/zend/lib/php_extensions, you will have to become a super user to perform this procedure. This is only needed for the actual make install.

To compile your own extension:

1. Assuming you have the Newt development package installed, run:
   # /usr/local/zend/bin/pecl install newt

The truncated output of this command, along with explanations:

PECL retrieves the package from the repository...*/ downloading newt-1.2.1.tgz

```
Starting to download newt-1.2.1.tgz (24,853 bytes)
.........done: 24,853 bytes
5 source files, building
/*The phpize script is executed...*/
running: phpize
Configuring for:
PHP Api Version:         20041225
Zend Module Api No:      20060613
Zend Extension Api No:   220060519
building in /var/tmp/pear-build-root/newt-1.2.1
```

Configure comes into play

```
running: /tmp/pear/download/newt-1.2.1/configure
checking for grep that handles long lines and -e... /bin/grep
checking for egrep... /bin/grep -E checking for a sed that does
not truncate output... /bin/sed checking for gcc... gcc checking
for C compiler default output file name... a.out checking whether
the C compiler works... yes checking whether we are cross
compiling... no checking for suffix of executables...
checking for suffix of object files... o
```

Next comes libtool.

```
creating libtool
appending configuration tag "CXX" to libtool
configure: creating ./config.status
config.status: creating config.h
```

The actual compilation process: calls make which internally triggers GCC and LD.

```
running: make
/bin/sh /var/tmp/pear-build-root/newt-1.2.1/libtool --
mode=compile gcc -I. -I/tmp/pear/download/newt-1.2.1 -
DPHP_ATOM_INC -I/var/tmp/pear-build-root/newt-1.2.1/include
```

```
-I/var/tmp/pear-build-root/newt-1.2.1/main

-I/tmp/pear/download/newt-1.2.1 -I/usr/local/zend/include/php -
I/usr/local/zend/include/php/main -
I/usr/local/zend/include/php/TSRM

-I/usr/local/zend/include/php/Zend -
I/usr/local/zend/include/php/ext -
I/usr/local/zend/include/php/ext/date/lib

-I/usr/local/zend/include/php -DHAVE_CONFIG_H  -g -O2   -c

/tmp/pear/download/newt-1.2.1/newt.c -o newt.lo mkdir .libs  gcc
-I. -I/tmp/pear/download/newt-1.2.1 -DPHP_ATOM_INC -
I/var/tmp/pear-build-root/newt-1.2.1/include

-I/var/tmp/pear-build-root/newt-1.2.1/main

-I/tmp/pear/download/newt-1.2.1 -I/usr/local/zend/include/php -
I/usr/local/zend/include/php/main -
I/usr/local/zend/include/php/TSRM

-I/usr/local/zend/include/php/Zend -
I/usr/local/zend/include/php/ext -
I/usr/local/zend/include/php/ext/date/lib -
I/usr/local/zend/include/php -DHAVE_CONFIG_H -g -O2 -c
/tmp/pear/download/newt-1.2.1/newt.c  -fPIC -DPIC -o .libs/newt.o
/bin/sh /var/tmp/pear-build-root/newt-1.2.1/libtool --
mode=compile gcc -I. -I/tmp/pear/download/newt-1.2.1 -
DPHP_ATOM_INC -I/var/tmp/pear-build-root/newt-1.2.1/include

-I/var/tmp/pear-build-root/newt-1.2.1/main

-I/tmp/pear/download/newt-1.2.1 -I/usr/local/zend/include/php -
I/usr/local/zend/include/php/main -
I/usr/local/zend/include/php/TSRM

-I/usr/local/zend/include/php/Zend -
I/usr/local/zend/include/php/ext -
I/usr/local/zend/include/php/ext/date/lib

-I/usr/local/zend/include/php -DHAVE_CONFIG_H  -g -O2   -c

/tmp/pear/download/newt-1.2.1/newt_vcall.c -o newt_vcall.lo  gcc
-I. -I/tmp/pear/download/newt-1.2.1 -DPHP_ATOM_INC -
I/var/tmp/pear-build-root/newt-1.2.1/include

-I/var/tmp/pear-build-root/newt-1.2.1/main

-I/tmp/pear/download/newt-1.2.1 -I/usr/local/zend/include/php -
I/usr/local/zend/include/php/main -
I/usr/local/zend/include/php/TSRM

-I/usr/local/zend/include/php/Zend -
I/usr/local/zend/include/php/ext -
I/usr/local/zend/include/php/ext/date/lib -
I/usr/local/zend/include/php -DHAVE_CONFIG_H -g -O2 -c
/tmp/pear/download/newt-1.2.1/newt_vcall.c

-fPIC -DPIC -o .libs/newt_vcall.o

/bin/sh /var/tmp/pear-build-root/newt-1.2.1/libtool --mode=link
gcc -DPHP_ATOM_INC -I/var/tmp/pear-build-root/newt-1.2.1/include

-I/var/tmp/pear-build-root/newt-1.2.1/main
```

```
-I/tmp/pear/download/newt-1.2.1 -I/usr/local/zend/include/php -
I/usr/local/zend/include/php/main -
I/usr/local/zend/include/php/TSRM

-I/usr/local/zend/include/php/Zend -
I/usr/local/zend/include/php/ext -
I/usr/local/zend/include/php/ext/date/lib

-I/usr/local/zend/include/php -DHAVE_CONFIG_H  -g -O2   -o
newt.la

-export-dynamic -avoid-version -prefer-pic -module -rpath
/var/tmp/pear-build-root/newt-1.2.1/modules  newt.lo
newt_vcall.lo -lnewt gcc -shared  .libs/newt.o .libs/newt_vcall.o
 -lnewt  -Wl,-soname -Wl,newt.so -o .libs/newt.so creating
newt.la (cd .libs && rm -f newt.la && ln -s ../newt.la newt.la)
/bin/sh /var/tmp/pear-build-root/newt-1.2.1/libtool --
mode=install cp ./newt.la /var/tmp/pear-build-root/newt-
1.2.1/modules

cp ./.libs/newt.so /var/tmp/pear-build-root/newt-
1.2.1/modules/newt.so

cp ./.libs/newt.lai /var/tmp/pear-build-root/newt-
1.2.1/modules/newt.la

PATH="$PATH:/sbin" ldconfig -n /var/tmp/pear-build-root/newt-
1.2.1/modules

----------------------------------------------------------------
-----

Libraries have been installed in:

   /var/tmp/pear-build-root/newt-1.2.1/modules

Build complete.
```

2. Run 'make test'.
3. Use PECL to put the newly built Newt extension into place.

   run: *make INSTALL_ROOT="/var/tmp/pear-build-root/install-newt-1.2.1"*
4. instal the shared extensions by running:

   var/tmp/pear-build-root/install-newt-1.2.1//usr/local/zend/lib/php_extensions/

```
running: find "/var/tmp/pear-build-root/install-newt-1.2.1" |
xargs ls -dils

574096   4 drwxr-xr-x 3 root root   4096 Mar 30 20:45

/var/tmp/pear-build-root/install-newt-1.2.1

574119   4 drwxr-xr-x 3 root root   4096 Mar 30 20:45

/var/tmp/pear-build-root/install-newt-1.2.1/usr

574120   4 drwxr-xr-x 3 root root   4096 Mar 30 20:45

/var/tmp/pear-build-root/install-newt-1.2.1/usr/local

574121   4 drwxr-xr-x 3 root root   4096 Mar 30 20:45
```

```
/var/tmp/pear-build-root/install-newt-1.2.1/usr/local/zend
574122   4 drwxr-xr-x 3 root root   4096 Mar 30 20:45
/var/tmp/pear-build-root/install-newt-1.2.1/usr/local/zend/lib
574123   4 drwxr-xr-x 2 root root   4096 Mar 30 20:45
/var/tmp/pear-build-root/install-newt-
1.2.1/usr/local/zend/lib/php_extensions
574118 244 -rwxr-xr-x 1 root root 241717 Mar 30 20:45
/var/tmp/pear-build-root/install-newt-
1.2.1/usr/local/zend/lib/php_extensions/newt.so
Build process completed successfully
Installing '/usr/local/zend/lib/php_extensions/newt.so'
install ok: channel://pear.php.net/newt-1.2.1
```

5. The Extension has been successfully compiled using PECL.
6. To load the extension, in the php.ini or in a separate file under the scan dir insert extension=<my_extension_name>.so and replace <my_extension_name> with your extension's binary name such as "*extension=newt.so*".
7. If you're using the DEB and RPM versions of Zend Server Community Edition, the best practice is to place a file called newt.ini under /usr/local/zend/etc/conf.d.
8. Restart your webserver.

Ensure the extension is properly loaded by checking the output of PHP Info. This can be viewed in the Zend Server Community Edition PHP Info page.
The extension will now appear in your Administration Interface under **Server Setup | Extensions** from which you can also load and unload the extension (for more information see: Working with Extensions).

Scenario 2: Compile a PHP extension included in the main PHP source called PSpell

Pspell (Portable Spell Checker Interface Library) provides a generic interface to the system spelling checking libraries. To compile PSpell first install the *php-source-zend-[ce|pe]* package for this procedure. Also, since this extension relies on the portable spell-checking interface (pspell) library, you will need to install its devel package. Debian and Ubuntu users should install the *libpspell-dev* package, on RedHat based distributions, the package name is *aspell-devel*.

To compile your own extension:

1. CD the extension's source dir (in our example, the PHP version is 5.2.9 as it is the current stable version Zend Server Community Edition is shipped with):
   $ cd /usr/local/zend/share/php-source/php-5.2.9/ext/pspell
2. Run phpize:
   $ /usr/local/zend/bin/phpize

The output should be similar to this:

```
/Configuring for:
PHP Api Version:         20041225
Zend Module Api No:      20060613
Zend Extension Api No:   220060519/
Run the configure script, generated by phpize:
$ ./configure --with-php-config=/usr/local/zend/bin/php-config
```

3. Run make:
   $ make
4. Become a super user [root] and run:
   # make install

The output should be:

```
/Installing shared extensions:
    /usr/local/zend/lib/php_extensions/
```

5. Insert the "extension=pspell.so" directive either in php.ini or in a separate file under the scan dir.
6. Restart your webserver.

Ensure the extension is properly loaded by checking the output of PHP Info. This can be viewed in the Zend Server Community Edition PHP Info page.

The extension will now appear in your Administration Interface under **Server Setup | Extensions** from which you can also load and unload the extension (for more information see: Working with Extensions).

Troubleshooting:
The configure script outputs messages as it goes along and many times you will be able to understand the problem just by looking at it, however, sometimes, the error doesn't necessarily reflect the real issue so it is always a good idea to review the config.log. This is a very generic statement but no other statement can be made as there are many different extensions and issues one may come across so attempting to list them all will be somewhat futile.

Loading the mod_ssl Module

The mod_ssl module allows you to enable SSL support on your Apache web server and is needed to enable Apache for SSL requests (https).
For more information on the mod_ssl module, see the mod_ssl user manual at http://www.modssl.org/docs/2.8.

The bundled Apache that comes with Zend Server Community Edition includes support for the ssl_module, but this needs to be loaded in order to activate it. You must have acquired an SSL certificate from an SSL certificate provider (e.g., http://www.slacksite.com/apache/certificate.html) or have created your own SSL certificate for the mod_ssl to be loaded.

To load the mod_ssl module:

1. Open your httpd.conf file.
   By default, this is located in:

   <install_path>/apache2/conf/httpd.conf
2. Un-comment the following line by removing the "#".

   ```
   Include conf/extra/httpd-ssl.conf
   ```

   This calls the SSL configuration file.
3. Place your server.crt and server.key certification files in the 'conf' folder.
4. Restart the Apache server for the changes to take effect.

The mod_ssl module is loaded.

Java Bridge Use Cases

This section describes some of the common uses for the Java Bridge. The usage scenarios and examples discussed here provide a framework for the Java Bridge's uses, rather than a complete picture. Real world experience indicates that companies are finding more and more applications for the Java Bridge, beyond what was initially anticipated.

Usage Scenarios

There are two usage scenarios that describe the most common applications for the PHP/Java Bridge:

- **Integration with Existing Java Infrastructure** - PHP is a fully featured scripting language engineered to cover virtually all of an enterprise's requirements. At the same time, many enterprises have a long history of application development in Java. The Java Bridge enables enterprises to continue to use their Java infrastructure - applications, databases, business logic and various Java servers (WebLogic, JBoss, Oracle Application Server, etc.).
- **Accessing Java Language and Architecture** - Some enterprises require the full set of PHP capabilities, yet have a specific need for select Java based applications. SIP signaling in the communications industry or JDBC for creating connectivity to SQL databases are two examples of impressive, industry specific products. The Java Bridge enables enterprises to adopt a PHP standard and to use their preferred Java based applications.

Activities

This section describes two sample activities that indicate some of what you can do with the PHP/Java Bridge. In the sample activities, it is important to differentiate between Java and J2EE. The difference will impact on architecture and in turn, on the script code.

The important differences are:

- Java is a programming language. Java applications created in Java for the enterprise are not bound to a specific framework. Therefore, it is possible and perhaps preferable for an enterprise to relocate code libraries to the server that runs Zend Server Community Edition .
- J2EE is a structured framework for application scripts developed for J2EE. It is preferable that J2EE servers be left intact.

Example 1: A Case Study in Java Bridge Performance (Java)

The Forever Times newspaper maintains a PHP-based website - let's call it ForeverOnline.com. The newspaper has been searching for a real-time Stock Ticker application to add to their already successful and heavily visited website. The Forever Times Newspaper feels that real-time financial information is the one thing their website is lacking.

Forever Times believes they have found exactly the Stock Ticker application they need. The application provides up-to-date quotations from all the major markets, currency rates, and even links to some of the local exchanges. However, the application is written in Java and uses existing Java libraries.

Forever Times realizes that a PHP-based Web implementation that handles Java requests - a Java Bridge - is their best bet. At the same time, they are concerned that the performance of their Website remains optimal. To Forever Times' horror, in testing the new application, they find that loading the site with user-requests for the Stock Ticker slows down the performance of the whole website.

The following code example illustrates how the Java Bridge applies to this business scenario and others like it:

Example:

```
<?
// create Java object
$stock = new Java("com.ticker.JavaStock");
// call the object
$news = $stock->get_news($_GET['ticker']);
// display results
foreach($news as $news_item) {
print "$news_item<br>\n";
}
?>
```

The example code can be understood as follows:

- The code example is written in PHP and forms part of a PHP Web application.
- The PHP code creates the Java object-"com.ticker.JavaStock"-which is the PHP proxy.
- Requests come into the PHP based Website - ForeverOnline.com - which then references the Stock Ticker application.
- Stock Ticker references a custom object- get_news-in the JVM library. This is all in native Java.
- The PHP code then outputs the results on the Website.

As opposed to a typical Java Bridge Implementation, the Zend Server Community Edition Java Bridge implementation addresses performance issues through the Java Bridge architecture. Implementing the Java Bridge is a way to address scalability issues by using the Java Bridge to handle all communication in a single JVM instance, instead of in several instances.

Note:

While the single JVM constitutes a single point of failure, the fact is, Zend's PHP-Java connection is the most robust on the market. Failures in systems of this type generally tend to occur when the Java Server is overloaded, rather than as a result of glitches in the applications. Zend Server Community Edition 's system architecture insures performance by diminishing overhead. However, in the event of failure, the Java Bridge supports a restart feature that makes monitoring the status of the Java Server and restarting quick and simple. One last point: if the failure was caused by a glitch in the application, the same thing would most likely occur in each of the JVMs in the non-Zend system!

Example 2: A Case Study in Management Integration (J2EE)

A company called FlowerPwr.com sells flowers over the Internet. They are a successful East Coast-based firm that has an aggressive management profile. They are currently in the process of acquiring a West Coast competitor - let's call it Yourflowers.com - that provides a similar service.

FlowerPwr.com has its own website: Its various enterprise applications are written in PHP. Yourflowers.com also has its own Website: However, all its applications are Java-based and were developed for J2EE. They have their own J2EE application server. FlowerPwr.com needs to begin operating as an integrated commercial entity as soon as possible, in a way that conceals the fact that the companies have merged.

Using the Java Bridge, FlowerPwr.com can create a common portal in PHP. The company can leave Java up and running and take full advantage of their acquisition's existing Java services. FlowerPwr.com can do this over an existing portal using PHP.

The following code example illustrates how the Java Bridge can apply to this business scenario and others like it:

Example:

```
<?
// EJB configuration for JBoss. Other servers may need other
settings.
// Note that CLASSPATH should contain these classes
$envt = array(
"java.naming.factory.initial" =>
```

```
"org.jnp.interfaces.NamingContextFactory",
"java.naming.factory.url.pkgs" =>
"org.jboss.naming:org.jnp.interfaces",
"java.naming.provider.url" => " jnp://yourflowers.com:1099");
$ctx = new Java("javax.naming.InitialContext", $envt);
// Try to find the object
$obj = $ctx->lookup("YourflowersBean");
// here we find an object - no error handling in this example
$rmi = new Java("javax.rmi.PortableRemoteObject");
$home = $rmi->narrow($obj, new
Java("com.yourflowers.StoreHome"));
$store = $home->create();
// add an order to the bean
$store->place_order($_GET['client_id'], $_GET['item_id']);
print "Order placed.<br>Current shopping cart: <br>";
// get shopping cart data from the bean
$cart = $store->get_cart($_GET['client_id']);
foreach($cart as $item) {
print "$item['name']: $item['count'] at $item['price']<br>\n";
}
// release the object
$store->remove();
?>
```

The example code can be understood as follows:

1. The code example is written in PHP and forms part of a PHP Web application.
2. The PHP application first initializes an operation with the EJB, located at a specific URL that has the name:"//yourflowers.com:1099."
3. The code then specifies the bean-YourflowersBean-that the application will look for.
4. Next, the bean object is returned from the EJB server.
5. The application then calls methods-in this case, the Java application includes two functions:

    - *place_order receiving two numbers* - client ID and the item ID to add to shopping cart
    - *get_cart receiving one number* - client ID and returning the list of the items placed in the shopping cart so far.

After script execution, the referenced class may be disposed.

Info Messages

Zend Server Community Edition displays different types of messages that are color coded according to their level of severity. The following list describes the four different options and what each color means:

Error Messages

Messages that are Red indicate that some kind of system error has occurred. If you receive a message like this follow the instructions in the message.

The recommended actions are:

- Follow the instructions in the message.
- If the message appeared after an action was performed - try to redo the last action (such as to click Save, Add etc.).
- Visit the Support Center - http://www.zend.com/en/support-center/
- Open a Support Ticket - Support
- Reinstall Zend Server Community Edition - Choosing Which Distribution to Install

Notices

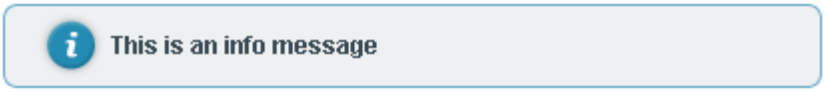
Messages that are Yellow indicate that a non-critical error occurred. If you receive a message like this it will contain information on how to proceed. This type of error includes messages to the user about usability issues.

Success Messages

Messages that are Green indicate the success of an action. If you receive a message like this it means that your last action was completed successfully and no additional actions are required (such as Restart Server).

Info Messages

Messages that are Blue indicate that there is an important message. If you receive a message like this, in most cases no action is required apart from reading the information.



For example:

```
Log file C:\Program Files\Zend\Apache2\logs\error.log does not exist or missing read permissions
```

When this Server Error Log Info Message is displayed, one of the following has occurred:

- No log files are available
- Files have been moved
- Permissions have been tampered with

Zend Server Best Practices

Welcome to the Zend Server Community Edition Best Practices Guide.

The following content is a collection of knowledge and information based on the experience of Zend's Development and Product Management team and the PHP community.

In this document, you will find reference information on the following development issues.

- The Performance section describes how to increase performance using Zend Server Community Edition .
- The Security section lists several additional security precautions you can take to secure your Zend Server Community Edition installation and Web application.
- The Development section includes instructions and tips for developers.
- The Deployment section describes the different deployment options (to remote servers, hosting, etc.) and how to go live with your Web application.
- The IIS Best Practices includes instructions and tips for configuring IIS on Windows.
- The Troubleshoot section includes solutions to known issues, possible problems and an error message reference.

If you have a tip or best practice that you would like to see here, please feel free to send it to documentation@zend.com.

Performance

In the Performance section, you will find information on how to configure and optimize Zend Server Community Edition and components to increase performance.

This document includes information on the following performance issues:

- Optimizing Zend Server Community Edition Performance - This section provides a description of each performance component and includes recommendations on when the component should be installed and for which conditions it should be disabled or removed.
- Optimizing Monitoring - This section provides suggestions on how to implement and configure the monitoring for production and development environments.
- Fine Tuning Optimizer+ - This section provides advanced settings to further enhance the performance gains achieved when Optimizer+ run out-of-the-box.
- Configuring PHP for Performance - This section explores the optimal php.ini configurations and settings to get the best PHP performance optimization.

Optimizing Zend Server Performance

The Zend Server Community Edition components are designed to encompass several different requirements. However, there is no point in adding or using certain components when they are not needed. This primarily happens when you install a component that you do not use. For example, if you do not need to call Java objects from your PHP code, there is no need to have the Java Bridge running. In addition, it would be better not to install this optional component at all, especially as you will be prompted to install a Java Runtime Environment that is not required if you are only running PHP.

In this section, we describe each performance component, including when you should install the component, when to disable the component and when applicable, when to remove the component.

Component	Description	Turn Off	Comment
Debugger	A remote debugging tool for developers working with Zend Studio.	Not recommended to turn off as it is great for development environments. In production when not debugging code	If you are not going to debug your code with the Debugger, for example in a production environment, disabling this component may provide a slight performance gain
Optimizer+	Speeds up PHP execution through opcode caching and optimization.	Disabling has a negative impact on performance.	
Guard Loader	Loads and runs encoded PHP scripts (Encoded with Zend Guard)	Required only if you are running PHP code that was encoded with Zend Guard.	If you are not a Zend Guard user either remove this component or do not install it (it is an optional component).
Data Cache	Cache data items or output	If you are not using the Data Cache API in your code for partial content caching.	
Java Bridge	Calls Java classes and code from PHP	Required only If you call Java code or objects from your PHP.	If you are not a Java user either remove this component or do not install it (it is an optional component).

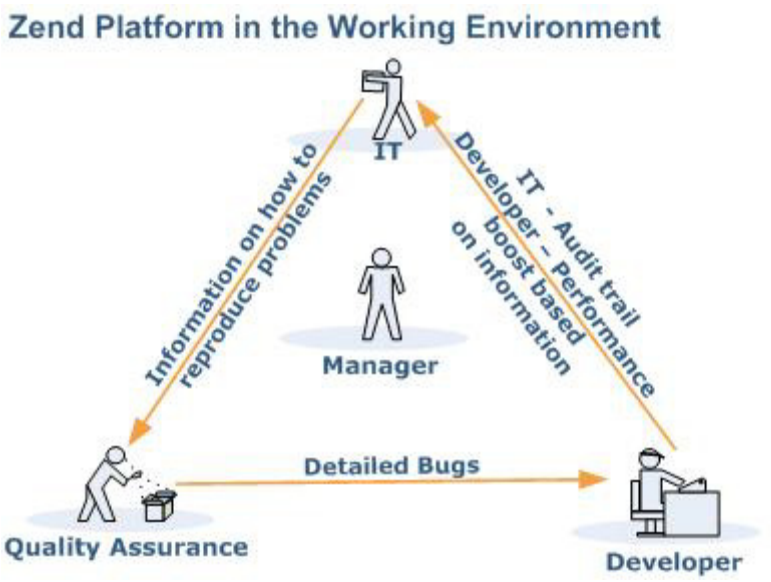
Component	Description	Turn Off	Comment
Monitor	Identifies performance issues	Turn off temporarily, only for performance testing reasons. Not recommended to remove this component however it is best to configure accordingly see " Working with Monitoring"	
Page Cache	A URL based HTML output cache for PHP scripts	Always If you are not using URL based Caching.	If you decide not to use this component.
ZDS (Zend Download Server)	Passing heavy download requests to a dedicated process to off load Apache	For testing reasons only. Or if you have a dedicated server for static content.	If you do not need to off-load large download traffic

Optimizing Monitoring

Developing and maintaining Web applications is an intricate and highly demanding process. Zend Server Community Edition facilitates the intricacies of the development process by employing an efficient problem resolution infrastructure. This infrastructure's main goal is to help make the most out of challenging environments and tight schedules and prevent problematic issues from falling between the cracks.

Using monitoring helps organizations improve communication between the development, testing and IT teams to streamline the development and deployment processes.

Development and production environments can unify the working environment and ensure improved information collection and distribution between development teams, testing teams and IT teams (See illustration below).



Using Zend Server Community Edition in your working environment ensures that pertinent and focused information reaches the right person at the right time. The enhanced information exchange results in major improvements in quality of code, time to production and overall performance and stability. The subsequent benefit is more resources dedicated to activities that focus on improving and expanding the current application and less time spent on locating the information that is necessary to recreate and resolve code and performance issues

The Monitor component assists the efforts of the development, testing and IT teams to quickly pinpoint, analyze, and resolve issues such as: PHP Slow Script Execution, Function Errors, Database Errors, etc.

Workflow:

- Implement customized Event Rules to areas prone to problems in your unique environment - facilitating focused and efficient problem resolution.
- Analyze "Full Problem Context" for a detailed insight of problematic occurrences.
- Integrate with Zend Studio to resolve problems with state-of-the-art development and debugging tools.

Implementing Monitoring

Implementing Monitoring is a process of defining Events according to acceptable runtime and performance parameters. When an Event occurs, the Monitor compiles a complete profile of the Event's occurrence and its precise details. The Event Details screen includes comprehensive details to enable developers and testers to recreate the Event in a way that mirrors the conditions of the original occurrence. This information can then be used to diagnose problems by fine-tuning Event rules to accommodate normal occurrences or resolve actual run-time problems and errors. The integration with Zend Studio makes it easy to diagnose problems and errors using the Debug Event and Profile Event options. In addition, problems in code can be immediately resolved using the Zend Studio Editor: The Zend Studio Editor makes it possible to both implement and deploy changes right away, not only to a single server, but also to all the nodes that belong to the same Group.

Code tracing provides an additional layer for analyzing

Events can be preserved to leave an indicator of these occurrences if necessary.

Configuring for Production or Development

In general, the best practice is the same: tune monitoring rules and thresholds to provide the information you need, without creating an overflow of events that you are not able to handle. This means that in development you may want focus on a specific rule type each time or set high thresholds and gradually modify them. In production it is preferred that you already come with an estimate of the thresholds that are necessary.

The difference between development and production is that usually in development environments you have to work very hard in order to have such an "overflow" - development environments are low traffic, low load systems. Additionally, the performance impact is negligible in development environment. In production, as a contrast, tuning is very important because of two reasons:

1. High traffic systems tend to generate hundreds and thousands of events per day if not properly tuned - even with aggregation, this tends to be more than what a development team can handle.
2. The more events you have, and the broader your thresholds are (for example the more functions you watch for Slow Function Execution events) the bigger the performance impact on your system is going to be. While under normal circumstances this impact is usually negligible, under high stress circumstances it could have an effect.

Given this, the best practice for tuning Zend Monitor thresholds is to start from relatively high thresholds, and lower them over time as old issues are fixed and the capacity for handling fine-grained errors grows. This is mostly true in production environments.

Fine Tuning Optimizer+

The performance improvement gained by letting the Optimizer+ run out-of-the-box can be further enhanced with fine tuning. These are advanced settings that need to be evaluated based on your environments usage specifications and performance requirements.

Note:

These are only recommendations, in most cases such fine tuning should not be necessary.

Disabling Code Change Auto-Detection

In the Administration Interface, to view the specific directives for Optimizer+, go to **Server Setup | Components** and click on the Directives link next to the Optimizer+.

Look for "***zend_optimizerplus.validate_timestamps***" and set the value to Off.

This speeds up the server, but also requires that you restart the server

( ) if you deploy new versions of existing files.

When to change: If your PHP code is rarely updated/changed or if you are capable of manually restarting your PHP on every code update.

When not to change: If you are in development and you are frequently changing code, or if you do not have control over the code update process.

Decreasing Code Validation Frequency

In the Administration Interface, to view the specific directives for Optimizer+, go to **Server Setup | Components** and click the Directives link next to the Optimizer+.

Look for "***zend_optimizerplus.revalidate_freq***" and set the value to 30 (seconds).Zend Server Community Edition is now set to check PHP file changes every 30 seconds.

When to change: If you do not change PHP files often and some delay between file update and site update is acceptable, you may set it even higher.

When not to change: If you have frequently changing files and you need the changes to take effect immediately.

Configuring PHP for Performance

You may be able to add an additional performance boost to your PHP applications by properly configuring your PHP runtime environment settings. You can edit the directives below from the Administration Interface via **Server Setup | Directives**.

Warning:

Changing some of these settings may cause certain PHP applications to stop functioning. Therefore, use discretion when you disable them and test your environment: It is important that you fully understand the purpose of each directive before you modify it.

Optimal php.ini configurations and settings for maximum performance optimization:

Name	Recommended Value	Zend Server Community Edition Default	Description
realpath_cache_size	256K	256K	Determines the size of the realpath cache to be used by PHP. This value should be increased on systems where PHP opens many files, to reflect the quantity of the file operations performed.
realpath_cache_ttl	120	120	Duration (in seconds) for which to cache realpath information for a given file or directory. For systems with rarely changing files, consider increasing the value.
error_reporting	E_ALL & ~E_NOTICE	E_ALL	The error_reporting() function sets the error_reporting directive at runtime. PHP has many levels of errors: Using this function sets the error level for the duration (runtime) of your script.

Name	Recommended Value	Zend Server Community Edition Default	Description
register_long_arrays	Off	Off	Tells PHP whether or not to register the deprecated long $HTTP_*_VARS type predefined variables. When On (default), long predefined PHP variables (like $HTTP_GET_VARS) are defined. If you're not using them, it's recommended to turn them off for performance reasons. Instead, use the superglobal arrays (like $_GET). This directive became available in PHP 5.0.0 and was dropped in PHP 6.0.0.
register_argc_argv	Off	Off	Tells PHP whether to declare the argv and argc variables (that contain the GET information).
magic_quotes_gpc	The default is: Off This feature is deprecated as of PHP 6.0.0.		Sets the magic_quotes state for GPC (Get/Post/Cookie) operations. When magic_quotes are On, all ' (single-quote), " (double quote), \ (backslash) and NULLs are escaped with a backslash automatically.

Name	Recommended Value	Zend Server Community Edition Default	Description
include_path	As short as possible, depending on the application's needs	".;/path/to/php/pear"	Specifies a list of directories where the require(), include(), fopen(), file(), readfile() and file_get_contents() functions look for files. The format is like the system's PATH environment variable: A list of directories separated with a colon in Unix or semicolon in Windows .

Name	Recommended Value	Zend Server Community Edition Default	Description
max_execution_time	30	30	This sets the maximum time (in seconds) that a script is allowed to run before it is terminated by PHP. This helps prevent poorly written scripts from tying up the server. The default setting is 30 s. When running PHP from the command line, the default setting is 0 s. The maximum execution time is not affected by system calls, stream operations, etc. See the set_time_limit() function for more details. You cannot change this setting with ini_set() when running in safe mode. The only workaround is to turn off safe mode or to change the time limit in the php.ini. Your Web server may have other timeout configurations that can also interrupt PHP execution. Apache has a Timeout directive and IIS has a CGI timeout function. Both default to 300 seconds. See your Web server documentation for specific details.

Name	Recommended Value	Zend Server Community Edition Default	Description
memory_limit	128M	128M	Sets the maximum amount of memory (in bytes) that a script can allocate. This helps prevent poorly written scripts from consuming all the available memory on a server. This setting can also be fine tuned during development to reach an optimal setting. When an integer is used, the value is measured in bytes. Note: To have no memory limit, set this directive to -1.
output_buffering	4096	4096	Allows you to buffer the PHP output instead of having it sent directly as soon as it is generated.

Security

In the Security section, you will find information on how to configure and optimize the Zend Server Community Edition and components to function more securely.

This document includes information on the following information:

- Allowed Hosts - This section describes the Allowed Hosts lists and offers recommendations on which hosts to add to the Allowed Hosts list for development and production environments.
- Securing the Administration Interface - This section provides information on how to set an IP address-based access control list on the Web server running the Administration Interface for the Windows, Linux and Mac OS X operating systems .
- Configuring PHP for Security - This section explores how you can add an additional security boost to your PHP applications by properly configuring your PHP runtime environment settings.
- Configuring Debugger Access Control - The how, when and why you should limit IP permissions.

Configuring Debugger Access Control

The allowed hosts list is a list of IP addresses that are permitted to initiate a Debugger session on the Web server on which Zend Server Community Edition is installed.
The default value for *zend_debugger.allow_hosts* intentionally covers a wide range of IP addresses. This is to make the initial installation of Zend Server Community Edition compatible for a large selection of environments.
However, **this also can be a security risk,** as you are permitting a wide range of IP addresses to access your Web server. Therefore, we recommend that you limit accessibility and create a secure environment by only using specific hosts (full IP address) recognized by you that you are sure you want to permit to connect.

To change this value in the Administration Interface, go to **Server Setup | Debugger**, remove all the IP range settings and set the specific IP's that you permit to connect to Zend Server Community Edition .

Depending on if you are working on a development or production environment, you may want to consider different defaults.
In development environments, all the machines that require access to debug should be allowed.
In production environments, it is safer to limit access or even allocate a single machine to allow access. Not only will this make your environment more secure, it may also help limit and prevent unnecessary traffic on your production server

Securing the Administration Interface

Purpose: To provide an additional security layer to the existing password protection – especially crucial to production environments.

Note:

This solution does not replace the appropriate firewall precautions you should take to deny access to the Administration Interface from certain IP addresses.

By default, access to the Administration Interface is password protected. If you want to secure access to the Administration Interface, you can do so by setting an IP address-based access control list on the Web server running the Administration Interface.
After following this procedure, users that try to access the Administration Interface from not-allowed (unauthorized) IP addresses are not able to access the Administration Interface.

Linux and Mac OS X:

The administration Interface runs on a dedicated lighttpd Web server. To secure access to the Administration Interface, edit your lighttpd configuration file in one of the following ways:

1. To only allow access from localhost, replace your lighttpd.conf with the pre-configured file called lighttpd.conf-localonly that is in the same directory.
2. To limit access to specific IP addresses, open your lighttpd.conf and add the IP addresses as follows:

   ```
   $HTTP["remoteip"] !~ "10.1.2.163|10.1.6.46|127.0.0.1" {
   $HTTP["url"] =~ "^/ZendServer/" { url.access-deny = ( "" ) } }
   ```

   This example shows how to allow access from 10.1.2.163, 10.1.6.46 and localhost and deny the rest.

   You can also do:

   ```
   $HTTP["remoteip"] !~ "10.1.2.163|10.1.6.*|127.0.0.1" {
   $HTTP["url"] =~ "^/ZendServer/" { url.access-deny = ( "" ) } }
   ```

   This means that you allow access from 10.1.2.163, 10.1.6.46, 127.0.0.1 (localhost) and hosts from 10.1.6.0 and deny the rest.

3. After applying the changes to your configurations, restart the lighttpd server with the command:
   # *<install_path>/bin/lighttpd.sh restart* or alternatively # *<install_path>/bin/zendctl.sh restart-lighttpd*

For additional resources and information on Lighttpd, see https://calomel.org/lighttpd.html .

Windows:

There are a few precautions you can take in order to secure your connection:

- Be secured using SSL connection - a certificate is needed by 3rd party vendors to enable encryption between client and server.
  All IIS versions (5,6,7) use this surf-safe mode.
- Use https connection which enables encryption.
- Configure your Username and Password using 7-12 alpha-numeric numerals. Set your Password immediately after first-time installation.
- Protect your connection using Anti-Virus.
- Windows users should update their Microsoft Installation packs with the provided updates to avoid back-doors and loop-holes.

To limit IP access:

- Enter your Web server's configuration and define the IP addresses that should be enabled.
  Apache users should refer to the Apache documentation - http://httpd.apache.org/docs/2.2/howto/access.html - Access control by host
  For more information about IIS security-related topics, visit the following Microsoft Web site:

http://www.microsoft.com/technet/security/prodtech/IIS.mspx

Configuring PHP for Security

You may be able to add an additional security boost to your PHP applications by properly configuring your PHP runtime environment settings. You can edit the directives below from the Administration Interface by going to **Server Setup | Directives**.

Warning:

Changing some of these settings may cause certain PHP Applications to stop functioning. Therefore, use discretion while disabling them and test you environment - it is important that you fully understand the purpose of each directive before modifying it.

Optimal php.ini configurations and settings for maximum security protection from external threats:

Name	Default	Optimal Value	Description
disable_functions			This directive allows you to disable certain functions for security reasons. It takes on a comma-delimited list of function names. disable_functions is not affected by Safe Mode. This directive must be set in the php.ini file: For example, you cannot set this in httpd.conf.
disable_classes			This directive allows you to disable certain classes for security reasons. It takes on a comma-delimited list of class names. The disable_classes directive is not affected by Safe Mode. This directive must be set in php.ini: For example, you cannot set this in httpd.conf.
magic_qotes_gpc	0	0	Sets the magic_quotes state for GPC (Get/Post/Cookie) operations. When magic_quotes are on, all ' (single-quotes), " (double quotes), \ (backslash) and NULLs are escaped with a backslash, automatically.
allow_url_include	0	0	This option allows the use of URL-aware fopen wrappers with the following functions: include(), include_once(), require(), require_once(). Note: This setting requires that allow_url_fopen be set to On.

Name	Default	Optimal Value	Description
expose_php	1	0	Decides whether PHP may expose the fact that it is installed on the server (e.g., by adding its signature to the Web server header). It is no security threat in any way, but it makes it possible to determine whether you use PHP on your server or not.
display_errors	1	0	This determines whether errors should be printed to the screen as part of the output or if they should be hidden from the user. Value "stderr" sends the errors to stderr instead of stdout. The value is available as of PHP 5.2.4. In earlier versions, this directive was of type boolean. Note: This is a feature to support your development and should never be used on production systems (e.g., systems connected to the Internet). Note: Although display_errors may be set at runtime (with ini_set()), it won't have any affect if the script has fatal errors. This is because the desired runtime action does not get executed.
register_globals	0	0	Whether or not to register the EGPCS (Environment, GET, POST, Cookie, Server) variables as global variables. Relying on this feature is highly discouraged. Please read the security chapter in the PHP manual on Using register_globals for related information. Note: register_globals is affected by the variables_order directive.

Do you want to learn more about securing your PHP ?
Why don't you take a look at our Security Training.

Configuring Debugger Access Control

The allowed hosts list is a list of IP addresses that are permitted to initiate a Debugger session on the Web server on which Zend Server Community Edition is installed.
The default value for *zend_debugger.allow_hosts* intentionally covers a wide range of IP addresses. This is to make the initial installation of Zend Server Community Edition compatible for a large selection of environments.
However, **this also can be a security risk,** as you are permitting a wide range of IP addresses to access your Web server. Therefore, we recommend that you limit accessibility and create a secure environment by only using specific hosts (full IP address) recognized by you that you are sure you want to permit to connect.

To change this value in the Administration Interface, go to **Server Setup | Debugger**, remove all the IP range settings and set the specific IP's that you permit to connect to Zend Server Community Edition .

Depending on if you are working on a development or production environment, you may want to consider different defaults.
In development environments, all the machines that require access to debug should be allowed.
In production environments, it is safer to limit access or even allocate a single machine to allow access. Not only will this make your environment more secure, it may also help limit and prevent unnecessary traffic on your production server

Development

In the Development section, you will find information on how to use Zend Server Community Edition and components in development for efficient detection and diagnosis of issues.

This document includes information on the following development issues:

- Working with Zend Framework - This section explores the benefits of the Zend Framework pre-configured stack that includes all the system components for developing Web applications with PHP and how to load Zend Framework's classes in your scripts.
- Configuring Zend Framework - This section presents the advantages of port-based virtual hosts and describes how to configure Zend Server Community Edition to run Zend Framework projects in a development environment, using port-based virtual hosts.
- Debugging - This section offers suggestions on improving the debugging process.
- Profiling - This section describes how to detect bottlenecks in your application using the Profiler and Zend Server Community Edition.
- Advanced Diagnostics with Zend Server Community Edition - This section presents suggestions to help diagnose problems by event rules.

Working with Zend Framework

Zend Framework users who deploy Zend Server Community Edition will benefit from a pre-configured stack that includes all the system components for developing Web applications with PHP.

The Zend Framework files are placed in:

- **Windows**: <install_path>\share\ZendFramework
- **RPM, DEB, Tarball and MAC** : <install_path>/share/ZendFramework

Loading Zend Framework Classes

There are two ways to load Zend Framework's classes in your script:

1. Using the Zend Loader:

The Zend Loader utility class checks whether the class already exists within the script. If it does, it will create the relevant file from the class name using Zend Framework's naming convention (See http://framework.zend.com/manual/en/coding-standard.naming-conventions.html for more information on Zend Framework's naming conventions). If the class already exists, this will speed up performance.
Using the Zend Loader also has the added advantage of loading classes outside of Zend Framework.

To use the Zend Loader:

1. Load the Zend Loader utility class once in your script:
   ```
   Require_once 'Zend/Loader.php';
   ```
2. From now, load each class using the class name:
   ```
   Zend_Loader::loadClass('Zend_Class_Name');
   ```
3. For example, in order to load the Zend Http Client:
   ```
   Zend_Loader::loadClass('Zend_Http_Client);
   ```

2. Using require / include calls

Classes can also be called using the conventional require or include calls:

To use 'require class':

1. Enter a 'require' command for the relevant file into your script:
   ```
   Require 'File.php';
   ```
2. For example, to require the Zend Http Client Class:
   ```
   require 'Zend/Http/client.php';
   ```

In order to see a full list of Zend Framework's components, including more information on the functionality and use of the various components, see http://framework.zend.com/manual

Configuring Zend Framework

Configuring Zend Server Community Edition to Run a Zend Framework Application

The following procedure describes how to configure Zend Server Community Edition to run Zend Framework projects in a development environment, using port-based virtual hosts. The advantage of port-based virtual hosts is in the ease of running several isolated applications on the same Web server. This overall solution allows developers working on a Zend Framework project in Zend Studio to immediately test any code changes locally.


The following procedure uses instructions suitable for Zend Studio for Eclipse and the Apache bundled with Zend Server. A similar procedure with some modifications can apply for other IDEs and web servers.

To configure Zend Server Community Edition to run a Zend Framework application:

1. Create a new Zend Framework project.
   If you have not already done so, create a new Zend Framework project using the New Zend Framework Wizard in Zend Studio for Eclipse.
2. Define a virtual host on Zend Server Community Edition that will point to the new project's public directory:
   a. Find the full path to your project's *public* directory.
      In Zend Studio for Eclipse, go to the project browser and right-click on the public directory from the menu choose Properties. The full path is listed in the Resource Tab's location field.
   b. Open your Apache configuration file (in most cases it will be httpd.conf and located in your Apache installation directory).
      Where is my Apache configuration file?
   c. Go to the end of the file and add the following section:

```
Listen 10089
< VirtualHost *:10089>
    DocumentRoot " DOCUMENT_ROOT"
    <Directory "DOCUMENT_ROOT">
        Order allow,deny
        Allow from all
    AllowOverride all
```

```
</Directory>
</VirtualHost>
```

3. Replace "DOCUMENT_ROOT" with the full path to the *public* directory, enclosed in double quotes ("DOCUMENT_ROOT")
   Replace the port number with a unique port number dedicated to this Virtual Host. The port number (10089) has to be the same value for "Listen" and "VirtualHost".
4. Zend Framework's MVC implementation makes use of the Front Controller pattern. You must therefore rewrite all incoming requests (except those for static resources, which your application need not handle) to a single script that will initialize the FrontController and route the request. If you're using mod_rewrite for the Apache web server, create the file <Project_Name>/public/.htaccess with the following contents:

```
# public/.htaccess
RewriteEngine On
RewriteCond %{REQUEST_FILENAME} -s [OR]
RewriteCond %{REQUEST_FILENAME} -l [OR]
RewriteCond %{REQUEST_FILENAME} -d
RewriteRule ^.*$ - [NC,L]
RewriteRule ^.*$ /index.php [NC,L]
```

Note:

Some web servers may ignore .htaccess files unless otherwise configured. Make sure that your web server is configured to read the .htaccess file in your public directory.

5. Restart your Web server from the command line (windows user can use the Apache Monitor tool) .

Your Zend Framework projects will now be accessible from a browser through: http://localhost:10089/ (the port number 10089 should be replaced with the unique port you dedicated to this virtual host).

Where is My Apache Configuration File?

Apache uses a main configuration file for all its settings, typically this file is called httpd.conf or apache2.conf. The location of this file varies depending on your installation:

- **Windows**:
  <install_dir>\Apache2.2\conf\httpd.conf
  If you changed the location of your Zend Server Community Edition installation, your document root will be located at <installation_directory>\ Apache2.2\conf\httod.conf, where <installation_directory> is the location of the directory in which Zend Server Community Edition is installed.
- **Linux**:
  If you installed Zend Server Community Edition from a repository (DEB or RPM packages), the location of your configuration file is defined by your distribution's Apache packages, and will vary depending on your distribution and configuration.
  Common locations include:
  - Debian / Ubuntu - /etc/apache2/apache2.conf
  - Fedora Core / RHEL / CentOS - /etc/httpd/httpd.conf

  If you installed Zend Server Community Edition using the generic Tarball package - /usr/local/ zend /apache2/conf/httpd.conf.
  If you changed the location of your Zend Server Community Edition installation, your document root will be located at <installation_directory>/ apache2/conf/httpd.conf, where <installation_directory> is the location of the directory in which Zend Server Community Edition is installed.
- **Mac**:
  /usr/local/zend/apache2/conf/httpd.conf

Deployment to Production

In the Deployment to Production section, you will find information on how to deploy code that runs on Zend Server Community Edition.

This document includes information on :

- Deploying Code with Zend Server Community Edition - This section presents suggestions on how to best deploy your PHP code to run with Zend Server Community Edition for production and development environments.

Deploying Code with Zend Server

This procedure describes how to deploy your PHP code to run with Zend Server.
Zend Server provides all the components for creating an environment suitable for developing and deploying PHP applications.
In order for a PHP Application to run you need a Web server. Apache is bundled by default with Zend Server and is used to run your PHP code. This option may vary depending on your operating system, for instance, MS Windows also supports an existing IIS installation so you can choose either Apache or IIS and in Mac, Zend Server uses the distribution's Apache.

The process of writing PHP applications is separated into two distinct sections: Development and Production.

- Development includes developing and debugging your code. In most cases this is done on a developers machine or on a remote server with limited or password protected access.
- Production is when the Web application has reached a level of maturity that allows it to be exposed to its target audience. The only tasks that should be done are debugging (remote) and uploading changes. It is against best practices to make changes to code running on a Production server and the preferred method is to use FTP/SFTP to upload changes.

Development

In order to run a PHP application, your PHP files must be placed in a specific location that indicates to the Web server what files to service.
When you are ready to run your PHP code on a Web server, place the files under the following directory according to your operating system and preferences:
Windows:

- Apache: <install_dir>\Apache2\htdocs
- IIS: C:\inetpub\wwwroot

Mac and Tarball:

- <install_path>/apache2/htdocs

DEB:

- The distribution's default location is: /var/www

RPM:

- The distribution's default location is: /var/www/html

Running the Code/Application

Open a browser and enter the URL: http://localhost: /<yourPHPfile>.php
Replace <port number> with the port you are using. The defaults are port: 80 (for Windows) and port: 10088 (for the other operating systems), unless you changed the port by preference.
Replace <yourPHPfile>.php with name of the file you want to access/run.

Note:
Remember to use the port name according to the port number you defined.

 To find out how to locally debug your code once its deployed in a Web server, see Working with Local Debugging.

Production

Deploying code to production is different than running your application in a controlled environment (such as a local server). Production basically means publishing your application to the internet.

So where do you publish your application?
Depending on the resources available to you, you either have a different server that is dedicated to servicing the web or a cluster of servers that are managed with a load balancer. In both cases a firewall or some other protection is necessary.
An additional option is to have your application run from a Web Hosting company.

Once your code is in its dedicated location, you will have to support the code so you will need to establish a way to upload files for purposes of issuing updates and fixing bugs or security threats. At this point if you have been locally debugging your code with Zend Studio you can now change your settings to remote debugging, if there is a firewall between you and your application's files you will need to use tunneling in order to debug through a firewall. Zend Studio users can also benefit from Remote Server support for uploading and synchronize your code.

IIS Best Practices

In the IIS Best Practices section, you will find information on how to configure and optimize IIS and to increase performance.

This document includes information on the following information:

- IIS Configuration Optimization - Tuning adjustment to optimize the FastCGI configuration for IIS6 and IIS7.
- Configuring IIS Timeouts

IIS Configuration Optimization

Note:

When moving from Zend Core to Zend Server Community Edition on IIS Microsoft's FastCGI is used instead of the Zend's FastCGI therfore the settings and configurations are in a different location. For more information per IIS version see below.

Tuning FastCGI Configuration for IIS6

Note:

These performance enhancements are defined by default when you install Zend Server Community Edition .

By default, Zend Server Community Edition runs with a maximum of ten concurrent PHP instances. For high load Web servers, it is recommended to increase this value, based on your performance requirements and other hardware/software limitations (such as memory, CPU, etc.).

To control the maximum amount of concurrent PHP instances:

1. Go to C:\WINDOWS\system32\inetsrv\fcgiext.ini.
2. Locate the entry for "php" under Types.
3. Locate the section corresponding to this entry (usually under "[PHP]").
4. Append the following line at the end of this section:
   MaxInstances=10

This will enable Zend Server Community Edition to run ten PHP instances, for high loads. If you have lots of memory and high loads, you can increase this value even more.

To control the amount of requests handled by a single PHP instance before recycling:

1. Go to C:\WINDOWS\system32\inetsrv\fcgiext.ini.
2. Locate the entry for "php" under Types.
3. Locate the section corresponding to this entry (usually under "[PHP]").
4. Append the following line at the end of this section:
   InstanceMaxRequests=10000

This will allow a single PHP instance to handle 10,000 requests, instead of the default 1,000. If you set this number higher, make sure you increase the value of *PHP_FCGI_MAX_REQUESTS* located in the same file accordingly.

Tuning FastCGI Configuration for IIS7

Note:

These performance enhancements are defined by default when installing Zend Server Community Edition .

By default, Zend Server Community Edition runs with a maximum of ten concurrent PHP instances. For high load Web servers, it is recommended to increase this value, based on your performance requirements and other hardware/software limitations (such as memory, CPU, etc.).

Requirements: IIS7 Resource Kit -

(x86) http://www.iis.net/downloads/default.aspx?tabid=34&i=1682&g=6

(x64) http://www.iis.net/downloads/default.aspx?tabid=34&i=1683&g=6

Once installed, you can administer your FastCGi settings from the Internet Information Services (IIS) Manager.

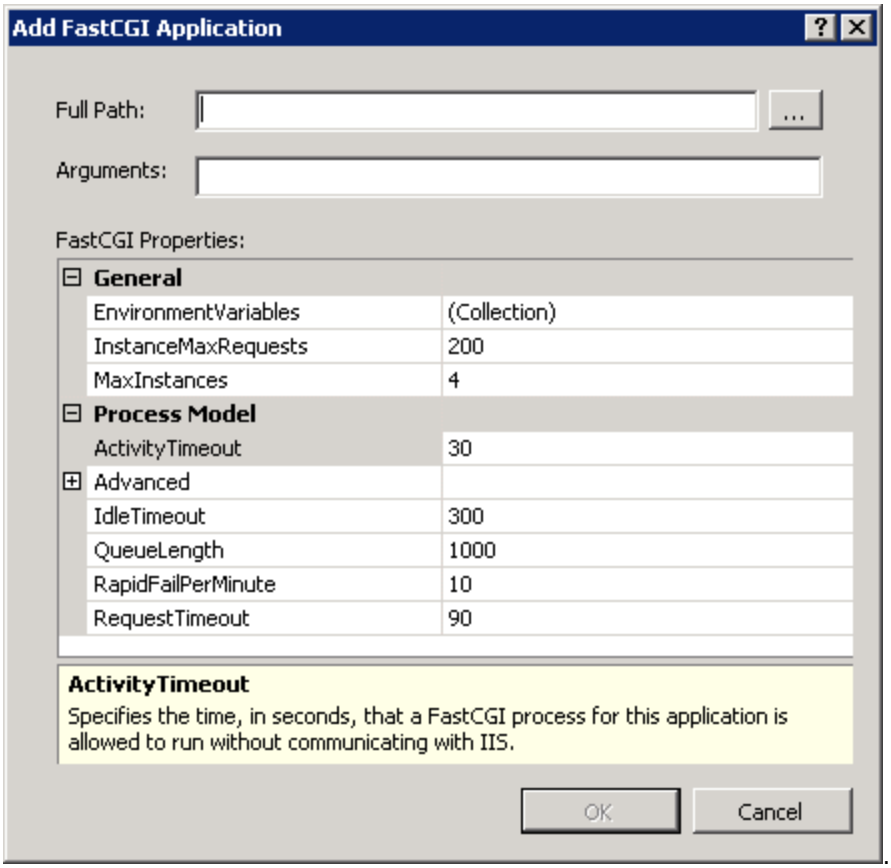
From here, you can configure your *MaxInstances* and *InstanceMaxRequests.*

To tune FastCGi configuration for IIS7:

1. Go to **Start | All Programs | Administrative Tools | Internet Information Services 7 - Application Server Manager**.
2. Select the server to manage from the left tree.
3. Click 

and select <install_dir>\bin\php-cgi.exe.
4. In the Actions section (on the right), click "Add Application..."

   The Add FastCGI Application dialog opens:



5. Tweak the variables as necessary.
   The recommended Zend default is *MaxInstances=10* and *InstanceMaxRequests=10000.*

Depending on which settings you change, the Web server's memory and CPU consumption are adjusted.

Configuring IIS Timeouts

The following instructions are intended for running Zend Server Community Edition with PHP FastCGI on Windows.

Issue:

The default timeout settings for FastCGI, may cause runtime failures for scripts that run longer than 30 seconds.

Resolution:

If you know that you have scripts that run more than 30 seconds set your FastCgi and PHP to a longer script timeout duration.

FastCgi Settings:

This procedure describes how to change your FastCgi timeout settings according to webserver type and version.

- **Apache 32bit**:
  Open C:\Program Files\Zend\ZendServer\etc and in **ZendEnablerConf.xml** the defaults should be changed to <Timeouts ***connectionTimeout***="<Number of Seconds>" ***requestTimeout***="<Number of Seconds>" />
- **Apache 64bit**:
  Open C:\Program Files (x86)\Zend\ZendServer\etc and in **ZendEnablerConf.xml** the defaults should be changed to <Timeouts ***connectionTimeout***="<Number of Seconds>" ***requestTimeout***="<Number of Seconds>" />
- **IIS 7**:
  In applicationHost.config locate the following:

```
<fastCgi>
  <application fullPath="C:\Program Files (x86)\Zend\ZendServer\bin\php-cgi.exe" maxInstances="10" instanceMaxRequests="10000" >
    <environmentVariables>
      <environmentVariable name="PHPRC" value="C:\Program Files (x86)\Zend\ZendServer\etc" />
      <environmentVariable name="PHP_FCGI_MAX_REQUESTS" value="10000" />
    </environmentVariables>
  </application>
```

</fastCgi>

- And change the following values:

  activityTimeout="<Number of Seconds>"

  requestTimeout="<Number of Seconds>"

PHP Settings

This procedure describes how to configure your PHP's execution time.

To configure your PHP's execution time:

1. In Zend Server Community Edition go to Server Setup | Directives
2. Edit the value of the following directives:
3. Change ***max_execution_time*** to <Number of Seconds> and ***max_input_time*** to<Number of Seconds>
4. **Restart** PHP

Scripts that run more than 30 seconds but less than <Number of Seconds> should now run. See below for instructions on how to test this.

Testing the Changes

The following procedure shows how to run a short script that checks if the settings have been properly applied.

To test your settings:

1. Open a text editor and insert the following code:

```
<?php
sleep(40);
echo "If you see this text the script completed and the defaults were changed";.
?>
```

2. Run the script from your docroot, if the script suceeded to run you will see the following message in your browser "If you see this text the script completed and the defaults were changed"

If the test failed you will not see a message in your browser. In that case try restarting your webserver and running the script again.

Troubleshoot

Welcome to the Zend Server Community Edition Troubleshoot section. The following content is a collection of knowledge and information based on the experience of Zend's Development and Support teams and the PHP community.

In the Troubleshoot section, you will find solutions to known issues, possible problems and an error message reference. If you encounter any of these issues while working with Zend Server, this information can help you resolve the matter quickly to enable you to return to your normal workflow.

This document includes information on the following issues:

All operating systems

- Zend Server Exception Caught - When the port settings are not configured as expected

Windows only

- Windows: Zend Server isn't Running Out of The Box - You've installed Zend Server successfully, but an error message is displayed in the browser when you click the short cut.
- Windows: Zend Server not Loading - Zend Server or a related process causes unexpected system behavior
- Windows: Internet Explorer Blocking Zend Server - IE7 running on Windows 2008 Server blocks Zend Server and prompts you to add its URL to the Trusted Zone.
- Windows: IIS URL Rewrite Setup - Recommendations on which URL rewrite engine to use and where to download from.

Linux and Windows

- Support Tool - Your opportunity to enable the Support team to provide better service by allowing us to gather server configuration and setup information.

Zend Server Exception Caught

Installing Zend Server Community Edition with a bundled Apache assumes that the following port settings are used: The Web server (Apache) is listening on port 10088; and the Zend Server Community Edition Administration Interface are listening on 10081,10082 . If your environment is configured differently, when trying to access the Administration Interface you will receive a "**Zend Server Exception Caught**" error message.

Note:

DEB and RPM installations do not need to listen to port 10088 because the Apache's distribution is used.

To fix this, the port settings must be changed.

To set the Administration Interface's settings to listen to a different Web server port:

After changing your Apache's port setting to another port:

1. Change the Administration Interface's port setting as follows:
   Go to <install_path>/gui/application/data
2. Open the file zend-server.ini. In the section called "userServer", set the URL to the new port number.
3. Restart Apache.

The different installation options set different Apache configuration file locations as follows:

- DEB Apache conf file: /etc/apache2/apache2.conf
- RPM Apache conf file: /etc/httpd/conf/httpd.conf
- Tarball Apache conf file: <install_path>/apache2/conf/httpd.conf
- Mac Apache conf file: /usr/local/zend/apache2/conf/httpd.conf

Windows: Zend Server isn't Running Out of The Box

This item refers to Windows OS using IIS (5-7)

After installing Zend Server Community Edition , clicking on the shortcut opens the browser with an error.

Possible cause: It could be that your Web site is not running

Solution: Turn on your Web site

To turn on your Web site:

1. Go to My Computer
2. Right-click and from the menu select Manage
   The management Console is displayed.
3. In the navigation tree locate the node "Internet Information Services"
4. Under this node is a list of Web sites, make sure that the Web site Zend Server Community Edition is associated with is running.
   If it is not running there will be a red indicator on the folder.
5. To set the Web site to run, right-click on the folder and set to start.
   Try to run Zend Server Community Edition again.

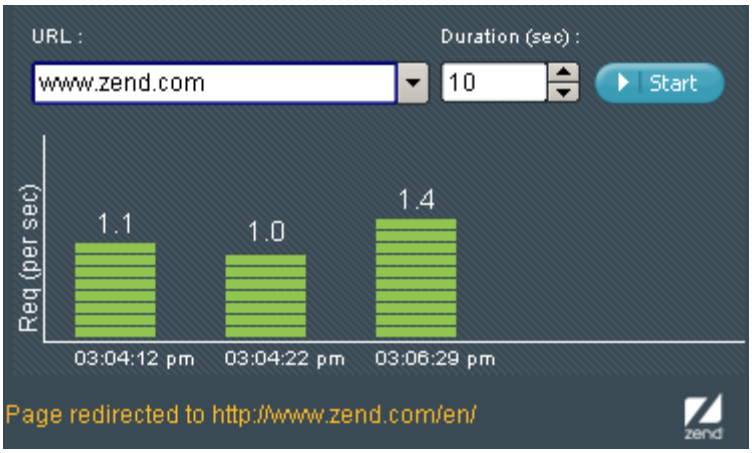
If this did not solve the problem more information can be found in the Zend Support Center: http://www.zend.com/en/support-center/.

Supported Web sites:

IIS5 users will only have one Web site. Whereas, IIS6 and IIS7 support multiple Web sites. When activating a Web site, make sure that you are activating the appropriate Web site (the site that was selected in the installation process).

Zend Controller Cannot Run Benchmark

The following message may appear after you enter a URL into the Zend Controller's benchmark: "Page redirected to ..."



This means that the URL that you entered is not the "exact" URL or is being redirected for some reason. In order to run the test, specify an exact URL or use the suggested address and click **Start** again.

Zend Controller Cannot Login

After installing Zend Server Community Edition you try to run the Zend Controller and a message is displayed in the Zend Controller stating that it cannot log in.

Possible causes:

1. You have not yet logged in to Zend Server Community Edition for the first time and therefore your password has not been defined.
   Log in to Zend Server Community Edition and set your password.
2. The password setting is incorrect.
   Open the Zend Controller settings menu, right click on  and select **Settings** from the menu. Reenter your password in the Password field.
3. Your port number is incorrect.
   Open the Zend Controller settings menu, right click on  and select **Settings** from the menu. Make sure the port number is correct (same as in the URL for opening Zend Server Community Edition .

Windows: Zend Server not Loading


This Item is only relevant for Windows.

If for any reason, you cannot load Zend Server Community Edition or one of the Zend Server related process causes a crash or unexpected system behavior, use the installer in Repair mode.

To run the installer in repair mode:

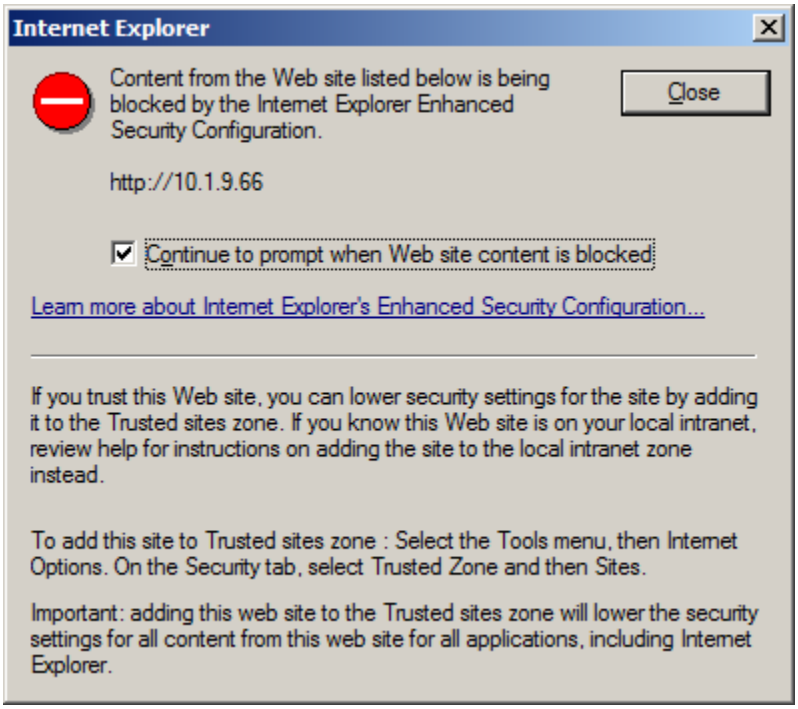
1. Run the installer file or go to Start | Control Panel | Add or Remove Programs | Zend Server and select Modify to run the installer
2. Click next to complete the repair process and Finish to close the Installer

You should now be able to run Zend Server Community Edition . If you are still encountering problems, check out our Support Center at: http://www.zend.com/en/support-center

Windows: Internet Explorer Blocking Zend Server

This item is relevant for Internet Explorer 7 running on Windows 2008 Server.

After installing Zend Server Community Edition for the first time, you may encounter an Internet Explorer system message stating that Zend Server Community Edition was blocked (see image below).



This is a security message prompting you to add Zend Server Community Edition to the trusted sites zone.

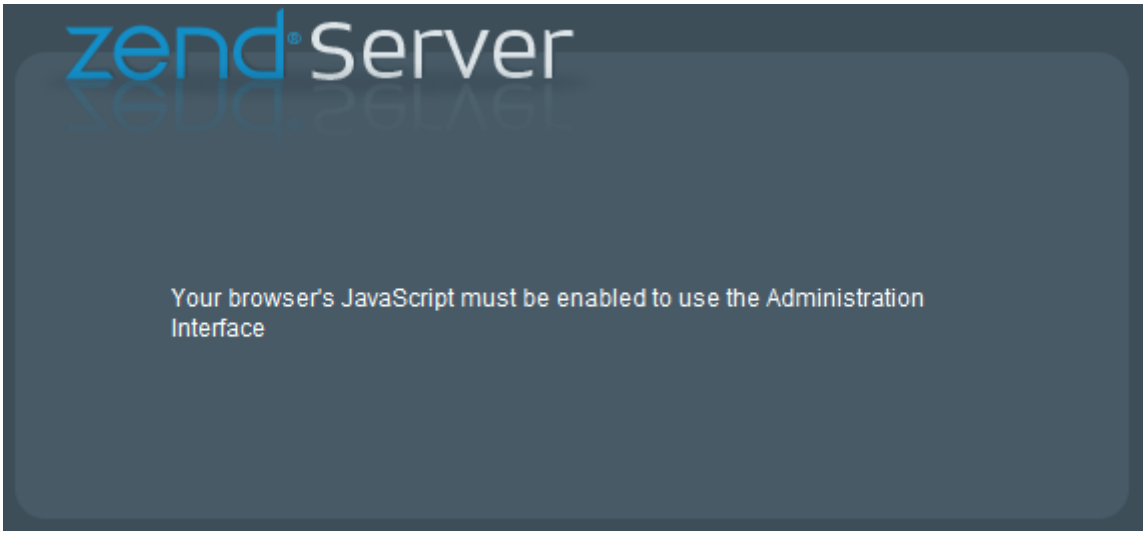
This procedure describes how to add Zend Server Community Edition to the trusted sites zone in Internet Explorer 7 running on Windows 2008 Server.

To add a Web site to the Trusted sites zone:

1. Go to **Tools | Internet Options.**
2. Click to display the **Security** tab.
3. Select "**Trusted Zone**" and then **Sites.**
4. Click **Add** to include Zend Server Community Edition as a trusted site.
5. Click **Close** and then **OK** to save the changes and close the dialog.

Zend Server Community Edition will now be added as a trusted site and the message will not appear.

Depending on your security settings, you may only see the following message:



This also indicates that Zend Server Community Edition is not a trusted site. As soon as the site is added to the trusted zone, this message is no longer displayed.

Windows: IIS URL Rewrite Setup

A rewrite engine does not come standard with IIS. If you haven't done so already, you will have to download and install one.

There are several online resources that can help you set this up:

- Zend Framework users should see: http://framework.zend.com/wiki/display/ZFDEV/Configuring+Your+URL+Rewriter
- For Microsoft's URL rewrite module for IIS 7.0 see: http://learn.iis.net/page.aspx/460/using-url-rewrite-module/.

Changing the Component's Log Directory

This issue is intended for advanced users who want to change the directory for storing Zend component Log files.
By default, component logs are written to the directory specified in the directive zend.log_dir in the ZendGlobalDirectives.ini file located in *<install_path>/etc/conf.d/ZendGlobalDirectives.ini* .
If you change the path, the following components will write their logs to the new location:

- monitor.log
- monitor_node.log
- monitor_zdo.log
- page_cache.log

Linux

To Change the Log directory in Linux:

1. Create the new logs directory with write permissions in order to be able to write the logs in the new directory.
2. The new directory has to be owned by the Apache NOBODY user profile and belong to the file system group *zend.* To move the directory to the zend group run the following command as user *root*:

   ```
   chown -r [Apache-user]:zend [new directory]
   ```

3. Open *<install_path>/etc/conf.d/ZendGglobalDdirectives.ini* and change the value of zend.log_dir to the new log directory
4. Run zendctl.sh stop and zendctl.sh start to apply the changes, this script is located in *<install_path>/bin/*

Now the log files for the Zend Page Cache and Zend Monitor components will be written to the new location. This means that some log files such as Apache and PHP, will still be written to the default directory (<install_path>/var/log)

Windows

To Change the Log directory in Windows:

1. Create the new logs directory
2. Open <*install_path*>*etc**php.ini* and change the value of zend.log_dir to the new log directory
3. To apply changes manually restart your Web server (Apache or IIS)

Now the log files for the Zend Page Cache and Zend Monitor components will be written to the new location. This means that some log files such as Apache and PHP, will still be written to the default directory (<install_path>\logs).

Note

The new directory must have the same permissions as the original logs directory.

Support Tool

(only)

The Zend Support Tool gathers server configurations and setup information.

The gathered information is used to help Zend's support team to troubleshoot support issues and provide comprehensive and efficient support.

Collected Information

In general, the information collected is defined in the definition file. If, for security reasons, you do not want to disclose specific information, you can edit the file to not include that information. However, the more information the support team can access, the better the chance of quickly resolving support-related issues.

Linux

To run the support tool:

```
<install_path>/bin/bugreport.sh
```

The default location for saving the files is:
$TMPDIR/zend_server_report_bug_$TIMESTAMP.tar.gz
If *TMPDIR* is not defined, it results to */tmp*

The definition file is located in:
<install_path>/share/bugreport/files
This file contains the definitions for which files and directories to collect. Through this file you can also define the name that will be used to create the archive, in case you do not want to use the default name.
Example:
/etc/apache2/conf.d apache_conf.d
Means, take the contents of the entire */etc/apache2/conf.d* directory and rename it to *apache_conf.d <install_path>/share/bugreport/commands*
Defines which commands to run and include in the output.

Once a report is generated, you will see the following output:
Sample Output:

```
#  <install_path>/bin/bugreport.sh
The information was collected successfully.
Use free text to describe the issue in your own words.
To submit the information press CONTROL-D
Archive created at /tmp/zend_server_report_bug_123008052721.tar.gz
```

Windows

The Support Tool software may be found in: <install_path>\bin\SupportTool.exe.

1. Open the Support Tool from Start menu, Zend Server/Support Tool.
2. Select a directory to generate the archive file to (Desktop is default).
3. Click Create.
   A Zip file is created on the desktop of the current user. The file is created with a time stamp including date and time.

Supported Browsers

For optimal stability and performance, only run Zend Server Community Edition on a supported browser from the Supported Browser List.

The following table lists the browsers that run Zend Server Community Edition .

Browser	Supported Operating Systems
Internet Explorer 7.0	Windows XP and Windows Vista
Firefox 2.x	Linux, Windows XP, Windows Vista, OS X 10.4 and OS X 10.5
Firefox 3.x	Linux, Windows XP, Windows Vista, OS X 10.4 and OS X 10.5
Safari 2.x	OS X 10.4
Safari 3.x	OS X 10.4 and OS X 10.5

Note:

Zend Server Community Edition may run on other browsers but with unpredictable behavior.

Log File Permissions

When the message "Log file /usr/local/zend/var/log/error.log does not exist or missing read permissions" appears it means that Zend Server Community Edition does not have permissions to read the log file, or, the file does not exist. If the file does exist, you will need to provide the 'zend' user permissions to access the directory containing the file, and read the file itself.

One example of enabling Zend Server Community Edition to read the Apache error log on Debian Linux is provided below:

To enable Zend Server Community Edition to read the Apache error log on Debian Linux:

1. Open a terminal and switch to root using "su" or "sudo -s".
2. Run the following command:
   chmod 644 /usr/local/zend/var/log/error.log

Note

On most Red Hat, Fedora and CentOS systems you will need to allow access to the Apache logs directory too. This can be done by running the following command as root or using 'sudo': chmod 755 /var/log/httpd

Index