The PHP Company

Zend Server Cluster Manager 5.1 Reference Manual

By Zend Technologies

zend server
cluster manager
zend server cluster manager
Enterprise-ready Web Application Server

www.zend.com

Abstract

This is the Reference Manual for Zend Server Cluster Manager Version 5.1.

Zend Server Cluster Manager Reference Manual, issued February 2011.

DN: ZSCM-RM-140211-5.1-07

Table of Contents

Zend Server Cluster Manager Reference Manual

About

Zend Server Cluster Manager is the main management console for managing a cluster or servers running Zend Server.
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. However, a cluster can only be consisted of servers of the same type.

The PHP versions included with Zend Server 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.

To get started with Zend Server Cluster Manager, go to Getting Started with ZSCM.

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

Zend Server Cluster Manager Overview

Zend Server Cluster Manager provides high availability and scalability to PHP applications running on Zend Server. By managing a number of Zend Server instances running simultaneously, Zend Server Cluster Manager allows them to work together function as a powerful web application server while appearing as a single instance to clients. Zend Server Cluster Manager enables you to scale your business-critical applications and ensure they perform well under extreme load conditions.

Capabilities Include:

- **High Availability** – session clustering in Zend Server Cluster Manager provides a high availability fail-over solution for your application.
- **Scalability** – Zend Server Cluster Manager's flexibility allows the capacity of the application to be increased quickly without service interruption or changes in the environment
- **Improved Application Performance** – a centralized *job queue* provides built-in optimization for multi-node applications, ensuring high performance and low resource utilization.
- **Centralized Configuration and Management** – Zend Server Cluster Manager's management console allows you to easily add or remove nodes and confirm that their configuration and operation are in sync, etc.
- **Centralized Monitoring** – unified event monitoring in Zend Server Cluster Manager provides a single interface for managing monitoring rules and viewing *events* across all nodes in the cluster, removing the need to duplicate rule management and to receive duplicate event notifications.

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:
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

Getting Started with ZSCM

The following list describes the initial configuration tasks that you can do with Zend Server Cluster Manager. Some of these tasks depend on if you already have Zend Server installed on a server or servers or if you are building a cluster and starting with Zend Server Cluster Manager.

If you have not yet installed Zend Server Cluster Manager see Installing Zend Server Cluster Manager.

Once Zend Server Cluster Manager is installed you can start configuring your settings. If you are using settings from a previously configured Zend Server you may not have to make any changes to your settings although you may want to check your events to see that they are generating at optimal levels and if necessary modify event thresholds. See Optimizing Monitoring
for more information on the monitoring workflow.

If you have installed and created a cluster and you have not previously configured any settings you will be running on default settings. The default settings are as follows:

- Zend Monitor will be running and collecting events based on default thresholds.
- Zend Session Clustering is in standby mode on the servers.
- Zend Job Queue is running without any active jobs.
- Page Cache is loaded without any active caching rules.

For a complete list of components see Installed Components

Cluster Management

Cluster Management is a highly scalable solution for creating a clustered environment using Zend Server. A clustered environment is deployed when it is necessary to synchronize data across a cluster of three or more PHP servers. Moreover, PHP implementations that were not originally designed with scalability in mind, Zend Server to grow beyond a single server, to a Web Cluster.

Cluster Management includes:

- Session Clustering - The sharing of session information across a cluster of servers.
- Centralized Monitoring - aggregation of issues and *events* throughout the clustered environment.
- Server Management - A single point of management for server configurations (PHP, components etc).

Centralized Monitoring:

Zend Server is a central monitoring management console, the main intent of this is to provide a central area that will aggregate all information collected in your system such as Monitor events, Job information, Code Tracing information etc. In addition to collecting information, the centralized system is a means for propagating settings throughout the system such as monitor rule settings, *jobs* and also component, extension and directive configurations.

Information Display

When first logging in to Zend Server, There will be no information displayed in the various tabs. Zend Server will start to collect information only after you have added at least one server to the cluster. The only exception to this is if you had servers running as a cluster and then removed them. All historical information in the Dashboard tabs is retained such as old queue statistics, jobs and trace information. All subsequent information displayed in the Administration Interface represents the settings of the server defined as the master server. This is defined when the server was added by selecting the option "Configure all the cluster like this new server".

Additional Information about Cluster Management:

The Cluster Management administration interface page

Managing Servers- instructions on how to deploy setup and manage your clustered environment

The Cluster Management component - a general overview of the component's architecture.

Cluster Management API - an API reference for Cluster Management functions and classes.

High Availability

HA (High Availability) is a solution that uses the Session Clustering infrastructure to provide availability and continuity of mission critical business applications.

Session Clustering HA (High Availability), is an additional safety layer for maintaining session information integrity in Web cluster environments. HA ensures that sessions will be serviced in case of a single failure.

Session Clustering HA provides all the current Session Clustering functionality and is an optional feature for environments that require High Availability.

Note:

As an additional functionality layer, running HA may have a slight impact on performance in comparison to regular session clustering response time.

About High Availability

The High Availability layer preserves session information when a server fails. Each session is saved twice, once on the master (originating) Session Clustering Daemon and one on the master's backup Session Clustering Daemon. This means that in the event that a master server fails, requests are re-routed to the backup Session Clustering Daemon. Once a request is re-routed to the backup Session Clustering Daemon, the Backup Session Clustering Daemon turns into a master Session Clustering Daemon and creates a new backup Session Clustering Daemon. A backup Session Clustering Daemon is chosen based on it activity locating the Session Clustering Daemon with the least amount of sessions and open sockets. In the event that two servers fail session information will be preserved in the backup and can be replicated if requested in its lifetime.

Regarding response time, the HA layer will not impose more than a 10% performance degradation over the existing session clustering solution (number of session requests per second).

As mentioned earlier, all servers that need to share session information have to be associated to a cluster in Session Clustering Daemon. The HA layer is also capable of identifying when a fallen server has been recovered and will automatically return the fallen server into service.

Graceful Shutdown

Session Clustering's Graceful Shutdown allows removing a server from a cluster for administrative purposes (such as maintenance) without risking session data loss.

Graceful Shutdown works by transferring all the sessions stored on the server being removed from the cluster to one of the other servers in the cluster. After transferring all the sessions, the server will notify other cluster members that it has been replaced. Requests for sessions originally stored on the now terminated server will be routed to the replacement server.

It is highly recommended to perform graceful shutdowns when using Session Clustering, especially when shutting down more than one server, to avoid potential session data loss.

Important Note:

When a server is in the process of shutting down or is being started up after a graceful shutdown, it will not allow the session handler to create new sessions until this process is complete. You should always disconnect the server from the load balancing pool before removing it from the Zend Server Cluster Manager cluster, and only add it back to the pool after you have successfully added it to the ZSCM cluster.

To initiate a Graceful Shutdown, follow the following procedure:

1. Remove the server you intend to shut down from your load balancing pool. This should be done following the recommended procedure depending on your load balancing configuration.
2. Log in to the Zend Server Cluster Manager GUI and Navigate to **Cluster Setup | Servers**
3. Select the server you want to remove from the cluster, and click **"Remove"**
4. The server's status should indicate that it is shutting down. This process may take between a few seconds and a minute, depending on the number and size of sessions and on your network throughput.

Refresh the Servers list. If the server is no longer listed, it has been successfully removed.

If you plan to remove more than one server, repeat this process for each server after receiving confirmation that the previous server has been removed. Shutting servers down one by one is required to ensure information consistency. While a server is being removed you will not be able to add, edit or remove any other servers.

When adding a server which has been removed using Graceful Shutdown back to the cluster, make sure to only connect the server to your load balancing pool after the servers list indicate that the server is in “Ok” status.

Scalability

Zend Server Cluster Manager is equipped with a comprehensive solution for synchronizing session data across a cluster. Protect your applications from session corruption and erratic application behavior while providing an additional performance boost (up to x10). Immediately implement this solution to existing PHP code and attain linear scalability. Fully integrated with load balancers the Session Clustering module is a mechanism to ensure session data quality and integrity.

Java Bridge - The PHP/Java Bridge module provides PHP centric companies with a well-rounded environment making sure that the organization benefits from the ”best of both worlds”. Be it existing investments in J2EE application servers that require this solution, or to provide a means for organizations, if they choose, to bridge language limitations by use of Java applications. The Java Bridge is not limited to interactions strictly with J2EE and legacy systems, the Platform PHP/Java Bridge also provides the ability to interact with plain Java objects.

Application Performance

Zend Server Cluster Manager is equipped with three management modules for tracking and improving speed and responsiveness of Web applications. Optimizer+ Content Caching and Job Queues.

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.

Caching is the process of storing data or pre-rendered web output that can dramatically reduce the time to present results to the users.
The available caching capabilities are: Data Cache API, Zend Framework Cache API (External Link) and Page Cache.

Job Queues, provide PHP production environments with a standard approach to streamline offline processing. This provides the ability to reroute and delay the execution of processes that are not essential during user interaction with the Web Server.
Additional Information: Working with the Optimizer+, Working with Caching and Working with Jobs.

Centralized Configuration and Management

Zend Server Cluster Manager's architecture provides full control of the PHP application platform, including performance management settings, event thresholds, etc., allowing administrators to apply a consistent set of configurations throughout a cluster.

Centralized Monitoring

Zend Server Cluster Manager's Monitoring component detects and recovers crashes, whether they occur in PHP itself, the database software, or your own application. The integrated suite of monitoring, detection, code tracing and recovery features allows users to drill down to critical issues and optimizations quickly and easily.

An additional integration layer with Zend Studio allows you to analyze and fix issues directly to your project's code.

Working with Zend Server Cluster Manager

Administration Interface

The Zend Server Cluster Manager Administration Interface is very similar to the Zend Server Administration Interface. However, instead of being designed to manage a single server the Zend Server Cluster Manager Administration Interface allows you to perform configuration and management tasks for an entire cluster.

The Administration Interface layout is as follows:

- **Monitor Tab** - The Monitor tab is the main area for system information and it includes
  - **Dashboard** - The Dashboard page is accessed from **Monitor | Dashboard** and is the default page after logging in to the Administration Interface. In addition to the **Recent Events**, there is also a **Server Status** area for basic information on the servers that belong to your cluster.
  - ***Events*** - The Events page is the main display for events that are generated in your cluster. Events are based on the conditions defined in Monitoring Rules. You can use the Events page to perform additional actions to diagnose the problem. For more information on this page see Events.
  - ***Jobs*** - The Jobs page is the main display for jobs (their history and their status) that are scheduled in your cluster. For more information on this page see Jobs.
  - **Queue Statistics** - The Queue Statistics page is the main display for information regarding active *Jobs* that you have defined in your cluster. For more information on this page see Queue Statistics.
  - **Code Tracing** - The Code Tracing page is a central display and management area for all traced information. From this page you can access the code tracing tree and Statistics per trace file (generated by an event) or URL (manually generated from this page). For more information on this page see Code Tracing.
    Note:
    When you enter a URL the trace will be done through your load balancer to a random server, if you want to do directly to a specific server, specify the server's IP address.
- **Rule Management** - The Rule Management tab is the main area for configuring performance, monitoring and Queue features and it includes

  - **Monitoring** - Monitoring Rules page is a central area for defining and activating rules. Rules are used to or alert and collect information about PHP script problems. Each monitoring rule can also be assigned one of the following actions; Save code tracing and send email. Generated *events* are displayed in the Events page. For more information on this page see Monitoring.
  - **Caching** - The Caching page is the central configuration area for configuring caching rules for specific URLs or URL ranges. For more information on this page see Caching.
  - **Recurring Jobs** - The Recurring Jobs page is the central configuration area to configure jobs to run by URL. For more information on this page see Recurring Jobs
- **Cluster Setup** - The Setup tab is the main area for configuring your PHP and it includes
  - **Servers** - The Servers page is a central configuration page for creating a clustered environment that will be governed by Zend Server. For more information on this page see Servers.
  - **Components** - The Components page provides a convenient way to view and configure the components installed in your cluster. For more information on this page see Components.
  - **Extensions** - The PHP Extensions page provides a convenient way to view and configure extensions available in your cluster. For more information on this page see Extensions.
  - **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. For more information on this page see Directives.
  - **Debugger** - The Debugger page is used to enable remote PHP debugging and profiling of Web applications using the Zend Debugger component. For more information on this page see Debugger.
  - **Monitor** - The Monitor page is used to define the different settings for configuring the Zend Monitor component. This component is used to capture PHP *events* when they happen and to alert developers and system administrators. For more information on this page see Monitor.
  - **Session Clustering** - The Session Clustering page is used to define, the different settings for the Session Clustering component. This component is used to provide a highly scalable solution for synchronizing session data across a cluster of PHP servers. For more information on this page see Session Clustering.
  - *Job Queue* - The Job Queue page is used to define the different settings for configuring the Job Queue Component. This component is used to schedule *jobs* to run a PHP script. This can be done by creating a Job in Recurring Jobs or by using the API. For more information on this page see Job Queue.

- **Administration** - The Administration tab is the main area for configuring your Zend Server system settings and it includes
    - Password and License - The Password and License page is used to change your login password and update your license. For more information on this page see Licenses and Registration.

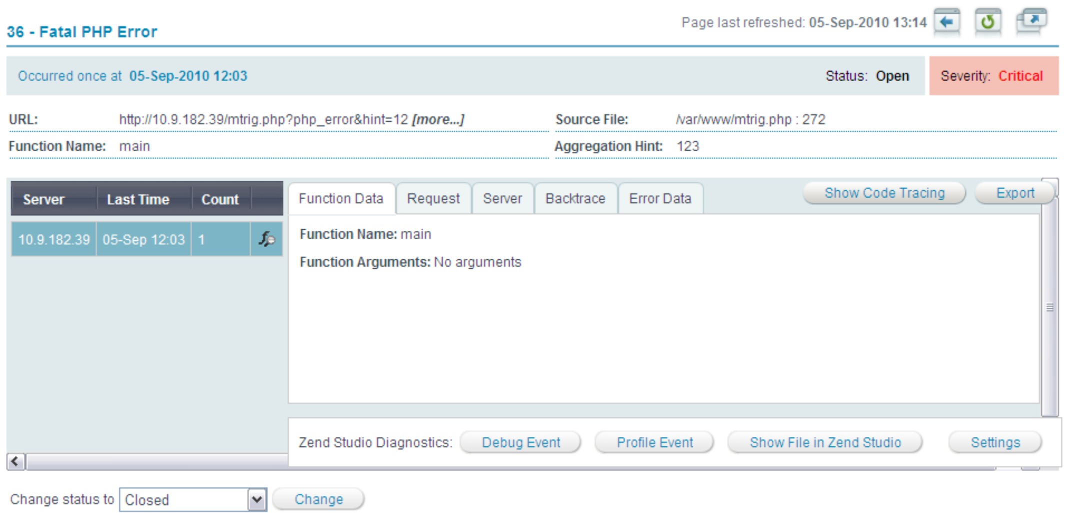
Working with Aggregated Events

The Zend Server Cluster Manager Event Details page, is accessed from **Monitor |** ***Events*** by selecting an event from the list and clicking on the row.

Aggregated events are events that are collected from all the servers that belong to the cluster. An event can occur on a single server or on some or all servers. the events details page helps to identify the source of the event and the server/s on which it happened.

The Event Details page is the main display area for aggregated information regarding the occurrence of a specific type of event in your cluster.

Information on how an event is triggered is presented in Event Rules.


36 - Fatal PHP Error

Page last refreshed: 05-Sep-2010 13:14

Occurred once at 05-Sep-2010 12:03 | Status: Open | Severity: Critical

URL:	http://10.9.182.39/mtrig.php?php_error&hint=12 *[more...]*	Source File:	/var/www/mtrig.php : 272
Function Name:	main	Aggregation Hint:	123

Server	Last Time	Count
10.9.182.39	05-Sep 12:03	1

Function Name: main

Function Arguments: No arguments

Note:

Not all events display the same information. Only information relevant to the specific event type will be shown.

The following actions can be performed from the Event Details page:

- **Back to Events** - Returns to the Events page.
- **Refresh** - Refreshes the report. Refreshing the report updates the event counter if the event occurred additional times.
- **Detach** - Opens the report in a new browser window.
- **Change Status** - Changes the status of the event displayed. For a complete explanation of event handling, see Working with Events.

On the Event Details page, users can view a general summary of an occurrence, its status and diagnose the occurrence with Zend Studio for Eclipse.

Event Detail - Information

The following list describes the information types displayed in the Event Details Page

Top Bar:

- **Number of Occurrences** - The accumulated amount of times the event was triggered between the first time the event occurred and the last time the event occurred. Refreshing the report updates this number if the event occurred additional times.
- **Status** - The status if the event: Open, Closed, Reopened and Ignored. For a specific event, the status can be changed in the Event Details page: For multiple events, the status can be changed in the Events page.
- **Severity** - The severity of the event (Warning or Critical). The severity is defined in the event's master settings in the Monitoring tab.

Event Details Table:

Events are aggregated into groups based on the time they occurred. The aggregation is set to five minutes: Thus, all the events that occur within that time frame are grouped together. Each time a set of events is aggregated (i.e., a new group is created), the occurrence details are collected again. To view the event occurrence details for a group, click on the group in the Event Details table: The display on the right will change to display the following options.

The Event Details Table options are:

- **Node** - The server on which the event occurred .If the node's status is "Unknown" the details you are viewing were collected from a node that no longer belongs to the cluster.
- **Last Time** - when the last event in this group occurred
- **Count** - the number of events triggered in the same time frame (set to five minutes)
- - an indicator that trace information was collected for that occurrence.

Clicking on one of these options updates the display with the relevant information.

Event Details

The Event Details are a collection of information relevant to the event that occurred. Clicking on a time in the table on the left (Event Details Table) will display the information relevant to the selected occurrence/es. Through this you can find out for example, if a different function caused the error or if a different message was thrown.

The following list presents the possible details that can be displayed for a specific occurrence:

- **Export -** generates a file containing the selected event's information.

- **General Data** - Displays information about the event: The data changes according to the event type.
- **Function Data** - Displays information on the function that triggered the error, including the function name and arguments.
- **Request** - Displays information about the request. The superglobals (POST, GET and COOKIE) are always displayed. The other parameters (RAWPOST and FILE) are displayed only when there is relevant information to display.
- **Server** - Displays the superglobals SERVER and ENV when there is relevant information.
- **Session** - Displays the superglobal SESSION when there is relevant information.
- **Backtrace -** Displays all the function calls that were made before the event was triggered, including the relevant files for each function.
- **Error Data** - Displays the PHP error and the Java backtrace if there was a Java exception.
- **Custom Variables** - Displays information for a custom event (i.e., class and user-defined information that was passed to the event when it was triggered).
- **Job Queu**e - *Job Queue* related events display the reason the Job generated an event.
- **Code Tracing** - Code Tracing related events display the reason that Code Tracing generated and event.
- **Zend Studio diagnostics -** Displays the actions can be applied to event details if Zend Studio for Eclipse (ZSE) is installed and Zend Server is configured to communicate with it:
  - **Debug Event** - Initiates a debug session for the event URL in ZSE.
  - **Profile Event** - Profiles the event URL, using the ZSE Profiler with the same parameters (GET, POST, COOKIE, HTTP headers, etc.).
  - **Show File in Zend Studio** - Opens the file where the event occurred in Zend Studio. This option makes it possible to use Zend Studio to edit files and implement changes for multiple servers.
- **Settings**: through this option you can choose to apply the Studio Integration actions to the Originating Server (the server on which the event was triggered) or to an Alternate Server (a different server running the same environment).Additional configuration settings are set in Server Setup | Monitor.

Managing Servers

To access the Server Management page, go to **Cluster Setup | Servers.**

from this page you can do the following:

- Add a Server
- Remove a Server
- Re-match your Cluster

Information Display

When first logging in to Zend Server, There will be no information displayed in the various tabs. Zend Server will start to collect information only after you have added at least one server to the cluster. The only exception to this is if you had servers running as a cluster and then removed them. All historical information in the Dashboard tabs is retained such as old queue statistics, jobs and trace information. All subsequent information displayed in the Administration Interface represents the settings of the server defined as the master server. This is defined when the server was added by selecting the option "Configure all the cluster like this new server".

Adding a Server

This procedure describes how to add a server to create a clustered environment that will be managed by Zend Server.

Once a server is added as a node you will no longer be able to login to the Zend Server Administration Interface that is running on the added server. All configuration and management actions will be done by Zend Server.

Before adding a node to the cluster, make sure that the server is running Session Clustering.

To add a server:

1. Go to **Cluster Setup | Servers**.
2. Click **Add Server**.
   The server configuration page will open
3. Enter the server's information as follows:

   - **Server Name** - a unique name for the server that will be used for identification purposes.
   - **Zend Server Address** -Complete the URL path to the Server you want to add by entering the server's IP address.
   - **Server Password** - the Administration Interface password you defined for accessing Zend Server.

   Note: If you do know what your password is, it may be because you installed Zend Server but never defined its password. To define the password, login to Zend Server and enter the password and license information.

   - **Configure all the cluster like this new server** - Select this option if you want to force-propogate the added server's settings throughout your cluster. If you do no select this option, the added server will inherit the Zend Server's settings.

The new server will be added to your cluster and will be shown in the Servers Status table in Monitor | Dashboard.

Note:
You can only add servers running Zend Server 5.0 and above. All servers must be running the same version.

Removing a Server

This procedure describes how to remove a server from your cluster

To remove a server:

1. Go to **Cluster Setup | Servers**.
2. Click **Remove**.

   A notification asking to confirm your decision will be displayed
3. Click **Yes** to remove the server

The server will still be listed in the table but will no longer belong to the cluster. If this server was the master server, the next server in line will become the master after the server was removed.

Once a server is removed as a node you will once again be able to login to the Zend Server Administration Interface that is running on the added server. All configuration and management actions will now be done by directly to the server.

Configurations: Even if there were different configurations on the server before it was added to the cluster, after removing a node, the settings that were given to the node as part of a cluster will stay the same even after it was removed. The one exception to this are the Zend Monitor settings which will be restored to the server after it was removed (if there were previous settings to restore, if not the Zend Server's settings will stay).

Session Clustering

The "Remove Server" action initiates a graceful shutdown process whereas all the active sessions will be rerouted to the other servers in the cluster. While the graceful shutdown is running the "Remove" option for removing additional servers will not be available. Once the shutdown is completed you will be able to remove more servers.

Force Remove

The force remove option is available when a server cannot for some reason be removed from the cluster. This is normally the case when trying to remove the last node from a cluster or when the regular removal hangs for more than a minute. Force Remove releases the user interface on the node's side without performing any shutdown actions that preserve information such as graceful shutdown for session information. The other nodes in the cluster will stop communicating with nodes that have been removed with Force Remove

Important Note:

Before using Force Remove try to access the node you are trying to remove and see if you can locate the reason it could not be removed properly.

Re-matching a Server

This procedure describes how to re-match a server. Re-match is the process of realigning the configurations on a given server that has different configurations. When this happens a notification will appear in **Cluster Setup | Servers** and a notice icon will appear next to the server's name.

Note:
Before re-matching a server check to see why the server's configurations changed i the first place.

To Rematch a server:

1. Go to **Cluster Setup | Servers**.
2. Click **Rematch Server**.

The Cluster manager's settings will be applied to the server.

Changing Cluster Configurations

Cluster configurations determine the behavior of all the Servers that were added as to Zend Server Cluster Manager. When changing a setting in Zend Server Cluster Manager such as a directive's value or loading an extension, once your PHP is restarted all the servers in the cluster will be modified accordingly.

Restart PHP Message

The Restart PHP message appears whenever a change is made to setting in your clusters php.ini file. in order to apply the settings click the "Restart PHP" button. The changes will be applied to the php.ini files on your nodes that are associated to this cluster.

If you are planning to make comprehensive changes and you are not sure what the outcome will be in terms of performance and behavior or you want to make changes to a cluster running a PHP application in production. The safer alternative is to apply the changes in a staging server, test it and then propagate the changes to the rest of the cluster. To do this you detach a server from the cluster, which will release the Administration Interface. Change the settings and then re-attach the server. When re-adding the server to the cluster use the option to propagate the server's settings to the rest of the cluster.

Changing Configurations from the Zend Server Cluster Manager Administration Interface

This procedure describes how to change your cluster's configurations. Any changes that you do in Zend Server Cluster Manager will be applied to all the servers in your cluster without exception. This procedure is more suited to changing configurations to an established cluster although its not mandatory. If you are in the middle of adding servers it is preferable to first add all the nodes and then modify the configuration once. Alternatively, you can add a single server first and change all the settings and only then add the additional servers and propagate that server's settings to the rest of the cluster as described in Adding_a_Server.

To change cluster configurations:

1. In the Administration Interface, edit your settings as necessary.
2. After making the first change a message will appear asking to restart the PHP. As long as you click save in each page you can ignore this message until you have completed customizing your configurations.
3. As soon as you have finished, click Restart PHP in any tab.

The changes will be applied to all the servers belonging to the cluster.

Servers

The Server page is accessed from **Cluster Setup | Servers.**

The ability to quickly and efficiently match the strength of the clustered application server to business demand is a huge plus from a business standpoint – for instance, rapidly add nodes to handle a spike in traffic to a web store during the holiday season.

The Servers page is a central configuration page for creating a clustered environment that will be governed by Zend Server Cluster Manager. Creating a clustered environment is a process of associating Webservers running the same application to a single point of management.

Note:
Once a server is added to your cluster the Administration Interface on the Server's side will be disabled and all configurations will be done using Zend Server Cluster Manager. Releasing the server from the cluster will also release the Administration Interface on the server's side.

To associate a cluster of servers under Zend Server Cluster Manager there are a few prerequisites:

- Each server needs to be running the same exact version of Zend Server.
- Each Server requires identical configurations.
- The servers in the cluster need to be running the same operating system however the server running the cluster manager can be different.

Once the servers are associated under Zend Server Cluster Manager, all the settings and configurations will be governed by Zend Server Cluster Manager. When adding a new server to the cluster created under Zend Server Cluster Manager you have two choices, to propagate the current settings of the added server or to change the added server's setting according to the current Zend Server Cluster Manager's settings (unless it's the first server you added).
If you choose to propagate the settings of the added server, all the servers currently belonging to the cluster will automatically be updated with the new servers settings. Once in a cluster environment, all the servers need to have exactly the same configurations in order to work smoothly.
If for some reason the master server is no longer available (removed, down, etc.) the next server in line will become the master server.

From the Servers page you can:

- **Add Server**: Add a server on which Zend Server is installed, once added the server will be managed and controlled by Zend Server.
- **To already added servers you can**:
    - **Remove -** completely removes the association with the Server. If in the future you want to add the server again you can add it as if it is a server that was added for the first time. When removing a server the Zend Server Cluster Manager settings will not be changed including the license even if there was another license there prior to the server being added to the cluster.
    - **Re-match Cluster -** realign the configurations on the server to the Zend Server's settings. This will overwrite any configurations on the Server.
    - **View Status -** if a server is not available or one of the server's settings has changed the server's status will change.
    - **Force Remove -** The force remove option is available when a server cannot for some reason be removed from the cluster. This is normally the case when trying to remove the last node from a cluster or when the regular removal hangs for more than a minute. Force Remove releases the user interface on the node's side without performing any shutdown actions that preserve information such as graceful shutdown for session information.

Important Note:
Before using Force Remove try to access the node you are trying to remove and see if you can locate the reason it could not be removed properly.

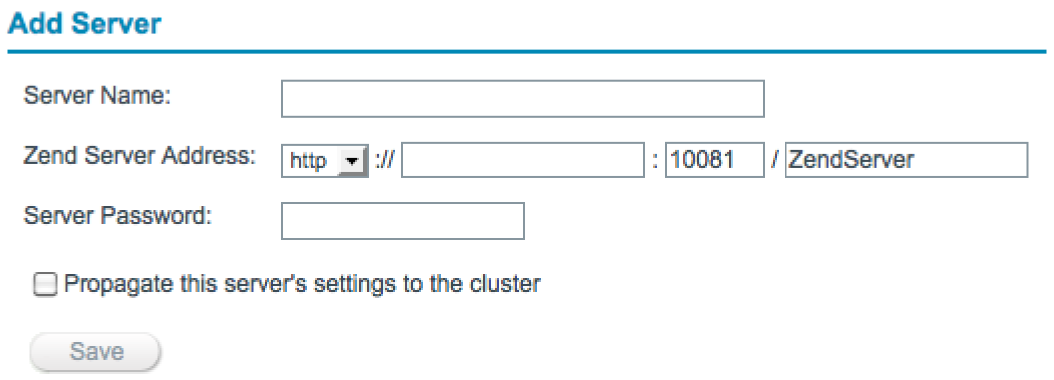
Add Server

Adding a server is the process of associating a machine that is running Zend Server to Zend Server Cluster Manager to establish a cluster environment. Once Zend Server is added to Zend Server Cluster Manager, you will no longer be able to directly access the Zend Server Administration Interface on that machine. All setting and configuration tasks will be done through Zend Server Cluster Manager and applied to all machines to maintain consistency throughout the cluster.

When adding a server all the settings and configurations of Zend Server Cluster Manager will be applied to the added server. the exception to this is the option to propagate settings, this option will inherit the Zend Server Settings and apply them to all the servers that have already been associated to Zend Server Cluster Manager and also to any subsequent servers that are added.

Note:
Once a server is added to your cluster the Administration Interface on the Server's side will be disabled and all configurations will be done using Zend Server Cluster Manager. Releasing the server from the cluster will also release the Administration Interface on the server's side.

The Add Server page is accessed from **Cluster Setup | Servers**, by clicking Add Server.


Add Server

Server Name:	
Zend Server Address:	http :// : 10081 / ZendServer
Server Password:	

☐ Propagate this server's settings to the cluster

Add Server Information:

- **Server Name**: a unique name for the server you want to add. This name is used for your identification purposes and therefore should be descriptive enough for you to identify what server it represents.
- **Zend Server Address**: the direct URL path to the server
- **Server Password**: the Zend Server administration interface login password that was defined when you first logged in to Zend Server after installing it for the first time. If you installed Zend Server on a server and you did not login at least once you will not be able to add this server to the cluster. To fix this go to the server, login and define your password - you will also at this point enter your license in order to activate Zend Server's functionality.
- **Configure all the cluster like this new node**: selecting this check box will update your cluster with the configurations on the server you are adding. These configurations will be propagated to all the nodes that have already been added to the cluster. If you don't want to propagate a specific server's setting and instead want the added server to accept the settings as defined in Zend Server - then, leave the check box unmarked.

Session Clustering

The Zend Server Cluster Manager, Session Clustering page is accessed from **Cluster Setup | Session Clustering**.

From this page you can define, the different settings for the Session Clustering component.
This component is used to provide a highly scalable solution for synchronizing session data across a cluster of PHP servers.

High-availability session clustering allows effective traffic balancing across your entire cluster, eliminating session loss in case of node failure and providing maximum uptime for your application.

From the Session Clustering page you can define the following settings:
Zend Session Clustering Settings:

- **Session Lifetime** - the duration for the session’s time-out limit.

Zend Session Network Settings:

- IP's allowed to communicate with the SCD (Session Clustering Daemon) - Zend Session Clustering settings provides a way to grant communication to the Session Clustering Daemon by server (for multiple servers you can use a Net Mask which implements Wildcards on IP addresses).

Notes:
*If your server is behind a firewall, ensure your broadcast ports are open so that your servers can broadcast to each other. The port that should be opened is defined in the mod_cluster.network.tcp_port_remote directive in your zend.ini file. The default port number is 34567.

*All servers in the cluster should be in the same subnet / broadcast domain.

Net Masks:

Net Masks are used to define a string of IP addresses using Wildcards '*' to specify the range of IPs that are either allowed or denied hosts. This option, allows to specify a range of IPs from 0-255 according to the selected amount of Wildcards for example if you choose to use the Net Mask option to grant communication to the following IPs: 24 (10.1.3. *) all IP addresses beginning with 10.1.3 will be granted access to the SC Daemon on this server.

Installing Zend Server Cluster Manager

Running the Zend Server Cluster Manager Installation

Installing Zend Server Cluster Manager

The following installation instructions refer to installing Zend Server Cluster Manager according to installation type (DEB, RPM and Windows). The instructions below state the installation command for complete information on installing and additional packages see each versions installation instructions in the Zend Server installation guide.

Important Note:
Zend Server Cluster Manager cannot be installed on a machine with an existing Zend Server installation.

DEB, RPM Automatic Installation Script

The following procedure describes how to run a script that will automatically create your DEB or RPM repositories and install Zend Server Cluster Manager.

1. Download the package called "Linux x86 Installer (RPM/DEB Setup Script)" from zend.com - http://www.zend.com/en/products/server-cluster-manager/downloads
2. Locate and extract the package:
   ZendServer-X.X.X-RepositioryInstaller-linux.tar.gz
3. To change to the directory with the installer scripts run:
   cd ZendServer-RepositoryInstaller-linux/
4. Run the following command:
   install.sh

After installing, a completion notification will appear, with a notice that Zend Serve Cluster Manager has been installed.
To access the Administration Interface (Web) open your browser at:
https://localhost:10082/ZendServerManager (secure) or http://localhost:10081/ZendServerManager.
Upon initial log in, you will be prompted to define your password.

DEB

The Deb installation method requires that you setup a repository before installing Zend Server Cluster Manager. For instructions on setting up a repository see Manually Installing Zend Server.
This method uses "aptitude" to handle the installations, upgrades and additional packages.

To install:

1.Once the repository is set up, run the appropriate command:

```
aptitude install zend-server-cluster-manager
```

The actual installation will require your conformation.

After installing, a completion notification will appear, with a notice that the servers have started.
To access the Administration Interface (Web) open your browser at:
https://localhost:10082/ZendServerManager (secure) or http://localhost:10081/ZendServerManager.
Upon initial log in, you will be prompted to define your password and enter your license information..
For information on how to upgrade your installation see Upgrading Zend Server.

RPM (RHEL, CentOS, Fedora and OEL)

The RPM installation method requires that you setup a repository before installing Zend Server Cluster Manager. For instructions on setting up a repository see Manually Installing Zend Server.
This method uses "yum" to handle all installations, upgrades and additional packages.

1.Once the environment is setup, run the appropriate command according to the product version and PHP support you require:
To install **Zend Server Cluster Manager** run:

```
yum install zend-server-cluster-manager
```

2. To clean your packages cache and ensure retrieval of updates from the web, run:

```
yum clean all
```

After installing, a completion notification will appear, with a notice that the servers have started.
To access the Administration Interface (Web) open your browser at:
https://localhost:10082/ZendServerManager (secure) or http://localhost:10081/ZendServerManager.
Upon initial log in, you will be prompted to define your password.

RPM (SLES and OpenSUSE)

The RPM installation method requires that you setup a repository before installing Zend Server Cluster Manager. For instructions on setting up a repository see Manually Installing Zend Server.

This method uses "zypper" to handle all installations, upgrades and additional packages.

1.Once the environment is setup, run the appropriate command according to the product version and PHP support you require:

To install **Zend Server Cluster Manager** run:

```
zypper install zend-server-cluster-manager
```

After installing, a completion notification will appear, with a notice that the servers have started.

To access the Administration Interface open your browser at: https://localhost:10082/ZendServerManager (secure) or http://localhost:10081/ZendServerManager.

Upon initial log in, you will be prompted to define your password.

Windows

The following procedure describes how to install Zend Server Cluster Manager on Windows using a binary distribution.

To install Zend Server Cluster Manager:

1. After completing the download, double-click on the .exe file to start the installation process.
2. Read and accept the License Agreement to start the installation process.
3. Select a Web Server type.
   There are two options, to set Zend Server Cluster Manager to run with an existing IIS Web Server or Install Apache.
4. Browse to a location for installing Zend Server Cluster Manager or use the default destination: "C:\Program Files\Zend\".
5. In the MySQL Database Installation dialog you are asked whether you would to install MySQL on your machine. The options are:
   - **No** - Does install MySQL on your machine. Select this option if you already have MySQL installed on your machine.
   - **Yes** - Installs MySQL on your machine. If you select this option, you must specify a root password in the Specify MySQL root password section.
6. Click the **NEXT** button to advance to the Confirmation dialog.
7. Click Install to start the installation.

A browser opens after the installation, to display the Administration Interface's login screen. Use the password you specified in the installation process to log in. If it was selected during the installation, a shortcut is added to your desktop, otherwise, bookmarking the page at this point will help you to easily locate the link.

Upgrading Zend Server Cluster Manager

The following procedures describe how to upgrade an existing Zend Server Cluster Manager for DEB, RPM and Windows.

RPM (using yum), RPM (using zypper), and DEB (using aptitude)

The following procedure describe how to upgrade Zend Server Cluster Manager using the supported methods (yum, aptitude and zypper).

RPM Upgrade Note:
After upgrading, you will need to manually start your server by running the command: <install_path>/bin/zendctl.sh start.

To perform these actions you must have root privileges.

To upgrade RPM (RHEL, CentOS, Fedora and OEL) using yum run:

```
yum update zend-server-manager
```

To upgrade DEB using aptitude run:

```
aptitude update

aptitude upgrade
```

To upgrade only Zend packages, run:

```
# aptitude install `dpkg --get-selections|grep zend| awk -F " "
'{print $1}' |xargs`
```

The upgrade process locates newer packages and downloads them.

To upgrade RPM (SLES and OpenSUSE) using zypper run:

```
zypper update
```

The upgrade process locates any components of the product version that are newer and downloads them.

Windows

To upgrade Zend Server Cluster Manager on Windows:

1. After completing the download, double click on the .exe file to start the upgrade process.
2. Click the **NEXT** button to advance to the Confirmation dialog.
3. Click **Install** to start the upgrade process.

Uninstalling Zend Server Cluster Manager

The following instructions describe how to uninstall Zend Server Cluster Manager according to operating system type.

DEB

The following instructions describe how to delete or uninstall using **'*aptitude*'**.

To perform these actions you must have root privileges.

To uninstall Zend Server Cluster Manager (leaving the configuration files in place) run:

```
 # aptitude remove '~nzend.* '
```

To delete Zend Server Cluster Manager from the system with no traces left run:

```
# aptitude purge '~nzend.* '
```

Both instances remove Zend Server Cluster Manager from your system. Information collected by Zend Server Cluster Manager and stored in the database will not be removed by the uninstall process.

If you want to delete this information:

To delete the database run the command:

```
mysql> drop database zend_monitor;
```

To delete the MySQL 'zend' user created by the Zend Server Cluster Manager installation:

```
mysql> drop user 'zend'@localhost;

mysql> drop user 'zend'@'%';
```

RPM

The following instructions describe how to uninstall Zend Server Cluster Manager:

To uninstall run:

```
zendctl.sh stop
```

And then run:

# yum -y remove zend-server-manager && yum -y remove `rpm -qa|grep zend|xargs`

To uninstall **ZSCM** with **PHP 5.2** run:

```
# yum -y remove zend-server-php-5.2 && yum -y remove `rpm -qa|grep
zend|xargs`
```

To uninstall **ZSCM** with **PHP 5.3** run:

```
# yum -y remove zend-server-php-5.3 && yum -y remove `rpm -qa|grep
zend|xargs`
```

This will stop the Zend Server Cluster Manager daemons and remove the program, including any additional packages that were installed.

When uninstalling, the configuration files are not removed. They remain in the same location with an additional suffix: .rpmsave so that they can be reused in a newer installation. For example: a file called example.ini is renamed to example.ini.rpmsave, after you run the uninstall.

Information collected by Zend Server Cluster Manager and stored in the database will not be removed by the uninstall process.

If you want to remove this information:

To drop the database run the command:

```
mysql> drop database zend_monitor;
```

To drop the MySQL 'zend' user created by the Zend Server Cluster Manager installation:

```
mysql> drop user 'zend'@localhost;

mysql> drop user 'zend'@'%';
```

Windows

The following instructions describe how to uninstall Zend Server Cluster Manager:

To uninstall:

1. Use the Windows Control Panel: **Start | Control Panel | Add or Remove Programs**.
2. In the **Add or Remove Programs** dialog, locate and click the Zend Server Cluster Manager package in the list.
3. Click "Remove".
   The Installer runs in uninstall mode.
4. Follow the instructions and click "Finish" to complete the uninstallation process.

This will stop the Zend Server Cluster Manager services and remove the program, including any additional packages that were installed.

To cleanup your system, after uninstalling Zend Server Cluster Manager also delete the following:

- Uninstall MySQL
- Delete <install folder>\ZendServerManager
- Delete <install folder>\MySQL51
- Delete <install folder>\Apache2

Licenses and Registration

Entering for the First Time

The first time Zend Server Cluster Manager runs, the Configuration Wizard is displayed.

In order to start working with Zend Server Cluster Manager you have to complete the information in the wizard by clicking Next to advance through the steps.

Note:

If you see a button called "Enter Without a License you are viewing Zend Server and not Zend Server Cluster Manager. This option is not available for Zend Server Cluster Manager.

The Zend Server Cluster Manager Setup Wizard

Once you have completed the wizard you will be directed to the Zend Server Cluster Manager's dashboard.

After your information is defined for the first time in the Configuration Wizard, you can always go to **Administration | License and Password** to change/update your information.

If you only enter partial information, the next time you login to Zend Server Cluster Manager you will be prompted to fill in the missing information as follows.

- **Step 1: End User License Agreement**
  This mandatory step requires that you read and agree to the license before you can continue, the **Next** button will be disabled until this option is approved
- **Step 2: Password**
  This step will be displayed when you access the system for the first time. This password will be used by you, to log in to the Zend Server Cluster Manager Administration Interface. Passwords must be between 4 - 20 characters long. Additional security information can be found in Securing the Administration Interface.
- **Step 3: Licensing Details**
  This step is displayed until you enter valid license details.
  Zend Server Cluster Manager - Your license key and Order Number should be in the email sent to you after purchasing Zend Server Cluster Manager. This information will also be stored in your Zend user account along with expiration information.
  Cluster Members - In addition to your Zend Server Cluster Manager license, you should have a Cluster Members license. This license determines how many servers you can add to your cluster
  Subscription details - This non-mandatory field allows you to subscribe receive product related updated my email (unsubscribe method and details will be displayed in the emails you will receive).

- **Step 4: Database Connection**

  The database is intended for storing event information that is aggregated from the servers in your cluster. Before continuing to the next step you can see a list of the database settings and information that you will require to either locate or install a Database.
  This step requires that you enter your database information. You can choose to connect to an existing MySQL 5.0 (or higher) database that you may have or allow the wizard to create a schema for you.
  Username -only accepts a valid database username.
  Password - only accepts a valid password to an existing database
  Host - only accepts a valid host name/IP Address of a server. The following values are not accepted, localhost, 127.x.x.x.x or 0.0.0.0
  Port - only accepts a valid port number by default, 3306.
  Create the database for me - This will create a MySQL 5.0 database with default values, the default username is 'root'.
  I have already set up the database - This will connect to the database on the defined host and the username will be changed to 'zend'.

For instructions on how to manually setup the database on your own see Zend Server Cluster Manager Database.

If you do not already have a license, go to the licensing page on zend.com to find out how to get a license.

License Expiration

Before a 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 or if you enter an invalid license, Zend Server Cluster Manager will display a License Page. Any page you try to access will keep redirecting to this page until a valid license is entered. During this time, all settings are kept and are immediately restored, along with the functionality, when a new license is entered.

Password Management

For security reasons, Zend Server cannot restore your password for you. However, you can reset your password.

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 Cluster Manager 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.

UNIX, Linux and Mac OS x, 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 Cluster Manager 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 Cluster Manager 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 Cluster Manager requiring you to log in with the new password the next time you access the Administration interface.

Zend Server Cluster Manager Database

Zend Server Cluster Manager requires a MySQL 5.0 (or above) database in order to store information aggregated from the servers belonging to the cluster.

When installing Zend Server Cluster Manager for the first time, you will be prompted to setup the Zend Server Cluster Manager database.
This can be done in one of two methods:

1. **Create the database for me** - allows you to provide an administrator username (usually root) and password for the database. The setup wizard will then use these credentials to create a schema and a dedicated (unprivileged) user which will be used by Zend Server Cluster Manager to store data. The administrator credentials are not saved and will never be used beyond this step.
2. **I have already set up the database** - allows you to manually create a schema and a user in your existing database and set Zend Server Cluster Manager to use these existing credentials. While this method requires additional manual intervention, it may be more suitable for environments where the person installing Zend Server Cluster Manager does not have administrator permissions to access the database. Instructions on how to manually configure the Zend Server Cluster Manager database are as follows:

Manually Configuring the Zend Server Cluster Manager Database

This procedure describes how to manually create and setup a Zend Server Cluster Manager database schema and user.
Before creating the database, make sure that the server on which you are installing the database is accessible by Zend Server Cluster Manager and all Zend Server instances that are potential cluster members.

To manually create and setup a Zend Server Cluster Manager database:

1. Install a MySQL Server 5.0.X or 5.1.X you can also use an existing database server for this purpose as long as it is a compatible MySQL version and accessible from all the servers.
2. Open MySQL to external connections by editing the MySQL configuration file (usually my.cnf on Linux; my.ini on Windows)
   i. Backup your current configuration file
   ii. Comment-out or remove *skip-networking* (if it is set)
   iii. Set *bind-address* to 0.0.0.0

    iv. Restart MySQL

3. Connect to the MySQL database with an administrator user (e.g. root)
4. Create a schema named 'zend_monitor'. This will be used by Zend Server Cluster Manager to store Data:

```
CREATE DATABASE IF NOT EXISTS zend_monitor;
```

5. Run the following commands to create a user (e.g. 'zend') which will be used by Zend Server Cluster Manager to access the database you just created:
   Note: Replace <username> and <password> in the following SQL commands with the username and password you wish to use.

```
GRANT CREATE,DROP,ALTER,DELETE,INDEX,INSERT,SELECT,UPDATE,CREATE TEMPORARY
TABLES,LOCK TABLES,CREATE VIEW,SHOW VIEW,ALTER ROUTINE,CREATE
ROUTINE,EXECUTE ON `zend_monitor`.* TO '<username>'@'%' IDENTIFIED BY
'<password>';
FLUSH PRIVILEGES;
```

6. Disconnect from the database and populate the database you have just created by running the following command as root:
   Replace <username> with the username you created in step 5, you will also be prompted to type the password for this user.

```
mysql -u <username> -p zend_monitor <
<install_path>/share/mysql_create_monitor_db.sql
```

In the Zend Server Cluster Manager Configuration Wizard (step 4), select the option "I have already setup the database and enter the user name, password, host, and port for the schema you have just created.

Troubleshooting Service Inconsistency

The following error message appears in Zend Server Cluster Manager when there is a service/daemon inconsistency in the cluster "State of services on this server is inconsistent with the rest of the cluster. Click here for more information"

Why does this happen?

This happens when a service or daemon's running state on one or more of the servers in the cluster is not consistent. Normally this will happen when a new "Vanilla" server is added to the cluster or when a service/daemon that needs to be running has to be started. Zend Server Cluster Manager, requires that all servers in the cluster have the same state and that all service/daemon states be the same throughout the cluster.

What should I do?

Depending on if you need this service/daemon or not, start or stop them to create a consistent state throughout the cluster.

To view the cluster's current running status see **Cluster Setup | Components**.

To control service/daemon activity see:
Windows: Package Setup and Control Scripts
Linux Mac: Package Setup and Control Scripts

Zend Server Reference Manual

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

About

Zend Server 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.

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 to optimize and extend your PHP capabilities. The tools included in Zend Server 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, 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
- For Zend Server installation directories, see the Zend Server for IBM i Installation Guide.

What's New in Zend Server

What's New in Zend Server 5.1.0

- Web API - For more information see the Web API Reference Guide.
- PHP 5.2 packages upgraded to version 5.2.17. Additional release information is available at http://www.php.net/releases/5_2_17.php.
- PHP 5.3 packages upgraded to version 5.3.5. Additional release information is available at http://www.php.net/releases/5_3_5.php.
- Zend Framework packages upgraded to version 1.11.3. Full change log is available at http://framework.zend.com/changelog/1.11.3.
- Full support of RedHat Enterprise Linux server version 6.0

What's New in Zend Server 5.0.3

- Code tracing supports next/previous error navigation and enables textual searching in the trace view.

What's New in Zend Server Cluster Manager 5.0.3

- Zend Server Cluster Manager disabled mode – enables to take a cluster member out of the cluster for offline maintenance and then re-joining the member while maintaining the server's configuration and Cluster Management related data .
- Detailed cluster members status reporting in Zend Server Cluster Manager.
- Zend Server Cluster Manager upgrade workflow improvements.

What's New in Zend Server 5.0.2

Video Procedures

Watch a video that shows the steps to complete a procedure. The video icon will be displayed next to procedural steps that already have a video.

Special Requests are welcome! Feel free to ask for videos that you would like to see by sending a message to documentation@zend.com.

Code Tracing?

Code tracing captures PHP application execution both in production and in test lab environments. This allows developers to replay reported problems instead of trying to re-create them. As a result, there is a dramatic decrease in time consumed by root cause analysis.

Job Queue

Job Queues provide offline asynchronous processing of tasks and activities that can be run independently of the end user experience. For an overview of the Job Queue architecture see Zend Job Queue.

PHP 5.3

New supported PHP version. This release is a major improvement in the 5.X series, which includes a large number of new features and bug fixes. For a detailed list of what's new see http://php.net/releases/5_3_0.php.

Password Management

After completing the Installation process and opening Zend Server, 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 cannot restore your password for you. However, you can reset your password.

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 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 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 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 requiring you to log in with the new password the next time you access the Administration interface.

Registration

Registration Wizard

The first time Zend Server runs, the registration wizard is displayed.

1. The first step is the license agreement.

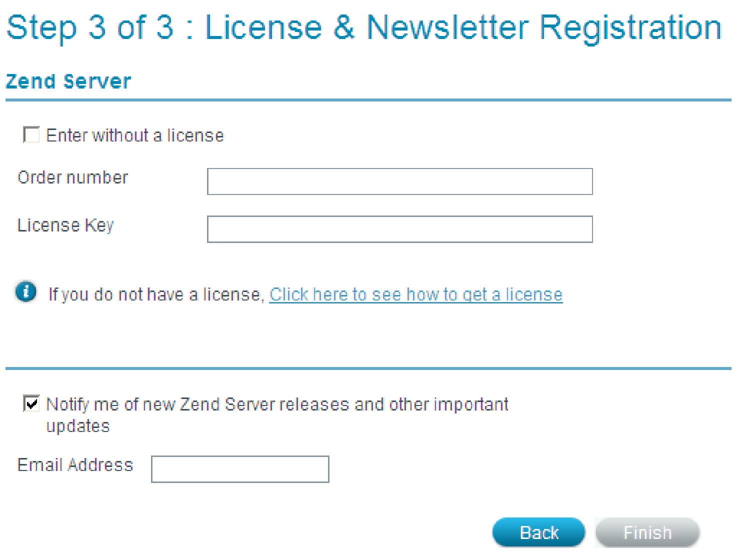
   To continue and install Zend Server, you must accept the license agreement.

2. The second step is the password page. Your password is used to log in to the Administration Interface, either from the main login page accessed from your browser or from the Zend Controller.

   - If you are using the Zend Controller locally or remotely (i.e., Zend Server and Zend Controller are located on separate machines), make sure that the Zend Controller settings match your Zend Server settings. Click here for instructions on how to change your Zend Controller settings according to your operating system.
   - Passwords must be between 4 - 20 characters long.
   - This step is also displayed when your license expires or when you reset your password. After you define your password the first time, you can always change your password from the Administration Interface. For more information, see Password Management.
   - To further secure Zend Server, please refer to Securing the Administration Interface

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

   1. To enter without a license mark the "Enter without License" check box.
   2. If you have your license details, enter them in the Order Number and License Key Fields. This information is stored in your zend.com account or under the account used for the purchase.



☐ Enter without a license

Order number	
License Key	

If you do not have a license, Click here to see how to get a license

☑ Notify me of new Zend Server releases and other important updates

Email Address	

Note:
When attaching a server to Zend Server Cluster Manager, the Zend Server GUI will be disabled. This is to allow Zend Server Cluster Manager the ability to have sole control over settings and configuration in order to prevent inconsistency that could result in loss of information.

License FAQ

How do I just take a look at the product?

If you enter Zend Server without a license, you can run Zend Server in Community Edition Mode. In this mode, Zend Server 's Community Edition features (PHP 5.x, Zend Data Cache, Zend Debugger, Zend Guard Loader, Zend Java Bridge and Zend Optimizer+) are available and the features that require a license are visible and disabled.

To enter the Community Edition mode, do not enter an Order Number and License Key.
Click "Enter Without a License" to start using Zend Server in Community Edition mode.

As soon as you enter a valid license, all licensed features are automatically activated for the license period.

How do I get a License?

If you do not already have a license, go to the licensing page on zend.com to find out how to get a 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.

If you have just installed Zend Server:
To enter a license, enter your Order Number and License Key as stated in your confirmation e-mail and click Enter.
If you have already been running Zend Server in Community Edition Mode or with an evaluation license:
In the Administration Interface go to Administration | Password and License.
Enter your new license details into the "Update License" area.
Click Update License to apply the changes.

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

License Expiration

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

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.

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'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.

Navigation inside the Administration Interface is done by clicking on the tab menus. Each main tab has several sub-tabs called pages. The layout is as follows:

- **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 Servers | Components | Extensions | Directives | Debugger | Monitor | Job Queue
- **Administration -** The Administration tab is the main area for configuring your Zend Server system settings and it includes Passwords | Update Notifications | API Keys

In addition to the main Administration Interface, Zend Server 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:

- Recent Events show the top five most critical events that occurred in your system. Clicking on an Event ID will display the full audit trail. The full list can be found in **Monitor | Events**.
- Tasks include quick links to configuration tasks and useful information. Clicking on a link directs you to the appropriate page in the Administration Interface.
- The System Overview lists information about your environment including PHP version and a Zend Components status display.

Events

The *Events* page is accessed from **Monitor | Events**

The Events page is the main display for events that are generated based on the conditions defined by the Monitoring Rules. Events contain information about a specific occurrence that indicates that your environment is displaying uncharacteristic behavior. You can use the Events page to perform additional actions to diagnose the problem.

The actions that can be performed from this page are Filter, View Event Details and Change Status. You can also search for an event using the event's ID number if you know what the number is.

Each individual event includes specific information about the occurrence, such as when it happened, how many times it happened and other details that can assist a developer in diagnosing the event. More advanced diagnostic information includes information about the request that generated the event or backtrace information.

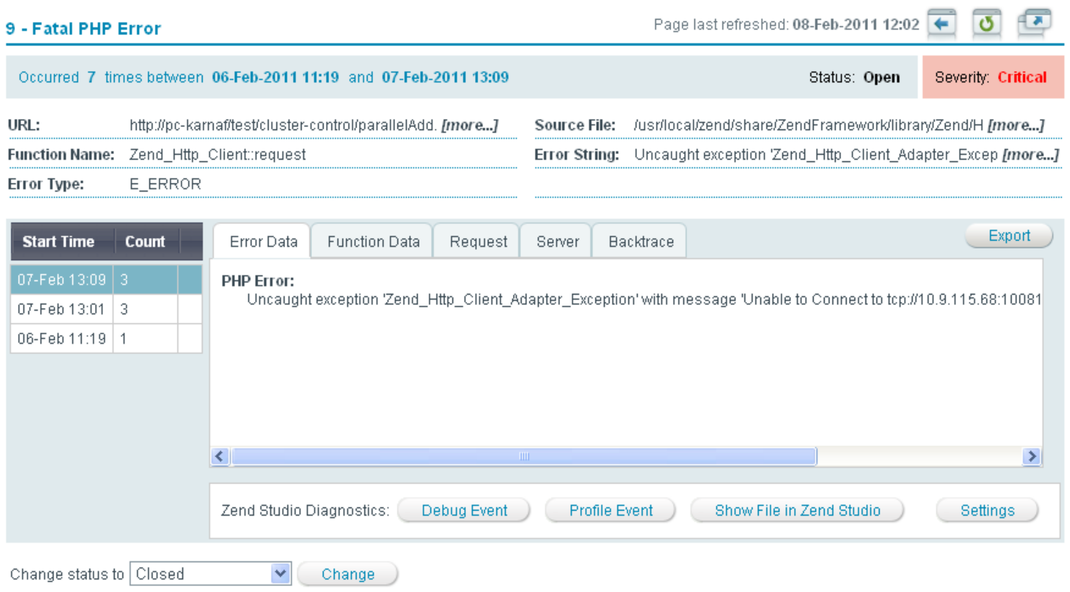
Each event type is slightly different and therefore the information collected and displayed for each event may differ. For example, a Slow Request event does not include information on a source file or line of code, because the event was generated by a request. The same is true for Java backtrace information: Java backtrace information is only included for Java exception events.

Event Details

The Event Details page is accessed from **Monitor |** ***Events*** by selecting an event from the list and clicking on the row.

The Event Details page is the main display area for information regarding the occurrence of a specific type of event.

Information on how an event is triggered is presented in Event Rules.



9 - Fatal PHP Error

Page last refreshed: 08-Feb-2011 12:02

Occurred 7 times between 06-Feb-2011 11:19 and 07-Feb-2011 13:09

Status: Open

Severity: Critical

URL: http://pc-karnaf/test/cluster-control/parallelAdd. [more...]

Function Name: Zend_Http_Client::request

Error Type: E_ERROR

Source File: /usr/local/zend/share/ZendFramework/library/Zend/H [more...]

Error String: Uncaught exception 'Zend_Http_Client_Adapter_Excep [more...]

Start Time	Count
07-Feb 13:09	3
07-Feb 13:01	3
06-Feb 11:19	1

PHP Error:
Uncaught exception 'Zend_Http_Client_Adapter_Exception' with message 'Unable to Connect to tcp://10.9.115.68:10081

Change status to Closed

Note:

Not all events display the same information. Only information relevant to the specific event type will be shown.

The following actions can be performed from the Event Details page:

- **Back to Events** - Returns to the Events page.
- **Refresh** - Refreshes the report. Refreshing the report updates the event counter if the event occurred additional times.
- **Detach** - Opens the report in a new browser window.
- **Change Status** - Changes the status of the event displayed. For a complete explanation of event handling, see Working with Events.

On the Event Details page, users can view a general summary of an occurrence, its status and diagnose the occurrence with Zend Studio for Eclipse.

Event Detail - Information

The following list describes the information types displayed in the Event Details Page

Top Bar:

- **Number of Occurrences** - The accumulated amount of times the event was triggered between the first time the event occurred and the last time the event occurred. Refreshing the report updates this number if the event occurred additional times.
- **Status** - The status if the event: Open, Closed, Reopened and Ignored. For a specific event, the status can be changed in the Event Details page: For multiple events, the status can be changed in the Events page.
- **Severity** - The severity of the event (Warning or Critical). The severity is defined in the event's master settings in the Monitoring tab.

Event Details Table:

Events are aggregated into groups based on the time they occurred. The aggregation is set to five minutes: Thus, all the events that occur within that time frame are grouped together. Each time a set of events is aggregated (i.e., a new group is created), the occurrence details are collected again. To view the event occurrence details for a group, click on the group in the Event Details table: The display on the right will change to display the following options.

The Event Details Table options are:

- **Node** - The server on which the event occurred .If the node's status is "Unknown" the details you are viewing were collected from a node that no longer belongs to the cluster.
- **Start Time** - When the first group of the instance occurred.
- **Count** - The number of events triggered in the same time frame (set to five minutes)
- - An indicator that trace information was collected for that occurrence.

Clicking on one of these options updates the display with the relevant information.

Event Details

The Event Details are a collection of information relevant to the event that occurred. Clicking on a time in the table on the left (Event Details Table) will display the information relevant to the selected occurrence/es. Through this you can find out for example, if a different function caused the error or if a different message was thrown.

The following list presents the possible details that can be displayed for a specific occurrence:

- **Export -** generates an XML file containing the selected event's information.
- **General Data** - Displays information about the event: The data changes according to the event type.
- **Function Data** - Displays information on the function that triggered the error, including the function name and arguments.
- **Request** - Displays information about the request. The superglobals (POST, GET and COOKIE) are always displayed. The other parameters (RAWPOST and FILE) are displayed only when there is relevant information to display.
- **Server** - Displays the superglobals SERVER and ENV when there is relevant information.
- **Session** - Displays the superglobal SESSION when there is relevant information.
- **Backtrace** - Displays all the function calls that were made before the event was triggered, including the relevant files for each function.
- **Error Data** - Displays the PHP error and the Java backtrace if there was a Java exception.
- **Custom Variables** - Displays information for a custom event (i.e., class and user-defined information that was passed to the event when it was triggered).
- **Job Queu**e - *Job Queue* related events display the reason the Job generated an event.
- **Code Tracing** - Code Tracing related events display the reason that Code Tracing g4eenrated and event.
- **Zend Studio diagnostics** - Displays the actions can be applied to event details if Zend Studio for Eclipse (ZSE) is installed and Zend Server is configured to communicate with it:
  - **Debug Event** - Initiates a debug session for the event URL in ZSE.
  - **Profile Event** - Profiles the event URL, using the ZSE Profiler with the same parameters (GET, POST, COOKIE, HTTP headers, etc.).
  - **Show File in Zend Studio -** Opens the file where the event occurred in Zend Studio. This option makes it possible to use Zend Studio to edit files and implement changes for multiple servers.
- **Settings**: through this option you can choose to apply the Studio Integration actions to the Originating Server (the server on which the event was triggered) or to an Alternate Server (a different server running the same environment).Additional configuration settings are set in Server Setup | Monitor.

Jobs

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

The Jobs page is the main display for Job Queue jobs that are scheduled in your environment. The jobs listed in this page are created as follows:

- Based on conditions defined in the Recurring Jobs tab.
- Non Recurring Jobs - one time jobs that have been manually triggered using the Job Queue API
  - Jobs generated by the Job Queue API - jobs created inside your code to off load resource intensive processes to another time.

You can use the Jobs page to perform additional actions to follow up on Job activity.

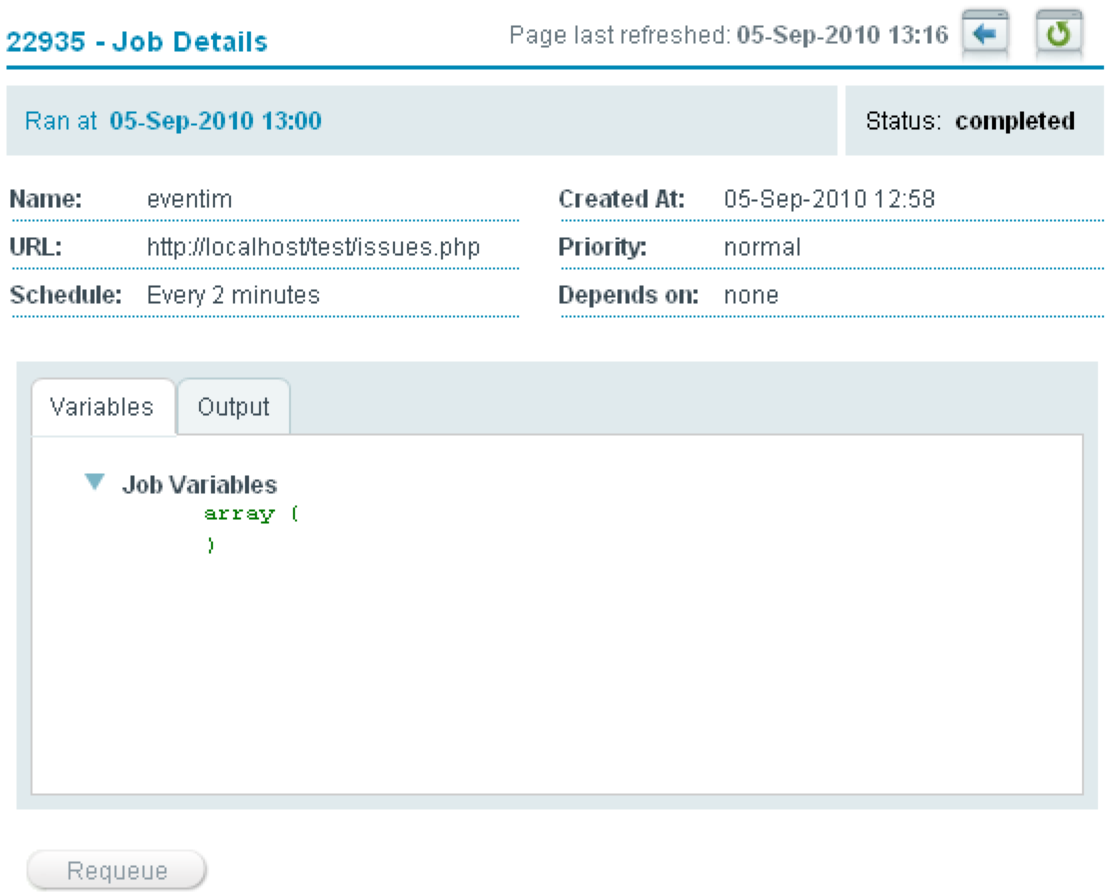
The actions that can be performed from this page are Filter, View Job Details and Delete Jobs. For more information see Managing Jobs.
New jobs are created in **Rule Management | Recurring Jobs**.

Each individual Job includes specific information about the occurrence, such as the URL, Application, Status, Priority and Run Time all details that help assess job activity. More advanced diagnostic information includes information about the variables, priority and dependencies. Clicking on a job will open the Job Details Page.

Job Details

The Job Details page is accessed from **Monitor |** ***Jobs*** by selecting a Job from the list and clicking on the row.

The Job Details page is the main display area for information regarding a specific *Job Queue* job. Information on how a job is created is presented in Recurring Jobs.



22935 - Job Details

Page last refreshed: 05-Sep-2010 13:16

Ran at 05-Sep-2010 13:00

Status: completed

Name:	eventim	**Created At:**	05-Sep-2010 12:58
URL:	http://localhost/test/issues.php	**Priority:**	normal
Schedule:	Every 2 minutes	**Depends on:**	none

```
array (
)
```

The information displayed in the Job Details page is as follows:

- **Ran at/Scheduled for** - the date and time the job ran.
- **Status** - an indicator of the job's state: Pending, Waiting for predecessor, Running, Completed, Ok, Failed, Logically failed, Timeout, Scheduled, Suspended.
- **Name** - the name given to the job when it was created.
- **URL** - the location of the job file to be run.

- **Priority** - The Job's priority. Recurring Jobs are always set to normal. Jobs created with the Job Queue API can hold different priorities. In addition to normal they can be Low, High or Urgent depending on the importance of the job.
- **Depends on** - If the Job should only be run after another job the Job's name will appear only for Jobs created with the Job Queue API.
- **Created at** - the date and time the Job Details report was created.
- **Applications** - The server on which the job was run.
- **Schedule** - the date and time intervals the job is set to run on.
- **Variables** - The input variables that the job received.
- **Output** - The HTML output of the Job which includes the HTTP response, headers and body.

The actions that can be performed from the Job Details page are:

- **Back to Jobs** - Returns to the Jobs page.
- **Refresh** - Refreshes the Job information.
- Requeue **Re-queue** - Reschedule a non re-occurring Job to run.

Queue Statistics

The Queue Statistics page is accessed from **Monitor | Queue Statistics**.

The Queue Statistics page is the main display for information regarding active *Jobs* that you have defined in your system.

With this information you can track, monitor and evaluate the scope of active Jobs in your system.

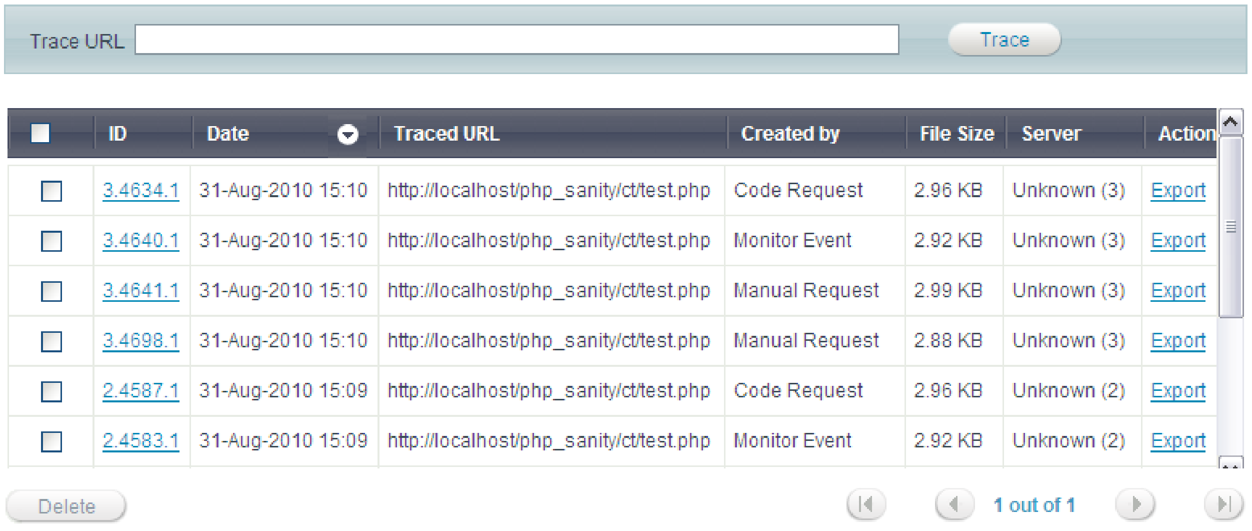
Statistical Parameters

- The daemon started at
- Number of jobs added since last startup
- Number of jobs served since last startup
- Number of waiting jobs
- Number of jobs waiting for predecessor
- Number of running jobs
- Number of completed jobs
- Number of failed jobs
- Number of scheduled jobs
- Average job wait time
- Average job run time

Code Tracing

The Code Tracing page is accessed from **Monitor | Code Tracing.**

The Code Tracing page is a central display and management area for all traced information. In addition Trace information can be viewed per event by drilling-down to a specific event in **Monitor |** ***Events***.
Zend Server Code Tracing captures full execution data of PHP applications in real time. The execution data includes function call trees, arguments and return values, function execution duration, memory usage and indication for an executed file's name and line of code. This enables you to capture problems when they occur, which eliminates the need to set up environments and reproduce the steps that led up to the failure.



☐	ID	Date	Traced URL	Created by	File Size	Server	Action
☐	3.4634.1	31-Aug-2010 15:10	http://localhost/php_sanity/ct/test.php	Code Request	2.96 KB	Unknown (3)	Export
☐	3.4640.1	31-Aug-2010 15:10	http://localhost/php_sanity/ct/test.php	Monitor Event	2.92 KB	Unknown (3)	Export
☐	3.4641.1	31-Aug-2010 15:10	http://localhost/php_sanity/ct/test.php	Manual Request	2.99 KB	Unknown (3)	Export
☐	3.4698.1	31-Aug-2010 15:10	http://localhost/php_sanity/ct/test.php	Manual Request	2.88 KB	Unknown (3)	Export
☐	2.4587.1	31-Aug-2010 15:09	http://localhost/php_sanity/ct/test.php	Code Request	2.96 KB	Unknown (2)	Export
☐	2.4583.1	31-Aug-2010 15:09	http://localhost/php_sanity/ct/test.php	Monitor Event	2.92 KB	Unknown (2)	Export

The trace displayed in the Zend Server web console enables you to view the execution history of your application and follow in the footsteps of an individual, problematic request to quickly pinpoint root cause. Furthermore, the **Export** option allows you to transfer this information into Zend Server allowing you to transfer the information to developers.

Zend Server Code Tracing is an in-depth diagnostic tool that will allow you to drill-down to the function level to view actual performance related information and statistics.
Trace information can be collected in one of two ways:

1. Collected as an additional level of event information by Monitor Rules mechanism to generate a trace when an event occurs.
   Traced information can contain information on more than one event that occurred according to the reoccurrences of the event.
2. Manually Triggered

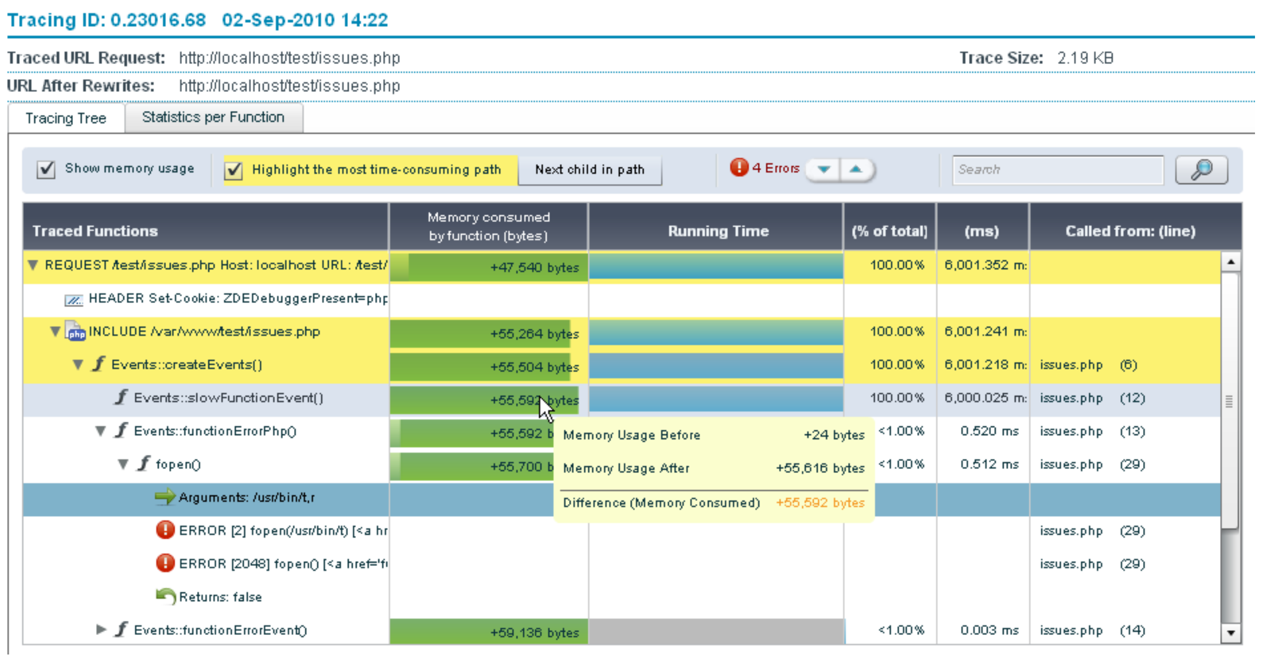
From this page you can:

- Trigger a URL trace - Manually run code tracing on a specific URL
- View trace information
- Delete trace information
- Export Trace Information

Code Tracing Tree

The Code Tracing Tree is accessed from **Monitor | Tracing** and selecting a trace ID from the list.

The Tree tab displays the call tree for a selected event or trace file.



Tracing ID: 0.23016.68 02-Sep-2010 14:22

Traced URL Request: http://localhost/test/issues.php Trace Size: 2.19 KB

URL After Rewrites: http://localhost/test/issues.php

Traced Functions	Memory consumed by function (bytes)	Running Time	(% of total)	(ms)	Called from: (line)
REQUEST /test/issues.php Host: localhost URL: /test/	+47,540 bytes		100.00%	6,001.352 m:	
HEADER Set-Cookie: ZDEDebuggerPresent=php					
INCLUDE /var/www/test/issues.php	+55,264 bytes		100.00%	6,001.241 m:	
Events::createEvents()	+55,504 bytes		100.00%	6,001.218 m:	issues.php (6)
Events::slowFunctionEvent()	+55,592 bytes		100.00%	6,000.025 m:	issues.php (12)
Events::functionErrorPhp()	+55,592 b		<1.00%	0.520 ms	issues.php (13)
fopen()	+55,700 b		<1.00%	0.512 ms	issues.php (29)
Arguments: /usr/bin/t,r					
ERROR [2] fopen(/usr/bin/t) [<a hr					issues.php (29)
ERROR [2048] fopen() [<a href='f					issues.php (29)
Returns: false					
Events::functionErrorEvent()	+59,136 bytes		<1.00%	0.003 ms	issues.php (14)

Memory Usage Before	+24 bytes
Memory Usage After	+55,616 bytes
Difference (Memory Consumed)	+55,592 bytes

Display Options:

- **Show Memory Usage** - Show or hide the Memory Usage column. Hovering over an item in this column will display a tool tip containing a comparison of memory usage.
- **Highlight most time consuming path** - shows the path that took the most time to run. Selecting this option makes the Next Child in path button appear:
  Next Child in Path - Clicking this button makes the blue indicator line and the display scroll and jump to the next child in the tree.
- **Errors** - An indicator for the amount of errors the code threw. Clicking on the up/down arrows will move the cursor to the error messages accordingly in the trace data and then it will return to the beginning.
- **Search** - a case insensitive search component for finding elements in the trace information. Entering a search item and clicking 🔍 will move the blue indicator to the first instance in the trace information, each time you click on 🔍 (or the return key) the indicator will move to the next instance

The Tree includes the following columns:

- **Traced Functions**: The name of the caught object in the trace file. This could be an argument, return value, function error, header etc.
- **Memory Consumed by function (bytes)**: The memory used by this item
- **Running Time**: A graphical representation of memory usage pointing out the before/after function runtime values of the total memory usage and the difference between them. This allows you to see the before value, the memory consumption after the function was run and the difference i.e. the amount of memory the function added by running.
- **(% of total)**: The percentage of the total script's runtime.
- **(ms)**: The time it took for the function to run in milliseconds (including children).
- **Called from (line)**: The line of code where the event happened (in the file stated in filename).

Navigation

In Tables:

- At any time you can hover over a component to see a tooltip that describes the component and in certain cases additional information.

In the Tree tab:

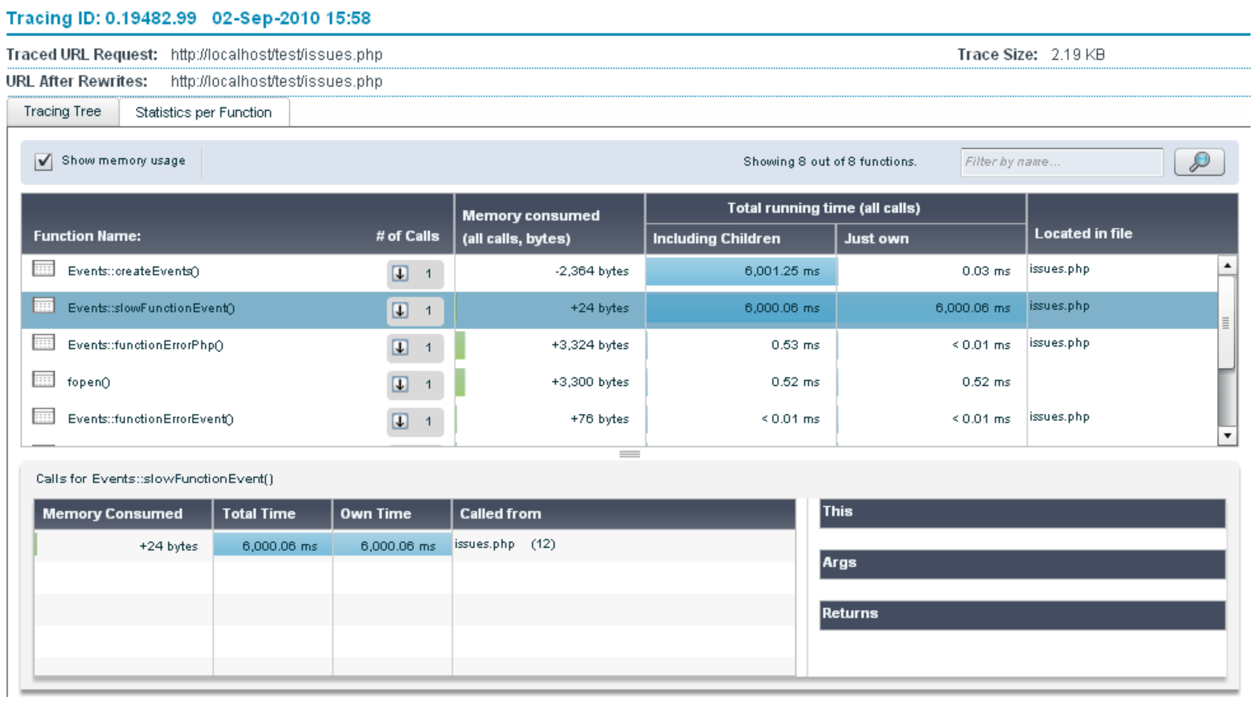
- Bold items indicate calls that took the most time to execute and this continues inside the call itself – indicating the slowest calls.
  This indicates the application’s critical path.
- Double clicking on any item will jump to the relevant function in the Statistics tab.

Code Tracing Statistics

Code Tracing Tree is accessed from **Monitor | Tracing** by selecting a trace ID from the list.

The Statistics per Function tab is a table based display that provides a statistical perspective of the data captured in the request. The same data is displayed in two different ways. The “All functions” area at the top that lists all the functions included in the dump for a certain occurrence and the “Calls for Functions” area at the bottom that displays the function calls for a function selected from the list (by clicking on the function).

Use this tab to investigate performance information such as the slowest function (sorting the table by 'own time' will help pinpoint this).



Tracing ID: 0.19482.99 02-Sep-2010 15:58

Traced URL Request: http://localhost/test/issues.php   Trace Size: 2.19 KB

URL After Rewrites: http://localhost/test/issues.php

Showing 8 out of 8 functions.

Function Name:	# of Calls	Memory consumed (all calls, bytes)	Total running time (all calls)		Located in file
			Including Children	Just own	
Events::createEvents()	1	-2,364 bytes	6,001.25 ms	0.03 ms	issues.php
Events::slowFunctionEvent()	1	+24 bytes	6,000.06 ms	6,000.06 ms	issues.php
Events::functionErrorPhp()	1	+3,324 bytes	0.53 ms	< 0.01 ms	issues.php
fopen()	1	+3,300 bytes	0.52 ms	0.52 ms	
Events::functionErrorEvent()	1	+76 bytes	< 0.01 ms	< 0.01 ms	issues.php

Calls for Events::slowFunctionEvent()

Memory Consumed	Total Time	Own Time	Called from
+24 bytes	6,000.06 ms	6,000.06 ms	issues.php (12)

The “All Functions” area includes the following columns:

- **Function Name**: The name of the function as it appears in the code.
- **# of Calls**: Function invocation count – how many times the function was called.
- **Memory Consumed (all calls, bytes)**: Graphical and numerical representation of the memory consumed by all invocations of the function.
  To highlight/disable the display of memory usage within the trace file, mark the check-box next to "**Show Memory Usage"**
- **Total Running Time**: Total time taken by this function’s invocations including nested function calls. Hovering over shows a tooltip with the average time.
  - **Including Children**: The total request time taken by this function’s invocation including nested function calls.

   - **Just Own**: Total time taken by this function's invocations not including nested function calls (i.e. the time it tool to call other functions).
- **Located in File**: The file where the function was defined. Internal functions are not defined in a file and therefore this column will be empty.

The "Calls for Functions" area includes the following columns:

- **Memory Consumed**
- **Total Time**
- **Own Time**
- **Called from**
- **File**: The file where the call happened
- **Line**: the line where the call happened
- **Total Time**: time consumed by the call, including nested functions.
- **Own Time**: time consumed by the call, excluding nested functions.

When selecting an item from the "Calls for Functions" list details about the actual call are displayed if it is an object.

Navigation

In Tables:

- At any time you can hover over a component to see a tooltip that describes the component and in certain cases additional information.

In the Statistics tab:

- Clicking on an item will display the function's calls in the "Calls for Functions" area.
- Double clicking on an item the "Calls for Functions" area will open it up to show where it happened in the Tree tab.
- Clicking on an item in the "Calls for Functions" area will display the calls argument and return values if it is an Object.

Search:

Enter a string in the search section Filter by name... and click to filter the display and only show the functions that match the entered string.

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 Code Tracing** - Release version and status.
- **Zend Data Cache** - Release version and status.
- **Zend Debugger** - Release version and status.
- **Zend Download Server** - Release version and status.
- **Zend Guard Loader -** Release version and status.
- **Zend Java Bridge -** Release version and status.
- **Zend *Job Queue*** - Release version and status.
- **Zend Monitor** - Release version and status.
- **Zend Optimizer+** - The status of the Optimizer+ component used for opcode caching and optimizations.
- **Zend Page Cache** - Release version and status.
- **Zend Session Clustering** - Release version and status

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.

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

Rule Management Tab

Monitoring

Monitoring Rules are defined and activated in **Rule Management | Monitoring** and they generate *events* that are displayed in the Events page.

The Monitor component is set to run out-of-the box, based on default settings. To change the Monitor component settings, see Working with Components. To configure a specific event, see Edit Rule (Monitoring) and to view generated events, see Events.

Monitoring is based on a set of predefined rules that can be configured to suit your environment's requirements (such as performance thresholds) or enabled and disabled as necessary. Once Zend Server is installed, the monitor component begins to create events. To find out more about event configuration methodologies, see Working with Monitoring.

From this page you can:

- Run a filter (start typing to filter the display).
- Edit a Rule - Click on a rules name to change the rule configurations.
- See if code tracing for events is active. This means that Events that are set to trigger the creation of Code Tracing information will collect that information when an event is triggered. Additional information about the different code tracing statuses can be found in Monitor.
- See which event actions are set to specific rules. This information is displayed in the main table under the "Event Actions" column.
- Change a Rule's Status - Select multiple rules and apply one of the following changes:
  - Enable Rules - Apply this to disabled rules to start generating events based on the conditions set in the rule.
  - Disable Rules - Apply this to enabled rules to stop generating events.
  - Enable Emailing Action - Reactivate the emailing action. This will trigger an email each time the event is generated.

Note:
Enabling emailing will start to send emails to the system default email address, if you want to send the email to a different address you have to manually edit the rule.

- Disable Emailing Action - This will disable the emailing action for the selected rules. Once disabled, the rule does not save non-default email addresses and when the Action is enabled the emails will be sent to the system default address.
- Enable Code Tracing Action - Activates code tracing and a trace file will be generated each time the event is triggered.
- Disable Code Tracing Action - disables Code tracing.

Note:
To apply changes you must restart the server.

For more information about when to disable a rule, see Working with Monitoring.

In addition to creating issues to display events that occurred, you can define a rule to generate trace information using Zend Code Tracing see Edit Rule to find out how to trigger code trace collection from a Rule.

Rule Types

The *events* listed in Monitor | Events are based on rules defined in Rule Management | Monitoring, If a rule is enabled, it is displayed in the Events page when it is triggered. When more than one event with a high percentage of similarity is triggered, it is aggregated into a single report. These reports are called issues.

The following list displays the possible rule types that can generate issues in the Events page:

- **Function Error** - A Function Error rule provides specific information about the root cause of an error that may not be related to a PHP error. QA and production teams can use this event rule type to identify run-time events, as opposed to PHP errors, which identify code-related/syntactical events. Severity: Warning.
- **Database Error** - A Database Error rule provides specific information about the root cause of an error that may not be related to a PHP error. QA and production teams can use this event rule type to identify database connectivity and query events, as opposed to PHP errors, which identify code-related/syntactical events. Severity: Warning.
- **Slow Function Execution** - Slow Function Execution rules identify bottlenecks within functions, providing a more granular approach than finding bottlenecks in pages. This type of event rule is particularly useful in the production process, because it can pinpoint performance bottlenecks for user-specific functions, as well as the predefined list of functions that are considered prone to slow execution. Severity: Warning and Critical.
- **Slow Query Execution** - Slow Query Execution rules generate an event when database related function execution rises above the given threshold. Slow Query Execution events identify slow queries that are related to database performance that can directly influence Web server performance.
- **Slow Request Execution** - Slow Request Execution rules generate an event when script execution time exceeds defined limits. These event rules are used to maintain script runtime performance standards. The settings can be relative to a specified percentage or set to an absolute value. Severity: Warning and Critical.
- **High Memory Usage** - Memory Usage event rules identify when scripts use excess memory resources that can, in turn, reduce the application's performance. This type of rule is primarily used in production environments, but QA teams can also benefit from monitoring by kilobyte (KB) or by the percentage of memory used by a script to execute. The settings can be relative to a specified percentage or set to an absolute value. Severity: Warning and Critical.
- **Inconsistent Output Size** - Inconsistent Output Size event rules verify that pages render the same output to the client each time. If pages do not render the same output each time, this

indicates that some clients may be seeing different output. This is an error situation. Production environments use these rules to indicate possible usability issues. Severity: Warning.

- **Uncaught Java Exception** - Java exception event rules increase the visibility of issues originating in the Java side, by indicating when uncaught Java exceptions occur in Java code invoked from PHP via the Java Bridge.
  This event identifies uncaught Java exceptions and provides Java-related backtrace information, including which part of the PHP code triggered the error. Severity: Critical.
- **Custom Event** - This unique event rule is used to initiate events from a PHP script. Custom events control event generation, in contrast to other events, which are triggered by specific occurrences. Custom events are used to generate an event whenever the API function *monitor_custom_event()* is called from the PHP script. Severity: Warning.
- **PHP Error** - PHP Error rules identify all types of PHP errors, including hard errors that cause stops in page execution, warnings that interrupt the end user experience, and notices that could lead to larger problems.
  This type of event rule is useful in QA processes, to identify problems that may have gone unnoticed during production. Production environments can benefit from using this PHP intelligence feature to alert administrators to application runtime errors that could seriously impact the end user's experience. Severity: Warning and Critical.
- **Job Execution Error** - This event is generated when a job could not run.
- **Job Logical Failure** - This event is generated when a job reports a logical failure.
- **Job Execution Delay** - This event is generated when job execution is delayed by x seconds from the planned start time that was defined in the job. The delay time is defined in the Job queue settings page: **Server Setup | Job Queue**.
- **Job Queue High Concurrency Level** - This event is generates when the job queue daemon is at or close to its maximal concurrent job limit.
- **Tracer - Failed to Write Dump File** - This event is generated when the Code Tracer could not create a dump file.
- **Skew Time** - when the job is executed later than scheduled

Important:
Caught Java exceptions are considered part of the normal exception flow, therefore only uncaught exceptions are reported.

Some rule types can be configured twice - once with absolute value settings and again with relative value settings.
An absolute setting is used to configure a specific value and a relative setting is used to configure a percentage of a selected value.

Cluster Rules

The following rules are triggered only in a Cluster

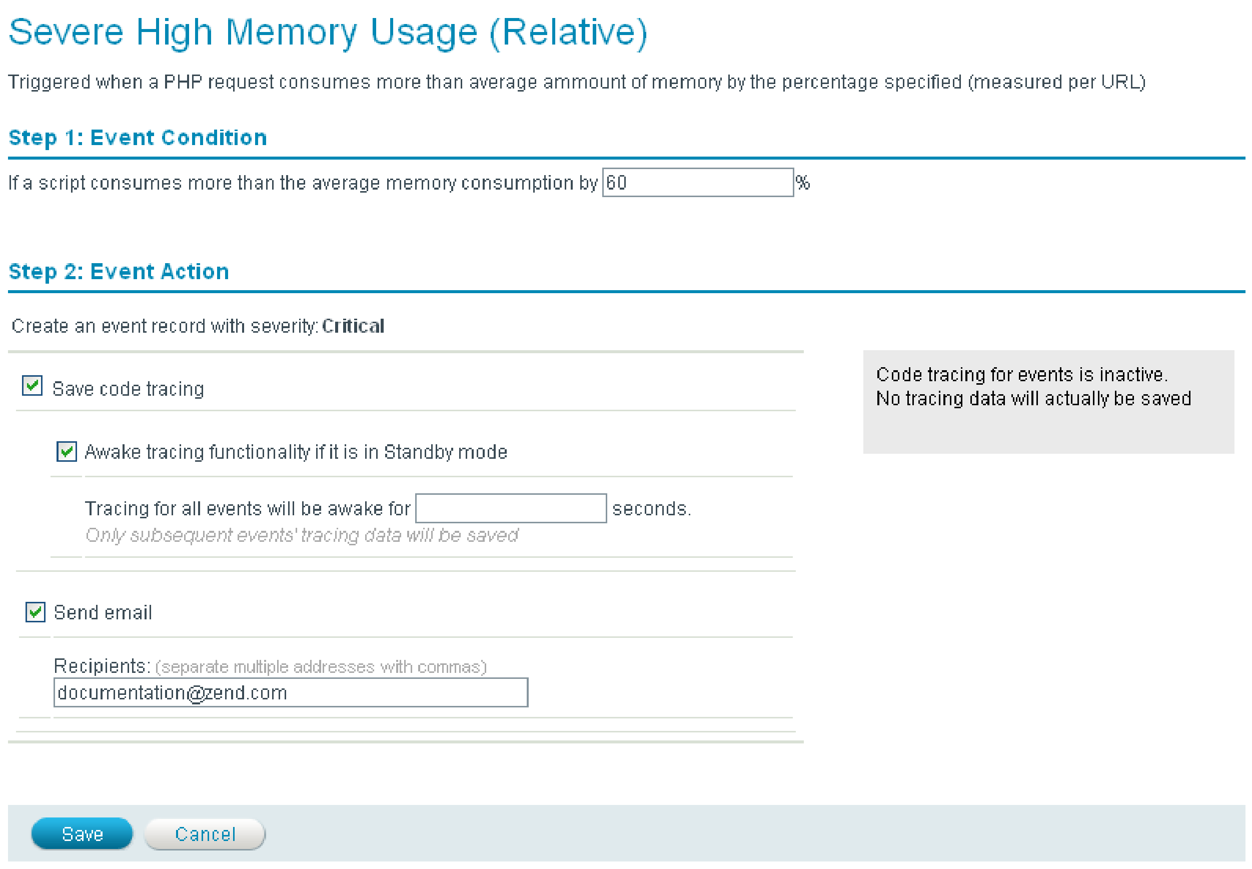
- **Restart failed** - This event is generated when the PHP on the nodes was not restarted. You should wait and try to restart again.
- **Configuration does not match the cluster -** This event is generated when all the cluster has a certain extension turned-on, but on the specific node it had an error loading. Possible diagnostic options are checking directives, files and extensions for mismatches.
- **Node added successfully**
- **Node removed successfully**
- **Node is not responding**

Edit Rule (Monitoring)

The rule editing page is accessed from **Rule Management | Monitoring.** To edit a specific rule, click the rule name and the rule editing page will open.

This page is used to edit the conditions which generate an event (as displayed in **Monitor |** **Events**).

The following image is an example of a rule, rule layout and parameters differ from rule to rule.



Severe High Memory Usage (Relative)

Triggered when a PHP request consumes more than average ammount of memory by the percentage specified (measured per URL)

Step 1: Event Condition

If a script consumes more than the average memory consumption by 60 %

Step 2: Event Action

Create an event record with severity: Critical

✓ Save code tracing

✓ Awake tracing functionality if it is in Standby mode

Tracing for all events will be awake for ____ seconds.

Only subsequent events' tracing data will be saved

✓ Send email

Recipients: (separate multiple addresses with commas)

documentation@zend.com

Code tracing for events is inactive.
No tracing data will actually be saved

Rule Details

The rule editing page lists all the relevant information for the selected event. The initial event settings are based on the system defaults.

- **Name -** A descriptive name for the event and the event's status (Moderate or Critical).
- **Description -** A generic description of what triggers the event.

Step 1: Event Condition

- **Event Type** - The type of event this rule generates. For a list of event types, see Event Types.
- **Event Condition -** The exact function or parameter value that triggers the event.

Step 2: Event Action

- **Event Action -** As soon as an event of this type is generated, trigger one of the following actions:
    - Save Code Tracing - Each time this event is triggered, create a trace file for the Event information. This option is dependent on the Code Tracing settings in Server Setup | Monitor and will only run if Code Tracing is set to **"Active". If it is set to "Standby"** you need to also set the rule to include the **"Awaken tracing functionality if currently in Standby mode"** option. You can set a duration for the Code Tracing component to be "awake" - this will revert the code tracing component to its former status after the duration specified this option.
    - Send email to: send an email to a specified recipient/s. If you disable the email sending setting (in **Rule Management | Monitoring** by selecting the rule and changing the status), when the setting is reactivated all the email addresses will be lost and the email setting will be set to the default email address.

Save Code Tracing Options

1. **Save Code Tracing**: This action will save code trace data when an event of the proper type is generated by Zend Server's monitoring capabilities. Note that tracing must be set to "active" mode.
2. **Awaken Tracing Functionality**: This action enables code tracing for a specific timeframe (only relevant when code tracing is set to "standby" mode). During this timeframe, code tracing for monitoring is active and trace data is available to be saved according to regular tracing rules and conditions. At the end of the timeframe, code tracing for events will automatically revert to standby mode.

Note:
When code tracing for monitoring is set to "inactive" mode, code tracing will ignore any save trace request coming from generated events.

Caching

The Caching page is accessed from **Rule Management | Caching.**

The Caching page is the central configuration area to configure rules to cache content by URL. Caching by URL makes it easy to eliminate situations where the same file is used in multiple instances, such as when the same file is used to redirect to several pages.

Note:
Zend Server also provides the ability to cache content using the Zend Data Cache (API). To read more about the Data Cache, see Working with the Data Cache.

From this page you can:

- **Filter** - Search for a specific rule by name.
- **Add Rule** - Add a new rule to the Caching page. Each new rule is applied after restarting PHP.
- **Delete Rules** - A multi-selection for deleting redundant or unused rules.
- **Clear Cache for Selected** - A multi-selection for clearing the cache for specific rules.

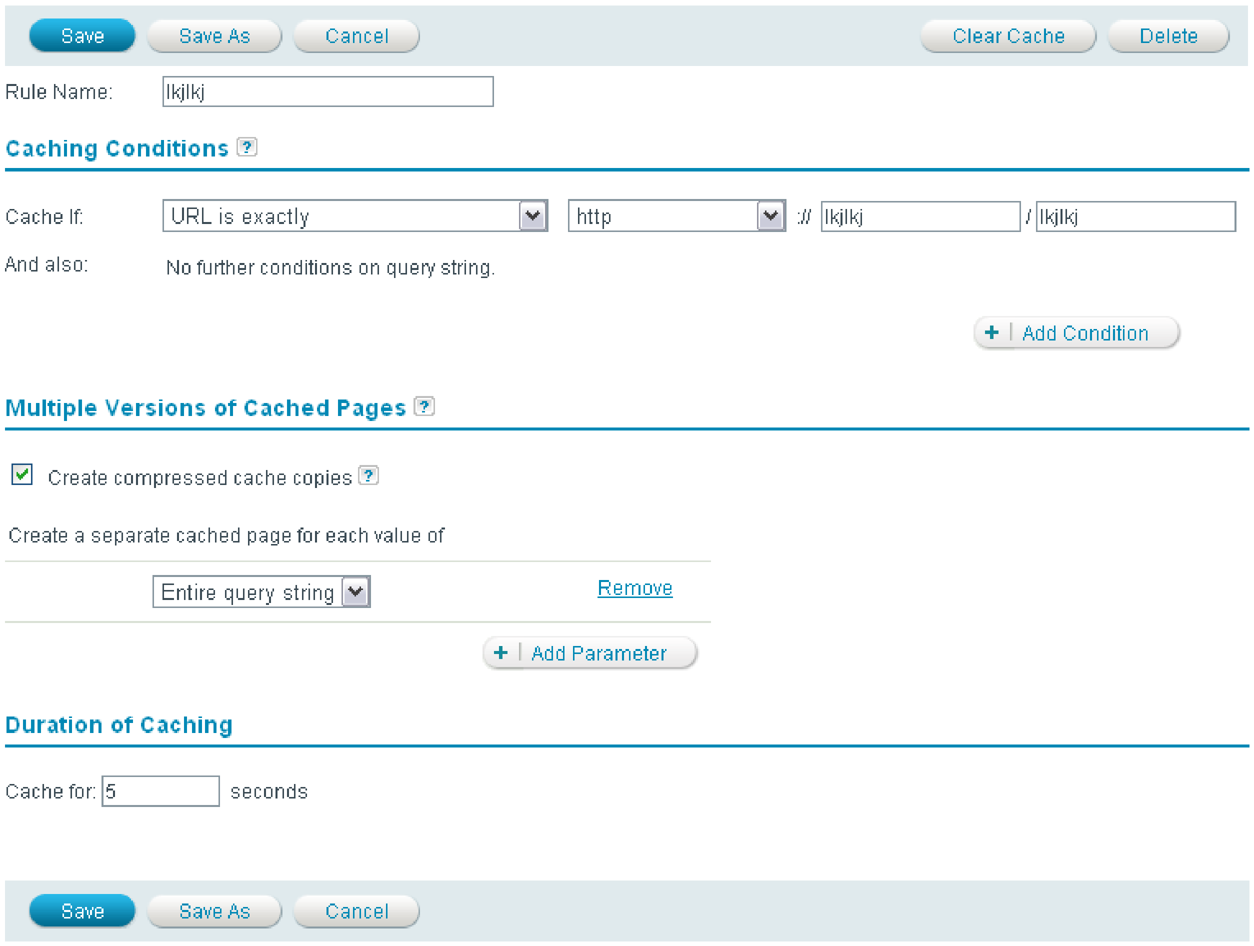
For each Rule you can:

- **Edit** - Open the rule for editing to modify settings by clicking on the rule.

To create a rule based on an existing rule's settings, open the rule, change the settings and use the "Save As" option to create a new rule.

Edit Rule (Caching)

The Edit Rule page is accessed from **Rule Management | Caching**, by clicking Add Rule or by clicking Edit next to a specific rule.



Rule Name: lkjlkj

Caching Conditions

Cache If: URL is exactly | http | :// lkjlkj / lkjlkj

And also: No further conditions on query string.

Multiple Versions of Cached Pages

✓ Create compressed cache copies

Create a separate cached page for each value of

Entire query string

Duration of Caching

Cache for: 5 seconds

Caching Information

Rule Name - The unique name you give the rule. This name appears in the list in **Rule Management | Caching.**

Caching Conditions

Define a Web page to cache by building the URL in the entry fields. Use the URL of the page you want to cache and define how long to cache the page and the caching format.
You can also add conditions that can further pinpoint what to cache. You can choose to cache when "**all of the following are true**" - i.e. all the conditions occur or when "**at least one is true**" - i.e. trigger caching when at least one of the conditions occur.
The conditions that can be applied to a rule are:

- **Get** - This refers to the parameters after the' ?' in the URL. Use this condition to cache specific URLs, for example, caching a page with the URL: http://localhost:81/index.php?page=gallery. When the rule is applied, only the specified page is cached out of all the pages on http://localhost:81/index.php.
- **Server** - This global variable can be used to determine server (Apache) parameters. The most common usage of SERVER variables is to use the headers that are sent in the request (i.e., variables that begin with HTTP - for example, HTTP_USER_AGENT), that can be used to define rules based on browser type.
- **Session** - This global variable originates from an active session and can be used to cache (or specifically not cache) scripts if a specific variable exists (or has a value) in the active session.
- **Cookie** - This global variable stores information that is sent to the server from the browser. A cookie can be used to cache banners such as "Related Search" banners (which usually take time to compile), by displaying pre-cached banners according to the information in the cookies.

Note:
You can only cache URLs that display static content with a long rendering time or dynamic content that you want to display statically according to time/parameters.

Multiple Versions of Cached Pages

- **Create Compressed Cache Copies** - This option allows you to disable the creation of a gzip-compressed version of each cached page as long as it is larger than 1KB. You should normally leave this option checked.
- **Create a separate cached page for each value of** - If you want to manage a different cache version according to an additional parameter, you can choose to create a separate cached page

for all the caching conditions (Entire Query String) or add one at a time each separate query string (Get, Server, Session or Cookie).

Duration of Caching

Define the cache's lifetime

- **Lifetime -** The duration of the cache content, after the set amount of seconds the cached content will be replaced by new content.

For more information and rule examples see: Working with Page Caching.

Note:
URL caching conditions can be defined using Perl-Compatible Regular Expressions (PCRE). The pattern syntax is the same as the syntax used by PHP's preg_match() and other preg_* functions. For more information on the PCRE syntax, see http://devzone.zend.com/manual/reference.pcre.pattern.syntax.html.

Recurring Jobs

The Recurring *Jobs* page is accessed from **Rule Management | Recurring Jobs.**

The Recurring Jobs page is the central configuration area to configure jobs to run by URL. Running jobs by URL makes it easy to eliminate situations where the same file is run in multiple instances, such as off-loading tasks from the synchronous user request, to an external, parallel PHP process.

Note:

Zend Server also provides the ability to Schedule Jobs using the *Job Queue* API Zend Job Queue API .

To read more about the Job Queue, see Working with Jobs.

From this page you can:

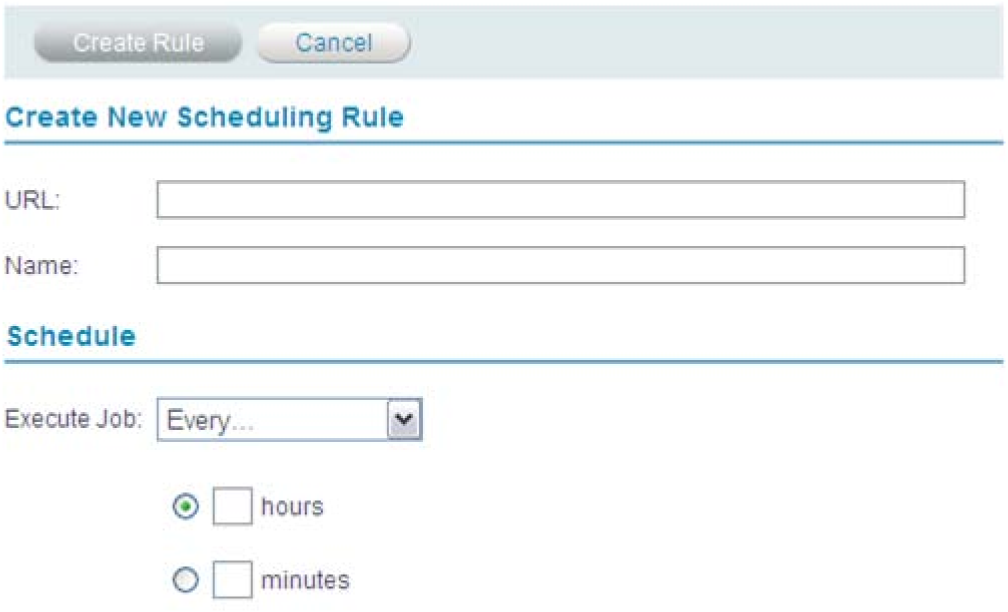
- **Filter** - Search for a specific job by name.
- **New Recurring Job** - Add a new job to the Recurring Jobs page. Each new job is applied after restarting PHP.
- **Suspend** - Temporarily stop the job from running while still saving the job definitions. Settings are applied after restarting PHP.
- **Resume** - Un-suspend a job. Settings are applied after restarting PHP.
- **Delete** - A multi-selection for deleting redundant or unused Rules.

For each Rule you can:

- **Edit** - Open the Rule for editing to modify settings. To**edit** a job, click on the **Job's ID** in the Job table.
- **History** - View the details of each time the job ran. To **view** a Job's history click on the **History** link next to a job in the Job table's Actions column

Edit Rule (Job Queue)

The Rule page is accessed from **Rule Management | Recurring** *Jobs*, by clicking New Recurring Job or by clicking on a specific rule to open the rule for editing.



URL:

Name:

Execute Job: Every...

[✓ hours] minutes

For more information and rule examples see: Creating a Job.

Rule Information

Create New Scheduling Rule Details:

- **URL** - the location of the job file you want to run
- **Name** - a descriptive name for the job.

Schedule Job Details:

Use the options to define when the job should re-occur based on an hourly, daily, weekly or monthly basis. Selecting an option will change the different parameters to allow you to define when the job will run. For example, choosing weekly scheduling will display options to set the job to run on a specific day of the week.

After completion of a job (it doesn't matter if it was succeeded or failed) the *Job Queue* re-schedules its next execution with the same values.

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**
Working with Code Tracing	**On** - Activates code tracing for events configured to create a trace file and for manually generating a trace file. **Off -** Code tracing will not be available at all.	
Working with 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.
Working with the 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"
Working with Zend Download Server	**On** - The specified file extensions can off-loaded to a separate server. **Off** - All file downloads are handled by the same Web server that runs the PHP.	This component is not relevant for Windows OS users.
Working with Zend Guard Loader	**On** - Scripts encoded with Zend Guard run. **Off** - Scripts encoded with Zend Guard **cannot** run.	
Working with 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.
Creating a Job	**On -** Activates the job queue, only when this component is on will *Jobs*	If you are using the Job Queue API in your code, always make sure that this option is

Component	Status	Comments
	run. This includes all Jobs either configured from the UI or by using the Job Queue API. **Off -** Jobs will not run.	running otherwise it could cause your code to fail with an "undefined method" fatal error.
Working with Monitoring	**On** - Event information, as defined in **Rule Management | Monitor,** is collected and displayed in **Monitor | *Events***. **Off** - Disables the Monitor component: Event information is not collected.	
Working with Optimizer+	**On** - PHP is optimized. **Off** - PHP **is not** optimized.	This component stores information and therefore has an additional action for clearing information.
Working with Caching (Page)	**On - Activates the page cache: URLs associated with caching rules are cached.** **Off - Disables the page cache: URLs marked to be cached are not cached.**	This component stores information and therefore has an additional action for clearing information.
Zend Session Clustering	**On -Session information will be centralized for all the servers belonging to the cluster.** **Off - Disable Session Clustering**	If you have Session Clustering running in your environment, do not turn off the component. Only turn the component Off if you do not use Session Clustering.

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

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.
The Extensions page is list of the extensions included with the Zend Server 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.

Monitor

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

From this page, you can define the different settings for configuring the Zend Monitor component.

This component is used to capture PHP *events* when they happen and to alert developers and system administrators. This can be done by using the events predefined in the Administration Interface or by using the API.

The Monitor page is a settings definition page for the Monitor component that provides event-based PHP monitoring.

Zend Studio Client Settings

Settings for using the integration with Zend Studio, to debug, profile and view event source code.

- **Automatically Detect Zend Studio Settings - On**
- **Automatically Detect Zend Studio Settings - Off**:
  When you click to turn 'auto-detect' off, the following additional fields will be displayed. Enter the correct information to enable integration with Zend Studio.
  - **Zend Studio IP address** - Manually configure the IP address according to the settings in Zend Studio or use Auto Detect.
  - **Use Browser IP Address check-box** - Automatically selects the Browser's IP Address.
  - **Zend Studio debug port** - Manually configure the port according to the settings in Zend Studio or use Auto Detect.
  - **Encrypt communication using SSL check-box -** increase the security protocol for communication.

Note:

Manual configuration settings should be based on the IP and port configured in Zend Studio's Installed Debuggers preferences page. The preferences page is accessed from **Window | Preferences | PHP | Debug | Installed Debuggers. (**Click 'Configure' to view and change the preferences).

The default port number is 20080.

Mail Server Settings

Define the action settings for rules that use the action 'Send Mail'. The Send Mail action sends an email that includes the event details and a link back to the Zend Server Administration Interface to the address(es) listed in the event rule.

If you do not plan to send email notifications for events, you do not need to configure these settings.

- **SMTP server address** - The server used to send mail.
- **SMTP server port** - The port used to send mail.
- **Sender's email address** - The email address that is displayed in the 'From' details of each message.
- **Administration Interface URL** - Use the valid Zend Server address, includes protocol (http or https), Server, and Port settings. The URL is included as a link in the event email.
- **Monitor Rules Default Emails** - A comma separated list of addresses. All emails sent by the monitoring module will be sent to these addresses unless a different address is specified inside a monitoring rule (in **Rule Management | Monitor**).

Note:
If you disable the email sending setting (in **Rule Management | Monitoring** by selecting the rule and changing the status), when the setting is reactivated all the email addresses will be lost and the email setting will be set to the default email address.

Zend Monitor Periodic Cleanup Settings

Define which events will be deleted in Zend Server's cleanup process, which is run every 24 hours to delete old events.

Note:
Any event that meets **either** of the below options will be deleted during the cleanup process.

- **Delete any event which did not occur in the last x days -** Define how long (in days) must have gone by since the event occurred for it to be deleted during the cleanup process. This applies to any event.
- **Delete closed events older than x days** - Define how old (in days) an event must be to be deleted during the cleanup process. This applies to closed events only.

Code Tracing Settings

Activation options for running a code trace.

- **Active** - Code tracing is available for events.
  Zend Code Tracing will collect trace information in Events that are defined for collecting trace information
- **Inactive** - Code tracing is not available for events.
  Zend Code Tracing is running but no new trace data will be collected even when a new event is triggered. The Manual Trace URL option (Monitor | Code Tracing) will remain active.
- **Standby** - Tracing is not available for events, but an event can temporarily activate tracing.
  The Zend Code Tracing will run in sleep mode and when an event defined to collect trace info is triggered, Code tracing will be active for a set period of time (as defined in the Monitor Rule).

Job Queue

The *Job Queue* page is accessed from **Server Setup | Job Queue**.

From this page, you can define the different settings for configuring the Job Queue Component.

This component is used to schedule *jobs* to run a PHP script. This can be done by creating a Job in Recurring Jobs or by using the API.

In a clustered environment all jobs will be run on a single dedicated server belonging to the cluster.

The Job Queue page is a settings definition page for the Job Queue Component that provides Job scheduling capabilities.

The following Job Queue Daemon settings available:

- Default Job Queue Daemon address-
- Maximum number of concurrent running jobs -Set the amount of jobs that are permitted to run at the same.
- Days to keep completed and failed jobs in database - define how long you want to keep job related information.
- Retry Interval - Set the amount of time to wait before trying to re-run a failed job.
- Maximum number of retries - Set the amount of times to re-run a job that failed to run.
- Report job run time skew after **X** seconds of delay
- Report high job concurrency when job count is **X** below the limit

Administration Tab

Password and License

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

From this page, you can change your login password and update your license.

For information on updating cluster licences see Licenses and Registration.

Updating your License

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

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 Update License to apply the changes.
4. Click Restart PHP.

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

License Expiration

The Password and License page displays your order number and expiration date. In addition, 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 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.

Further information about your license/s can be found in your zend.com account.

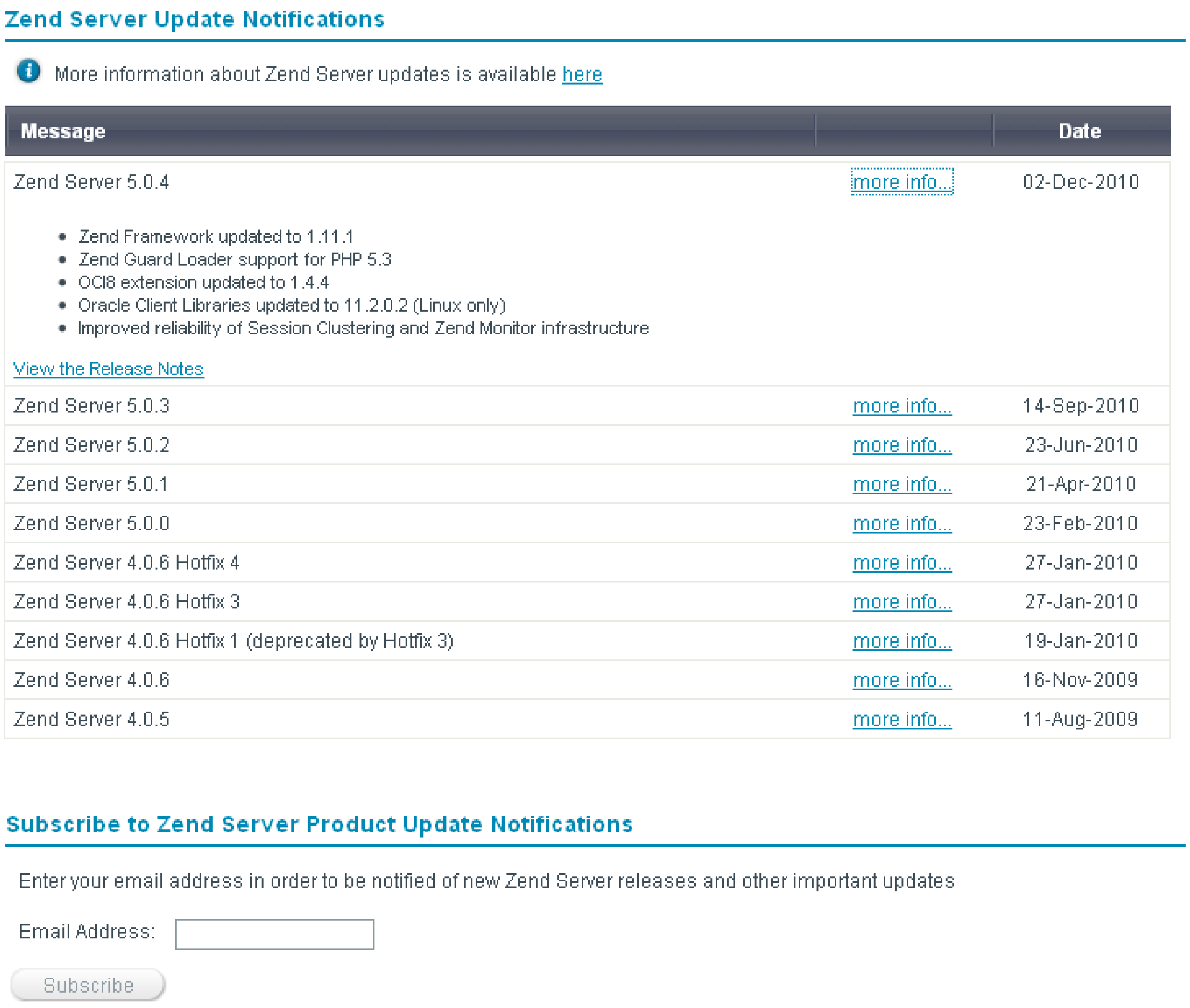
Update Notifications

The Update Notifications page is accessed from **Administration | Update Notifications.**

The Update Notifications page displays important product update notifications from Zend. You can read brief information about each update from Zend Server's administration interface, along with a link to the detailed release notes and to download the update.

Zend Server checks for updates each time you log in to Zend Server, or every 72 hours, provided that you are connected to the Internet.

When new update notifications are available, you will see a message at the bottom of the screen with a yellow 'warning' triangle next to it.



Zend Server Update Notifications

More information about Zend Server updates is available here

Message		Date
Zend Server 5.0.4 • Zend Framework updated to 1.11.1 • Zend Guard Loader support for PHP 5.3 • OCI8 extension updated to 1.4.4 • Oracle Client Libraries updated to 11.2.0.2 (Linux only) • Improved reliability of Session Clustering and Zend Monitor infrastructure View the Release Notes	more info...	02-Dec-2010
Zend Server 5.0.3	more info...	14-Sep-2010
Zend Server 5.0.2	more info...	23-Jun-2010
Zend Server 5.0.1	more info...	21-Apr-2010
Zend Server 5.0.0	more info...	23-Feb-2010
Zend Server 4.0.6 Hotfix 4	more info...	27-Jan-2010
Zend Server 4.0.6 Hotfix 3	more info...	27-Jan-2010
Zend Server 4.0.6 Hotfix 1 (deprecated by Hotfix 3)	more info...	19-Jan-2010
Zend Server 4.0.6	more info...	16-Nov-2009
Zend Server 4.0.5	more info...	11-Aug-2009

Subscribe to Zend Server Product Update Notifications

Enter your email address in order to be notified of new Zend Server releases and other important updates

Email Address:

The update notification information is provided through an Atom feed which you can subscribe to using any standard feed reader program or service. The update notification feed URL is http://www.zend.com/news/server-updates/feed.

If you would like to be notified of Zend Server updates, enter your email address in the Subscribe to Zend Server Product Update Notifications area and click **Subscribe**.

For information on upgrading see your system's procedure in Upgrading Zend Server Cluster Manager.

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.

If you are accessing Zend Server that is running on a different machine you will not be able to see the Zend Controller unless you installed an additional instance on your machine.

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

The following text describes how to work with Zend Server . 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.	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 , Download Server and Page Cache).	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.	Use the local and remote debugging features in Zend Studio for Eclipse.
Working with Local Debugging	How to configure the Debugger to debug and profile code running with Zend Server.	Use the local debugging feature in Zend Studio for Eclipse.

Task	Description	Outcome
Working with Firewall Tunneling	How to configure communication between Zend Server and Zend Studio for Eclipse when there is a firewall.	Debug PHP applications through a firewall using Zend Studio for Eclipse.
Working with Zend Download Server	How to configure the Zend Download Server to manage large file downloads.	Create a list of files to be offloaded.
Working with Zend Controller	How to configure your Zend Controller and use it to activate components and benchmark URLs.	Use the Zend Controller. The configuration creates a start button in the system tray.
Working with Monitoring	How to configure monitoring rules.	Setup your Monitor component for development or production environments.
Editing Monitoring Rules	How to find and customize rule settings.	You will have a set of rules in place to monitor the activity of your PHP applications.
Working with Events	How to find and manage *events*.	Find events, view event details and change event statuses.
Working with Event Details	What to do with the Event Details report.	Understand the information provided and diagnose events using Zend Studio for Eclipse.
Working with Code Tracing	How to locate trace information for events and analyze. Define which events should collect trace information and how to trigger a trace.	The ability to perform root cause analysis on information collected about PHP performance.
Creating a Job	How to create a job to run a script.	You will have a set of *jobs* running in your environment to perform tasks that require scheduling or triggering from inside your code.
Managing Jobs	How to use Jobs to help manage your environment	Know how to implement Jobs for off loading processes.
Working with Caching	How to choose the right caching option for your needs.	You will be able to choose the right caching options.
Working with Data Cache	How to use the Data Cache API.	Implement the Data Cache API functions into your PHP code.
Working with	How to configure page caching rules.	Define page caching rules for URLs.

Task	**Description**	**Outcome**
Caching (Page)		
Working with phpMyAdmin to Manage MySQL	How to configure PHPMyAdmin to work with a MySQL database.	Manage your MySQL from PHPMyAdmin through a link in the Administration Interface.

Getting Started with Zend Server

Zend Server 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 is what to do after installation.

What to do After Installing Zend Server

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

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 locally, open a browser and enter the following URL:

   For Windows: http://localhost/ZendServer;

   For Linux: 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 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, you are prompted for the password you set the first time you opened Zend Server.

Configure Your Password

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

Result: Your password is created.

When you first run Zend Server, the registration screen is displayed. Define your Zend Server 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 webserver 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.

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 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

This section refers to the actual configuration workflow for using Zend Server 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's Administration Interface is the main control center for configuring your PHP and Zend Server components. After installing Zend Server, 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 components that should be "turned on" or "turned off". If you are planning to use functions related to Zend Server 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 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.
5. In **Rule Management | Monitoring** , define the rules to generate *events* in Monitor | **Events** . Additional Monitor settings to define behavior (such as integration with Zend Studio for Eclipse and Mail settings for forwarding event information outside Zend Server) can be found in Server Setup | **Monitor .**
6.

Restart PHP Message

The Restart PHP message appears whenever a change is made to setting in your clusters php.ini file. in order to apply the settings click the "Restart PHP" button. The changes will be applied to the php.ini files on your nodes that are associated to this cluster.

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 setting in your clusters php.ini file. in order to apply the settings click the "Restart PHP" button. The changes will be applied to the php.ini files on your nodes that are associated to this cluster.

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 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 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, filter, navigate and refresh logs.

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 Show 20 lines (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 Show 20 lines. 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 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.

Power users may edit the XML file in /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 Restart PHP .

   - 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 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).
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 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 Restart PHP .

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.

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 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 versionand 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:

```
UNIX, Linux, IBM i and Mac <install_dir>/bin/javamw.jar

Windows <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) 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 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 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 + Add to add the Host.
5. The changes are applied after you restart the Server Restart PHP

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 Restart PHP .

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 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 Firewall Tunneling

Tunneling is not available for MS Windows Operating Systems.

Tunneling provides a means to establish a persistent connection between Zend Studio and a remote server situated behind a firewall or NAT. After a firewall tunnel is created, instead of assigning more communication ports for the Debug / Profile Sessions with the remote server, all communication between Zend Studio and the remote server proceeds through the tunnel.

Note for Windows users:

Tunneling does not work when Zend Server is installed on a Windows-based server. However, you can perform remote debugging to a machine behind a firewall as long as the correct ports are open on both ends.

The tunnel communication port should be used in the following circumstances:

1. When debugging or profiling files on a remote server which is behind a firewall or other security device.
2. Establishing communication between Zend Studio and Zend Server when Zend Server is running on a remote server which is behind a firewall or other security device. The communication between Zend Studio and Zend Server facilitates the integration of Zend Server event reporting capabilities with Zend Studio’s editing, debugging and profiling features, and makes it more effective.

To set up a tunneling connection, several configuration settings must be defined, both in Zend Studio and on your server's debugger. This can be done through Zend Server , Zend Platform, Zend Core or your php.ini file.

Setting-up Tunneling in Zend Server

The following procedure describes how to define allowed hosts for firewall tunneling. Users define allowed hosts for firewall tunneling to create a list of IP addresses (of computers that run Zend Studio) that connect to Zend Server for debugging purposes, in situations when a firewall stands between the machines.

Before setting-up tunneling, verify that the IP of the machine running Zend Studio is set as an allowed host in **Setup | Debugger**.

Note:
The following instructions define how to configure Zend Server's side for tunneling. Additional Zend Studio configurations are required to complete the process.

To define allowed hosts for firewall tunneling:

1. In the Administration Interface, go to **Setup | Debugger**.
2. In the "Allowed Zend Studio Clients for Debugging" section, enter a valid IP address or specify a range of IPs using Wildcards (*).
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 + | Add to add the host.
5. The changes are applied after clicking Restart PHP

The IP or range of IPs are now allowed to connect to the server through the firewall to debug PHP code with Zend Studio.
To remove a specific IP from the list, mark the check-box next to the IP and click Remove.
Make sure that you have not specified the IP in the "Denied Zend Studio Clients for Debugging" area.

Setting Up Tunneling in Zend Studio

In order to properly setup a tunnel connection to Zend Studio you have to configure Zend Studio to allow Zend Server to establish a connection.
Zend Studio 7 users should follow the instructions in: http://files.zend.com/help/Zend-Studio-7/setting_up_tunneling.htm - External Link
Zend Studio for Eclipse 6 users should follow the instructions in:http://files.zend.com/help/Zend-Studio-Eclipse/setting_up_tunneling.htm

Working with Zend Download Server

This component is not relevant for Windows and IBM i Servers.

The Zend Download Server (ZDS) can be used to increase your Apache Web server's capacity by automatically taking over static downloads. This releases Apache processes to handle more dynamic requests.

Files can be sent through the ZDS in two ways:

1. **Automatically** - By configuring Apache to pass specific file types to PHP: For example, images, PDF files or any other relatively large static files. When the ZDS is loaded and configured, PHP passes the request to the ZDS, which takes over sending the file to the user.
2. **Manually** - By calling the Zend Download Server API functions from within your PHP script. This is useful for situations where you require some code or logic to run before allowing a download, such as authenticating users before permitting them to download.

Configuring Zend Download Server to Automatically Handle Files

This procedure describes how to configure your environment to handle certain file types with the ZDS. Although transparent to end users, this action is expected to increase server capacity.

To set up ZDS to automatically handle specific file types:

1. Locate the Apache configuration file that contains the PHP handler configuration by searching for the line starting with "AddHandler php5-script" (you can do this using *grep* or similar tools). This line defines the file extensions that are passed to PHP.
2. Add the extensions of the file types you want ZDS to handle at the end of this line.
3. Make sure that the same file extension is listed in the ZDS MIME type configuration file located in <install_path>/etc/zend_mime_types.ini. If not, add it and make sure to define the correct MIME type for it. The configured MIME type is sent as the value of the "Content-type" HTTP response header, which (according to the browser's individual settings) determines the browser's behavior when it receives the file.
4. Save your files and restart the Apache server.

All the defined files types are now handled by the ZDS (as long as this component is set to 'On' in the Administration Interface).

Sending Files Using the Zend Download Server API

This procedure describes how to call the ZDS API functions from within your code, to handle specific types of file downloads.

Although transparent to end-users, this action is expected to increase server capacity.

The following example demonstrates the logical flow of sending a file using ZDS. In this example, the download only proceed if the user is authenticated.

Example:

```
if ($user->isAuthenticated()) {

    zend_send_file($filename, $mime_types, $headers);

    // -- execution stops here --

} else {

    echo "Sorry, you are not authorized to download this file.";


}
```

If the user is not authenticated, the download does not begin and the following message is displayed: "Sorry, you are not authorized to download this file."

Working with Zend Controller

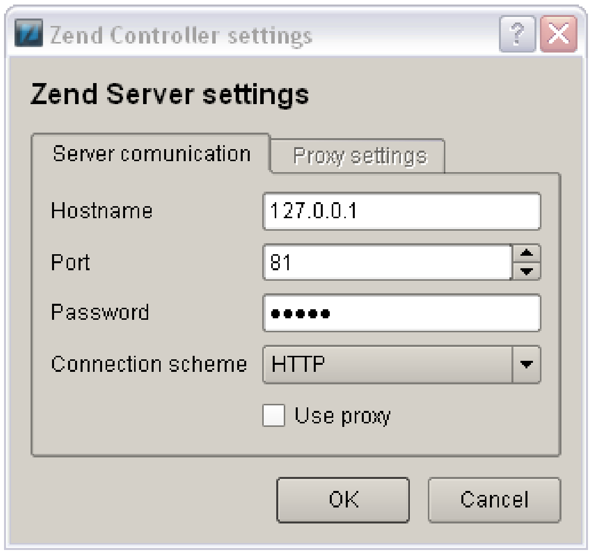
Initial Setup

The following procedure describes how to configure the Zend Controller's settings to communicate with Zend Server. 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 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 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 for communication purposes, where HTTPS is a secured connection protocol.



Hostname 127.0.0.1
Port 81
Password ●●●●●
Connection scheme HTTP
☐ Use proxy

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's Optimizer+ and Data Caching components. This can be done by turning the different Zend Server 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.

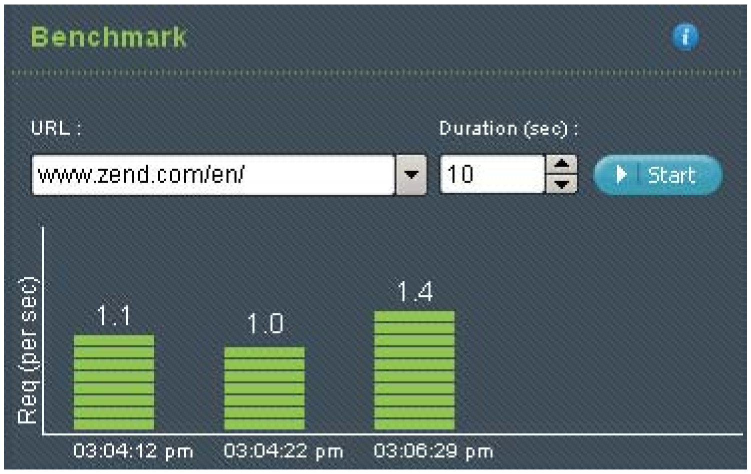
To run a Benchmark:

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 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 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.

Monitor

Working with Monitoring

To access the Monitoring page, go to **Rule Management | Monitoring**.

This page is the central management area for defining the conditions by which *events* are generated (as displayed in **Monitor | Events**).

Monitoring and management capabilities let you quickly detect, analyze and fix errors reducing time-to-resolution and avoiding costly downtime.

Creating Events

Event generation is an out-of-the-box feature. On installation, the Monitor component begins to collect and report information about events, according to the Monitor's default settings. The resulting events are displayed in **Monitor | Events**. To further enhance monitoring effectiveness, event thresholds can be customized. In a similar manner, thresholds can be gradually modified to not only reflect improvements in performance, but also to verify that problematic issues have been resolved.

Configuring Events

Events can be configured according to each environment’s specific requirements. The main configuration changes that should be performed relate to tuning rule values and defining a list of functions and PHP errors to monitor.

The following procedure describes how to configure event rules

To configure event rules:

1. Go to **Rule Management | Monitoring**.
2. Select a rule from the list. Hovering over the information icon displays a description of the selected event and the event’s parameters.
3. Click **Edit** to change the default settings according to your requirements.
   Each event type has different configuration options suited to the nature of the event.
4. Scroll to the bottom of the page and click **Save** to save changes.

The new settings are applied after you click **Restart PHP**.

To return to the main Monitoring page, click Cancel, Save or use your browser's Back button.

See Edit Rule (Monitoring) for more information about rule settings.

Disabling Event Rules

In some cases, there may be events that are either not applicable to your system or unnecessary. Events are disabled from the Monitor page. When an event is disabled, the event is not monitored and information is not stored for the event.

To disable event rules:

1. Go to **Rule Management | Monitoring**.
2. Select a rule from the list. Hovering over the information icon displays a description of the selected event and the event's parameters.
3. Click **Disable** to stop the Monitor from collecting and reporting information relevant to this event type.

This event type is no longer monitored.

Editing Monitoring Rules

To access this tab go to **Rule Management | Monitoring**.

This section describes how to activate, change and deactivate a monitoring rule in your environment.
In order to define rules the Zend Monitor component must be installed and running.

To Deactivate a Rule

The following procedure describes how to deactivate a rule. By default, all rules are set with predefined conditions. As long as the Monitor component is running, theses rules will be active and *events* will be generated based on the pre-defined parameters in each rule. A disabled rule will appear "grayed-out" but you can still edit

To deactivate a rule:

1. In **Rule Management | Monitoring** select a rule or rules that are not "grayed-out" and at the bottom of the page in the drop-down list, choose the option "Disable Rules".
2. Click Change to save.
3. To apply the changes click Restart PHP.

The rule will now stop generating Events based on the rule's settings. Disabled rules will not generate trace file information.

To Activate a Rule

The following procedure describes how to activate a rule. By default, all rules are set with predefined conditions. As long as the Monitor component is running, theses rules will be active and events will be generated based on the pre-defined parameters in each rule. A disabled rule will appear "grayed-out" but you can still edit

To activate a rule:

1. In **Rule Management | Monitoring** select the rule or rules that are "grayed-out" and at the bottom of the page in the drop-down list, choose the option "Enable Rules".
2. Click Change to save.
3. To apply the changes click Restart PHP.

The rule will now start generating Events based on the rule's settings.

To Change a Rule

The following procedure describes how to change a rule. By default, all rules are set with predefined conditions. Each rule has different parameters that can be configured based on what the rule monitors. See Advanced Diagnostics with Zend Server for more information on the available rule types.

To change a rule:

1. In **Rule Management | Monitoring** click on a rule.
   The rule will open for editing.
2. change the settings according to the rules configuration options.
   Each rule has different settings.
3. Define the actions that will be applied to the rule.
   If you define an action for an event it will show-up in Rule Management | Monitoring in the Event Actions column.
4. Click Save Changes to save.
5. To apply the changes click Restart PHP.

The rule will now start generating Events based on the new settings.

Once you have set actions for an event (tracing and send mail), you can deactivate these actions from the main page (Rule Management | Monitoring) by selecting a rule or multiple rules from the list and from the bottom of the page in the drop-down list, choose the applicable option:

- Enable Emailing Action
- Disable Emailing Action
- Enable Code Tracing Action
- Disable Code Tracing Action

Working with Events

To access this tab go to **Monitor |** ***Events*****.**

An event is a collection of runtime-related information collected by Zend Server . This information is collected when an event is triggered, according to the Monitor component's rule settings. An event indicates that something happened in your environment that exceeded your definitions and the standards of how you want your PHP code to run.

From this page you can:

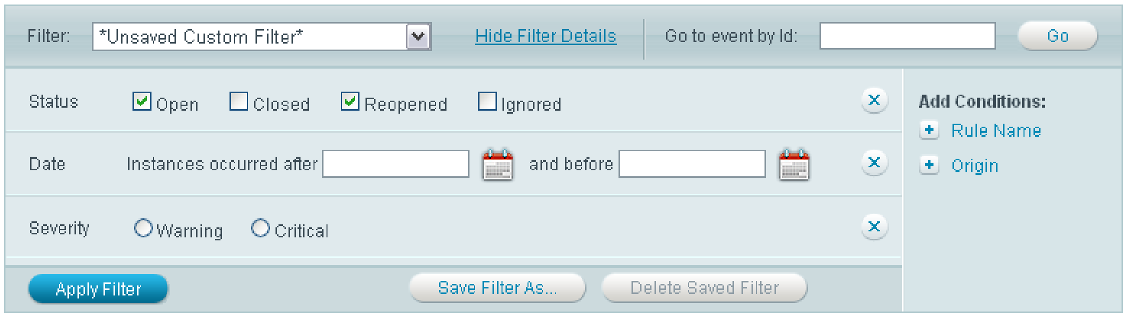
- Find Events
- View Event Details
- Change an Event's Status

See Working with Monitoring to learn how to define which events you want Zend Server to generate and under which conditions.

Finding Events

By default, all events are displayed in the Events page (**Monitor | Events**). The filter option allows you to reduce the number of events displayed in the Events table to easily locate specific events. There are seven filtering categories for events:

- **Occurred Before...** - Only show events that first happened before the specified date.
- **Occurred After...** - Only show events that first happened after the specified date.
- **Rule Name** - Only show rules of a specific type.
- **Occurred at....** - Filter to display the results for each server.
- **Severity** - Filter to display Severe or Warning events.
- **Event Type** - Filter by 'Event Type'.
- **Status** - Show events by status: Open, Closed, Reopened or Ignored.


Filter: *Unsaved Custom Filter*

Status: [✓ Open] Closed [✓ Reopened] Ignored

Date: Instances occurred after ___ and before ___

Severity: Warning Critical

The following procedure describes how to create a filter for events. Events can be filtered by one or more of the categories

To filter events:

1. In the Events page, click "Show Filter Details".
   A filter controller is added beneath the filter. To hide the filter controller, click "Hide Filter Details".
2. On the right hand side click on a filter type to add.
   Each time a different filter is added, an additional selection area will appear in the filter area. You can add as many filters you like. Each time a filter is selected, a new selection area is added.
3. To delete a specific filter option, click the delete button (X).
4. Set your preferences according to the filter type. For example, set a date range for the Date filter.
5. When you have finished configuring the filter, click "**Save Filter As...**" to name the filter and save it for future use.
6. Click "**Apply Filter**" to re-display events according to the filter parameters.
   If you do not want to save the filter directly, click "Apply Filter" without saving.
   Only events matching the filter are displayed.

Use "Delete Saved filter" to remove the filter. Filter information is saved until the next login: You can safely navigate to other pages and tabs without losing your filter settings.

Viewing Event Details

The following procedure describes how to open an event to view the event's details. Event Details are the actual context of the event. Use these event details to locate information for root cause analysis of the issue.

To open an event:

1. Go to **Monitor | Events** to display the Events page.
2. In the Events page, go to the Event table.
3. Locate an event that interests you (the filter option can be used to decrease the number of events listed in the table).
4. Click on the event's ID number in the ID column.

The Event Details page opens.

Changing an Event's Status

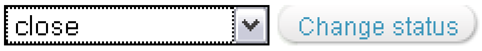
This procedure describes how to change an event's status. The event status is used to handle the information gathered by an event in a way that is similar to an issue tracking system. For example, if you know that a certain problem has been solved, you can close the event or, if several events of the same type are triggered, you can choose to ignore them.

The possible statuses are:

- **Open** - The basic status of an event.
- **Closed** - Closes the event (i.e., changes the event status to closed). If this event occurs again, it is reopened .
- **Ignored** - Ignores future instances of this event (i.e., changes the event status to ignore). Therefore, a new event is not created if the same event occurs again.
- **Reopened** - A closed event that was opened again because it reoccurred after the event was closed.

Important:
You can select events with different statuses and apply the same new status to them.

To change an event's status:

1. In the Events page, select the relevant event/s from the events list.
2. Scroll down to the end of the page and select the relevant status from the Status list 

.
3. Click **Change Status** to apply the changes.

The status of the selected event/s is changed. You can also change an event's status from inside an event details page by using the "Change status..." controller at the bottom of the report.

Working with Event Details

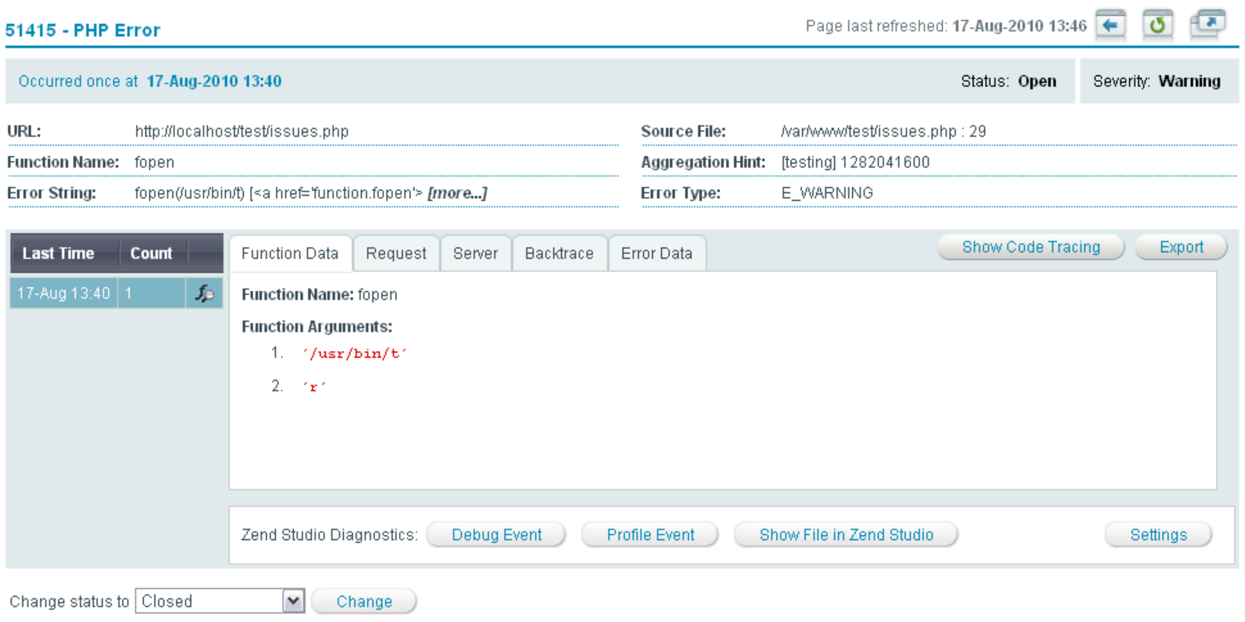
To access this page, go to **Monitor |** ***Events*** and select an issue ID from the list.

This page shows aggregated details regarding the events that triggered a specific rule. The details include audit trail information and options for investigating and resolving issues.

For a more detailed description of events and issues, see Monitor.

The actions in the issue details page can be divided into two categories:

- Basic Tasks: Simple actions that can be applied to the page.
- Advanced Tasks: Advanced actions for diagnosing and managing the content of the issue.


51415 - PHP Error

Page last refreshed: 17-Aug-2010 13:46

Occurred once at 17-Aug-2010 13:40 | Status: Open | Severity: Warning

URL:	http://localhost/test/issues.php	Source File:	/var/www/test/issues.php : 29
Function Name:	fopen	Aggregation Hint:	[testing] 1282041600
Error String:	fopen(/usr/bin/t) [<a href='function.fopen'> *[more...]*	Error Type:	E_WARNING

Last Time	Count
17-Aug 13:40	1

Function Name: fopen

Function Arguments:

1. '/usr/bin/t'
2. 'r'

Change status to: Closed

Basic Tasks

The following actions can be applied in an issue.

- Return to the Events page.

  To return to the Events page, click Return
- Refresh an issue's details.

  To refresh an issue, click Refresh
- Detach an issue

Detaching an Issue

This procedure describes how to detach an issue. Detaching an issue is the process of opening an issue in a new popup window. This feature can be used to display several events at the same time.

To detach an event:

1. Go to Monitor | Events.
2. Select an issue from the table and open the issue (click on the event ID).
   The event is displayed.
3. In the opened issue, click Detach .
   The issue is opened in a new popup and the main page returns to the Events list.

Once an issue is detached, it cannot be re-attached. This process can be repeated to open several issues at the same time.

Advanced Tasks

Issues contain different types of information and actions that allow you to detect the source and drill down to investigate what caused a problem. In some cases (and depending on the type of event that happened), the information also indicates the solution. For example, issue information includes the exact line of code that has a problem and the nature of the problem.

Basic Details

The first layer of issue details are the basic details. These details are used to identify an issue with high-level characteristics. Every issue includes the following basic details:

- **ID and Rule Name** - The ID number is a unique identifier for a specific issue. The Rule Name states the rule (defined in **Rule Management | Monitoring**) that triggered the event. The ID number can be used to locate a specific issue using Go to event by Id: in **Monitor | Events**. The Rule Name indicates the rule that triggered the event: Several issues can be triggered by the same rule. Knowing the rule name makes it easy to find the rule if you want to modify the rule settings (**Rule Management | Monitoring | Edit**).
- **Occurrence Info** - An issue is a collection of aggregated events: This detail specifies how many times this event occurred since the first time.
- **Status** - The Status indicates the state of the issue. In some cases, the Status changes automatically. The Status can be used to locate events in the Events page: Use a filter to display the specific status.
- **Severity** - Currently there are two severity levels, Critical and Warning. These levels reflect your preferences, in terms of the importance of the event.

General Details

Each issue is an aggregation of one or more events with common, predefined characteristics. These common characteristics are displayed in the General Details section. Therefore, you can assume that each event that occurred for this issue has at least these details in common. This is the first step in identifying the source of the event and understanding the circumstances surrounding the event.

Group Details

Grouping is yet another additional aggregation layer applied to an issue. Inside a single issue, events are divided into groups according to the time they occurred. A new group is created only if there was no activity for at least five minutes. If a new event occurs after five minutes have passed, a new group is added to the issue.

- **Group Details** show how many groups were created for an event, when a group was created and how many events are in a group. When the event is also a relative or quantitative event, an additional "Runtime" column is added to show the time it took to perform the action. This helps you determine if there are differences in the event's runtime, identify which events took longer and what else may have happened when the event occurred (for example, did the event occur during a peak load, or during a DB query, etc.).
- **Group Drilldown -** Clicking on a group 13-Jan 13:10 | 17 changes the contents of tabbed display to show the details collected for that specific group of events.
- **Diagnostic Actions -** These actions can be applied to each group to investigate what caused the event. The diagnostic actions are Debug, Profile and Show File in Zend Studio. All these actions run on the server defined in ▼ Settings. By default, the settings are set to run diagnostic actions on the originating server (the server on which the event was created). You can change the settings to run on a different server.

Note:
When you use these actions, make sure that you have Zend Studio on the remote machine, access to the remote server and that the remote server is an allowed host. For more details, see Error: Failed to Communicate with Zend Studio.

Action Buttons

Export

The Export option creates a file with a .zsf prefix that contains all the information collected for the event. This includes and Basic and General Details of a specific Group.

If Zend Code Tracing data was collected for the group (see Setting a Monitor Rule to Run a Code Trace), it will also be included in the exported file.

This file can then be viewed in Zend Studio's Integration section. The "Export" feature allows you to export all the data related to an event as a file that can be opened in Zend Studio for further analysis. Examples of analysis are, executing a debugging session based on the event context information, or viewing and analyzing relevant Code Tracing information, again only if such information was captured by the server.

The flexibility of creating a file means that you can easily email the file to be opened in a Zend Studio that does not have any connection to a server. This can be useful for diagnosing sensitive trace information and preserving security protocols.

For instructions on exporting Zend Code Trace information to a Trace file see Exporting Trace Information.

Some events may not include this option, This only happen in events that collect information that is not relevant for debugging.

Show Code Tracing

This option is only available if you have Zend Code Tracing set for the Event, see Setting a Monitor Rule to Run a Code Trace.

The **Show Code Tracing** option opens the Zend Code Trace Tree and Statistics views for the specific trace information collected for the event you are in.

Change Status

The Change Status option provides an easy way to manage issues. Issues that are assigned an appropriate status can be efficiently filtered to control which issues to display in **Monitor | Events**.

The following summary describes what happens to issues when they are created and what happens when new events are added to issues that have a changed status.

- New events are created with the status New.
- If the event status is Closed and a new issue occurs, the event status is changed to Reopened.
- If the event status is Ignored and a new issue occurs, the event status does not change. However, the system continues to collect information about the event.

To change an issue's status:

1. Go to **Events** and select an issue from the list.
2. In the issue details page, scroll down to the bottom and use the drop-down list to select a status.
   Closed - Changing an issue's status to closed will close the issue until a new event of that type occurs and then the issue will automatically be changed to Open.
   Open - Changing an issue's status to Open can be applied to issues that are in status, closed or ignored.
   Ignore - Changing an issue's status to Ignored will keep the same status even when new events occur.
3. Click Change to apply changes.

The status in the basic details area and in **Monitor | Events** will be changed accordingly.

Working with Code Tracing

Code Tracing records execution activity in real-time. This provides an in-depth diagnostic tool that will allow you to drill-down to the function level to view actual performance related information and statistics.

In order to work with Code Tracing you first have to have the Code Tracing component installed and activated (see the Installation Guide for information on adding components and to see Working with Components for information on the components in your system).

Setting a Monitor Rule to Run a Code Trace

This procedure describes how to define monitor rule settings to in addition to the event related information also collect trace information when the event is generated.

To set a monitor rule to save trace data:

1. Go to **Rule Management | Monitoring**.
2. Select a rule from the list.
3. Click **Edit** to open the rule for editing.
4. Scroll down the page to "Step 2: Event Action"
5. Select the one of the options and click + | Add.

   The Tracing options are:

   Save Trace Data - Generate a trace file each time the event is triggered.

   Activate Trace Data Collection for... -Generate a trace file for a given time frame each time an event is triggered.
6. Scroll to the bottom (or the top) of the page and click **Save** to save changes.

The next time the event is triggered trace information for that specific occurrence will be collected. The trace information can be viewed in **Monitor | Code Tracing** or directly from the event (see important note below).

Important Note:

A single issue can contain aggregated information about more than one instance of an occurrence. This information is also grouped according to when the event occurred. When Code Trace information is collected it is collected for a specific time and therefore will be associated to a specific group inside the event for more information see "Viewing Trace information Inside an Event".

Manually Triggering a URL Trace

This procedure describes how to manually run a code tracing on a specific URL. Running a code trace on a specific URL will capture the current activity at the time the trace was triggered.

To manually trigger code tracing on a specific URL:

1. Go to **Monitor | Code Tracing**.
2. Enter the URL/IP in into the "Trace URL field and click Trace
   The URL has to be running on the Server running Zend Server.

The trace information will be collected and added to the list in the Code Tracing page.

Running Code Tracing on Another Machine

You can run Code Tracing on another machine as long as the Zend Code Trace component is installed and running on it. However, when running a trace on a remote machine the trace information is collected on that machine therefore, Instead of running the trace from a remote machine always run a trace on the specific machine you need. If you do want to run a trace on a remote server after all, make sure that the server's IP is an allowed host in **Server Setup |** **Debugger**.

Viewing Trace Information

There are two ways to view trace information according to how the trace was generated. Manually generated traces can be viewed in the Code Tracing page **Monitor | Code Trace**. Traces that were generated alongside an event can be viewed in the Code Tracing page **Monitor | Code Trace** and also from the issue's details.

Viewing Trace Information Inside an Event

This procedure describes how to view trace information that was collected at the time a monitor event was triggered.
Trace information for a monitor event is only collected if the specific event was defined to trigger a code trace.

To view trace information inside an event:

1. Go to **Monitor | *Events***:
2. Select an event from the list by clicking on the event's ID.
   The Issue Details page will be displayed.
3. In the occurrences section, navigate to the time that the trace was supposed to run. clicking on that group will add an additional "View Trace" button to the display. Clicking on it will open the Code Tracing view where you will be able to see the trace information.

See Code Tracing Tree and Code Tracing Statistics for more information about the Zend Code Tracing view.

Viewing all Trace Information

This procedure describes how to view trace information for traces that were triggered manually or by an event.
The Code tracing view is populated after trace information has been collected either from an event or manually.

To view trace information:

1. Go to **Monitor | Code Tracing**.
2. Select a trace from the list by clicking on the trace's ID.
   The Code Tracing view will open in a new window.

Use the Code Tracing Tree and Code Tracing Statistics tabs to navigate through the information.

Exporting Trace Information

This procedure describes how to export trace information. Exported trace information can be used to be imported into Zend Studio for further diagnostic information when you do not have access to the server or if you have been sent trace information from another person.

To export Zend Code Trace information:

1. Go to **Monitor | Events** and select an event that has been set to collect trace information.
2. In the event details page, click **Export.**
   A dialog will open prompting you to save the file to a location.
   The file will be created with a .zsf prefix for example "event_trace_20_25_0.7843.1.zsf"

The Zend Code Tracing file can be sent to any Zend Server user and imported as an event file into Zend Studio. For more information on how to import a trace file into Zend Studio see The Zend Studio Help

Deleting Trace Information

This procedure describes how to delete trace information.

1. Go to **Monitor | Code Tracing**.
2. Select a trace or multiple traces from the list by clicking the trace's check box.
3. scroll to the bottom of the page and click delete.

The trace information will be permanently deleted. New trace information for the same URL will continue to be recreated as long as the setting in the rule is not changed. To change the rule's settings see "Setting a Monitor Rule to Run a Code Trace".

Jobs

Creating a Job

The Zend Server Job Queue component provides the ability to off-load script execution using *Jobs*. A Job is in essence a command to execute a PHP script according to the specific parameters you give it. The *Job Queue* component manages the timing and execution of jobs in the system.

Jobs can be created to facilitate a wide range of situations that require you to asynchronously run PHP code. Here are a few examples:

- Action Based Jobs for backend activities which are not a part of the HTTP request-response interaction with end users and need to be executed asynchronously.
- Scheduled Jobs that are set to execute according to a predefined schedule.
- One-time jobs that need to be run for any reason.
- Improving website performance by offloading processes to a different server.
- Optimizing long script execution by deferring running scripts to low peak times.
- Unifying script usage by referring to a job script outside the application code instead of repeating the same code for the same functionality. This also provides the added advantage of being able to implement changes throughout the application be editing a single script (instead of editing each individual instance of the script throughout the application's code).

Working with The Job Queue API

The Job Queue API is a set of functions that allow you to create, define and manage the execution of PHP code using a job.
There are three types of Jobs you can create with the Job Queue API

- **Queued Jobs -** triggered as a direct result of an end-user activity
- **Recurring Jobs** - usually defined by the system's programmer. They are usually related to the system's maintenance and are not triggered by an action made by an end-user. The most common existing system to handle such jobs is cron which is usually a part of the operating system on most UNIX-like systems.
- **One-Time Deferred Tasks (jobs)** - usually a result of an end-user interaction, and may be deferred in order to optimize the system's load (e.g. cleanup or report generation tasks may be pushed to off-hours) but may also be a part of the application's business logic.

Queued Jobs

This procedure describes how to create a Job that will be triggered as a result of end-user activity using the Job Queue API.

To create a Job that will be triggered as a result of end-user activity:

1. Open your existing code.
2. Isolate the part of the code that should be executed as a job.
   This code should be a viable script that performs some sort of action that can be run at a later time without disturbing the general functionality (such as sending an e-mail, confirming a credit card etc.).
3. Put the code into a file and name the file. The location and the name of the file will become the value of the URL parameter therefore always provide descriptive names and place job files in the same location.
4. In the original code, replace the code you removed with a call to the Job Queue function *createHttpJob.*
5. In the function, pass additional GET parameters in the job's URL query string, to handle different data.
6. Publish the fixed code to your webserver.

The next time the code is used the job will be triggered inside the code. To find out how the job ran go to **Monitor | Jobs.**

To see what jobs are currently running go to **Monitor | Queue Statistics.**

For additional information on handling jobs see Managing Jobs.

Example:

The following example shows what the original code looks like. How it will look like using the Job Queue API, and what the new Job file looks like.

Before converting to a job:

```
<?php
function validate_credit_card_number($string)
{
    if(preg_match('/^([0-9]{4})[\-|\s]*([0-9]{4})[\-|\s]*([0-
9]{4})[\-|\s]*([0-9]{2,4})$/', $string))
    {
        return TRUE;
    }
    else
    {
        return FALSE;
    }
}
// here we call the function to validate the credit card validate_cr
edit_card_number($_POST['credit_code']);
```

Instead of calling the function from inside the script we will create a job that will be executed when we specify it to run. The script will be replaced with a call to the Job Queue API as follows:

The application script after modification:

```
<?php


$cc = $_POST['credit_card'];


$q = new ZendJobQueue();


$ts = date('Y-m-d H:i:s', time()+10);


$id=$q-
>createHttpJob('/jobs/validate_credit_card.php',array('credit_card'=
>$cc),array('name'=>'Credit


card validation using a single job execution scheduled to run after
10 seconds','schedule_time'=>$ts));


if(!$id){


exit(1);
```

The new job script:

```
The job script:
<?php
function validate_credit_card_number($string)
{
    if(preg_match('/^([0-9]{4})[\-|\s]*([0-9]{4})[\-|\s]*([0-
9]{4})[\-|\s]*([0-9]{2,4})$/', $string))
    {
        return ZendJobQueue::setCurrentJobStatus(ZendJobQueue::OK);
    }
    else
    {
        return ZendJobQueue::setCurrentJobStatus(ZendJobQueue::FAILE
D);
    }
}

$params = ZendJobQueue::getCurrentJobParams();

validate_credit_card_number($params['credit_card']);
?>
```

Recurring Jobs

This procedure describes how to create a recurring Job. A recurring job will be executed periodically based on a defined schedule.

Creating recurring jobs from API is usually useful when the application's workflow requires the creation (and usually subsequent deletion) of recurring tasks following some user interaction. For example, in a feed aggregator, it might make sense to create a recurring job that hourly pulls updates for each new feed added by the user. Another example might be a reporting system in which users can create and delete daily, weekly or monthly reports.

To create a recurring job:

1. Follow the instructions in Queued Jobs to create an *createHttpJob* job using the API.
2. Pass the *schedule* option as part of the options parameter to describe the recurrance schedule.

The *schedule* option is a CRON-like expression that consists of string with 5 fields separated by spaces. The fields define the minute, hour, day of month, month and day of week (in this order) in which the job will run.

In each field, you can use one of the following notations:

- A single number (with valid ranges listed below)
- An asterisk ('*') to designate "any" (e.g. to run a on every day of the month, put '*' in the 4th field)
- A comma-separated list of values (e.g. to run a job on Sundays, Tuesdays and Thursdays, put '0,2,4'in the 5th field)
- An interval specified by '*/n' where n is the interval (e.g. to run a job every 2 hours, use '*/2' in the2nd field). To be accurate, this means "on every hour that evenly divides by two – meaning 2,4,6,…)

Example:

The following example represents a job that will run every day at 3:15am:

```
$job = $jq->createHttpJob('http://localhost/jobs/feed/405', null,
array(
'schedule' => '15 3 * * *'
```

Ranges for each field:

Field	**Range**	**Comments**
Minute	0 - 59	
Hour	0 - 23	In 24 hour format, where 0 is midnight
Day of Month	1 - 31	29,30 and 31 will only work for months of that length
Month	1 - 12	
Day of Week	0 - 7	Sunday is either 0 or 7

One-Time Deferred Tasks (jobs)

This procedure describes how to create a one-time deferred task. In some use cases, it makes sense to defer a certain task to a later time. For example, many applications have clear peak hours (e.g. between 8am and 11pm, or like in many Intranet applications, during office work hours).
If these applications need to perform some off-line processing, it might make sense to defer some of the heavy processing tasks to off hours (in our examples to late night or early morning hours), in order to maximize the efficiency of hardware utilization.

To create a time deferred task:

1. Follow the instructions in Queued Jobs to create an *createHttpJob* job using the API.
2. Pass a date/time string as the *schedule_time* option, as part of the options array passed as the 3rd parameter to *createHttpJob().*

Note

The format used in date() to pass the execution time – this is an SQL-like ‘YYYY-MM-DD hh:mm:ss’format (e.g. “2009-06-25 23:45:00” for June 25th 2009 at 45 minutes past 11pm).

Zend Job Queue is not designed to execute jobs exactly on the specified time. For example, if the queue is limited to execute 10 jobs concurrently (more on that later on), and 1,000 jobs are scheduled for the exact same time – jobs will have to wait until other jobs finish. You should consider the *schedule_time* option as a request not to run a job before this time.

Example:

The following example shows a time-deferred task that has been scheduled to run a process at 2:00am.

```
$options = array(

'schedule_time' => date('Y-m-d h:i:s', strtotime('tomorrow 2am'))

);

$jq->createHttpJob('http://localhost/jobs/formproc.php', $_POST,
$options);
```

Note:

the format used in *date()* to pass the execution time – this is an SQL-*like ‘YYYY-MM-DD hh:mm:ss’ format (e.g. "2009-.6-25 23:45:00" for June 25th 2009 at 45 minutes past 11pm).*

Using the Job Queue from the Zend Server Administration Interface

The Zend Server administration interface provides tools for creating recurring jobs. These jobs have a set schedule and will be run at specific times.
This procedure describes how to create a recurring job.

To create a recurring job:

1. In the administration Interface go to **Rule Management | Recurring Jobs**
2. Click New Recurring Job to open the New Scheduling Rule page.
3. Enter the following rule related information:
   a. **URL** - the path indicating to where the code is for the job to execute and the server on which to run it.
   b. **Name** - Optional, a name describing the job
   c. **Application** - Optional, the name of the application the job is related to. This information can be used for grouping jobs.
4. Define when the job should run by using the time options (Hourly, Daily, Weekly or Monthly). Selecting an option will change the parameters displayed. For example selecting daily will display hour and minute options whereas, weekly will display the days of the week.
   When all the settings are properly defined, the "Create Rule" button will change from Grey to Blue.
5. Click Create Rule to save the changes.

As soon as the new Recurring Job is saved, it will be put in the Job Queue at the time specified in the Rule.
To find out how the job ran go to **Monitor | Jobs.**
To see what jobs are currently running go to **Monitor | Queue Statistics.**
For additional information on handling jobs see Managing Jobs.

Note:
The Jobs and Queues Statistics pages display information about all the Jobs in your system, including Jobs triggered by the Job Queue API.

Managing Jobs

The following information refers to managing Zend *Job Queue*, *jobs* from the Zend Server Administration Interface.

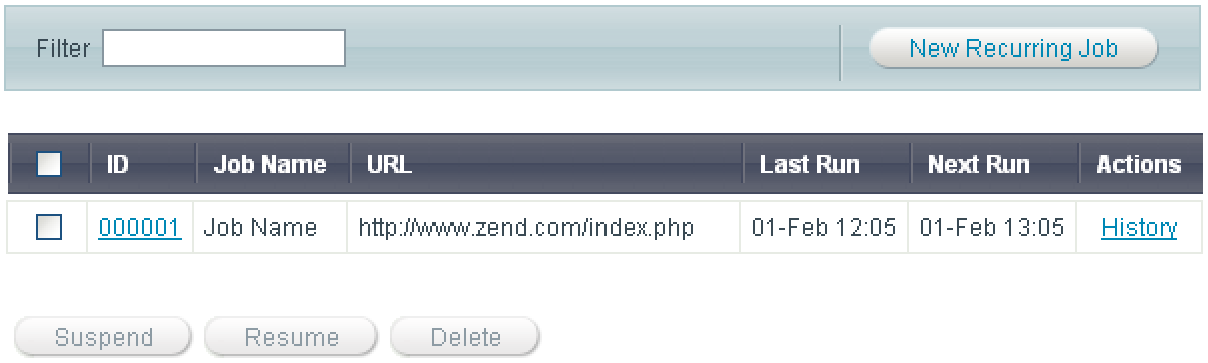
There are three different ways to manage jobs:

1. Manage Job Rules
2. Manage Job History

Managing Job Rules

The following procedure describes how to manage recurring jobs. These jobs are created in **Rule Management | Recurring Jobs** or by using the Job Queue API.

Once a job is created it is added to the jobs table.



☐	ID	Job Name	URL	Last Run	Next Run	Actions
☐	000001	Job Name	http://www.zend.com/index.php	01-Feb 12:05	01-Feb 13:05	History

Job Management Actions

Action	**Description**	**Instructions**
Filter	Search for a specific job by name.	Enter at least 2 characters of the name of the job into the Filter field to start filtering the list of jobs by name.
New Recurring Job	Add a new job to the Recurring Jobs page. Each new job is applied after restarting PHP.	See Creating a new job
Suspend	Temporarily stop the job from running while still saving the job definitions. Settings are applied after restarting PHP.	Select one or more jobs from the list by clicking the check-box next to the job and click Suspend.
Resume	Un-suspend a job. Settings are applied after restarting PHP.	Select one or more jobs from the list that were suspended and click Resume. suspended jobs are grayed-out and have the word suspended in brackets next to the Job Name.
Delete	A multi-selection for deleting redundant or unused Rules.	Select one or more Jobs from the list and click Delete.
Edit a Rule	Open the Rule for editing to modify settings. To **Edit** a job, click on the **Job's ID** in the Job table.	To edit a rule, click on the rule's ID number to view the rule in edit mode. Make sure to click "Update Rule" after making the changes.
History of a Rule	View the details of each time the job ran.	To **view** a Job's history click on the **History** link next to a job in the Job table's Actions column. The Jobs page (**Monitor | Jobs)** will be displayed.

Managing Job History

The Jobs page is accessed from **Monitor | Jobs.**
the Jobs page displays information on all the jobs that have run, are running and are scheduled to run including jobs that were created with the Zend Job Queue API.

There are a few ways that Zend Server allows you to manage the jobs listed in the jobs page:

1. **Global Settings**: In **Server Setup | Job Queue** you can use the option define when to clear completed and failed jobs.
2. **Manually**: Using the Delete option in **Monitor | Jobs** allows you to manually delete one or more jobs from the list. The deletion process removes the job and its history completely.
3. **Filtering**: The filter allows you to view a selection of jobs by category, this is especially useful when there are a lot of jobs in the system. There is a predefined filter that allows you to filter by failed jobs and you can add an unlimited number of filters.
4. **Additional Information**: View the job's details to view additional information such as extended status information; headers, output and if the job while running generated an error.

Filtering Jobs

To access the *Jobs* page go to **Monitor | jobs.**

The filter allows you to view a selection of jobs by category, this is especially useful when there are a lot of jobs in the system. There is a predefined filter that allows you to filter by failed jobs and you can add an unlimited number of filters.

The following prcedures describe how to create and maintain filters.

To create a filter:

1. In **Monitor | Jobs** click "**Show Filter Details**" to expand the filter dialog.
2. Add conditions to the filter by clicking on the plus sign next to a condition in the "Add Conditions" area.
3. Modify the conditions according to each specific condition's filter options.
4. Click Save Filter As... and in the dialog enter a name for the filter and click save. The filter will be added as a "User Defined" option in the **Filter list** and can be applied be selecting it whenever necessary.

You can also use an existing filter to create another filter by opening it and using Save Filter As....

To apply a filter:

Select a filter from the filter list to automatically display the results.

To delete a filter:

1. Select the filter from the list (it has to be a "User Defined" filter, default filters cannot be deleted).
2. Click "**Show Filter Details**" and Delete Saved Filter to permanently remove the filter.

Cache

Working with Caching

This procedure describes the different caching capabilities available to you when using Zend Server.
Caching is the process of storing data or pre-rendered web output that can dramatically reduce the time to present results to the users.
The available caching capabilities are:

- Data Cache API
- Zend Framework Cache API (External Link)
- Page Cache

API versus Interface

The first step in choosing a suitable caching option is to discern who should be setting-up the caching.

The decision what caching option to use, is also based on the type of content you are interested in caching. Any long running operations such as query results, can be cached using an API as long as the information is relatively static (i.e. does not frequently change). For example if you are using a web service to collect weather information you can use the API to cache the information and refresh every few hours and gain a performance boost without compromising the accuracy of your information. When planning to develop a PHP application, long processes should be pinpointed in advance in order to implement caching in an early stage.

On the other hand if you have entire web pages that include static content such as home pages, online stores and catalogs you can cache the entire page using the Page Cache. When you have pages that include a mixture of static information and personalized information you can in essence cache per user but that should be done carefully as it may cancel out the caching performance gain.

The last point in deciding is, ease of use. Using the Data Cache and the Zend Framework Cache API requires programming skills to insert the API in the code and a deep knowledge of how the PHP application works. Page Caching also requires a deep understanding of how the PHP application works but does not require any programming skills and the entire caching process can be done from the Zend Server Administration Interface (Rule Management | Caching).

Data Cache API

The Data Cache API is a caching option that is integrated into Zend Server. In terms of value the option provides high performance in memory caching. As it is integrated into Zend Server you also gain a certain amount of control such as the ability to turn on/off the component and manually clear the cache with a click of a button. The Data Cache API is designed with an APC compatibility layer ensuring that extendibility with most APC compatible applications.

How it works:
Zend Data Cache is a set of API functions enabling a developer to store and manage data items (PHP strings, arrays and other data) and even output elements in either disk-based cache or shared memory cache. Zend Data Cache allows for precision guided caching when Page Caching is not an option. The provided API is easy-to-use on existing code, and in many cases a developer can skip existing code sections by simply wrapping them with caching APIs.

Page Caching

Page Caching is ideal for cases when you want to avoid programmatic intervention to implement, configure and manage caching capabilities. The integration into Zend Server allows complete control over the caching activity and with relative ease you can add, remove and change caching rules as well as empty the cache with a click of a button.

How it works:
Zend Page Cache allows caching of entire PHP web pages, or more correctly put, entire HTTP responses sent out from PHP. Page Caching permits dramatically improving the performance of pages on web applications (sometimes running as high as 10x, 20x and even 100x times faster), while maintaining dynamic capabilities through an elaborate system of caching rules that could be based on request parameters and user session data.

Zend Framework Cache API

The Zend Framework Cache API is a standardized open source caching solution. This grants you the freedom to not only configure it based on your needs but also to contribute extensions and fixes and benefit from contributions from other developers. The Zend framework API has pluggable backends so that you can create a generic implementation and swap the backend with different solutions such as memcache, APC and the Data Cache API. Using the Data Cache API provides an additional advantage as it allows you to use an option that is integrated into Zend Server gain a certain amount of control such as the ability to turn on/off the component and manually clear the cache with a click of a button.

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.

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

Working with Caching (Page)

To access the Caching page, go to **Rule Management | Caching**.

The Page Cache is used to speed-up recurring executions of PHP scripts in your application. This is achieved by caching the PHP output (HTML) for specific URLs on first execution, to reuse the cached data for subsequent calls.
Cache behavior is defined using a flexible rule system that allows you to maintain the dynamic capabilities of your applications.

As opposed to other caching alternatives (Zend Server Data Cache and Zend Framework Zend Cache), the Zend Server Cache does not require any code changes and can be easily applied to existing applications. Moreover, while other caching solutions still run some code on recurring executions, the cache does not run any code to display the cached content, which results in improved performance.

Creating URL cache rules with Zend Server is a two-step process. In step one, you define the basic URL and conditions to apply. In step two, you define the cache duration and output options.

The following procedure describes how to create a cache rule. Click here to see how to create a copy of a cache rule

To create a Cache rule:

1. Go to **Rule Management | Caching**
2. Click Add Rule to open the New Rule page.
3. Name the rule.
   Make sure the name is descriptive and easy to remember: This name will appear in the main Caching page.
4. Enter the information according to the following steps and click Save to apply the changes:

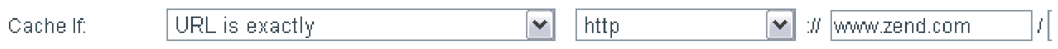
Caching Conditions

1. Use the fields to define the URL that you want to cache.

   A URL can be an exact URL or a representation of a pattern of URLs using Regular Expressions (external link), which can be either case sensitive or case insensitive.

   Example:

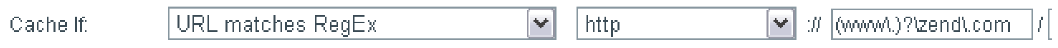
   Exact URL

   Cache If: URL is exactly | http :// www.zend.com /

   This representation sets the Page Cache to cache the URL http://www.zend.com/index only.

   URL pattern using Regular Expressions

   Cache If: URL matches RegEx | http :// (www\.)?\zend\.com /

   Note:
   Using regular expressions consumes more time and CPU than using exact URLs.

   This example sets the Page Cache to cache the following:

   - **URL matches regex** - Any URL that matches this pattern is cached.
   - **Scheme** - Only URLs that begin with 'http' are cached (the alternatives are 'https or 'either').
   - **Host -** The host name and port part (optional) of the URL. By using a regular expression, you can specify whether to cache URLs that begin (or do not begin) with "www".

     For example: (www\.)? - Indicates that the URL may or may not begin with 'www.\'.
   - **Path** - the path and query part of the URL.

     For example: /.* - Indicates that any string after the host name 'zend.com/articles/' is acceptable. To be precise, it represents 0 or more of any character.

2. Click + Add Condition to create additional caching rules based on HTTP requests and session parameters.

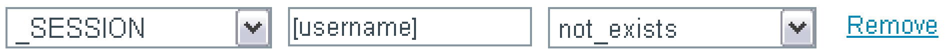
   There is no limit to the amount of conditions that can be added.

   Example:

   How can we configure the system to use only cached content when a user is not logged in?

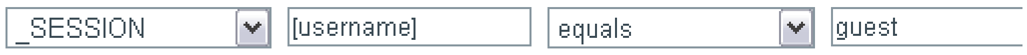
Usage Scenario: Websites (portals, news sites, forums...) that provide different content to premium users and non-registered users.

Assuming that the _SESSION '*username*' parameter is used to store the name of a currently logged in user, create a rule based on this parameter, as follows:



This sets the Page Cache to cache content only when the _SESSION parameter 'username' is not set. Subsequently, users that do have the _SESSION 'username' set are presented with a live version of the page.

If necessary, we can extend this limit to include all users whose 'username' _SESSION parameter equals 'guest' as follows:



Note:
Square brackets are not required for non-nested variables when referring to superglobal variables in caching conditions. If you do not use square brackets, the system adds them automatically after you click "Save".

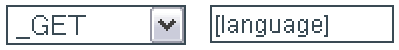
This completes creating a Cache Rule.

Multiple Versions of Cached Pages

1. Choose to create compressed cache copies.
   This option allows you to disable the creation of a gzip-compressed version of each cached page, as long as it is larger than 1KB. You should normally leave this option checked.
   By default, Zend Server creates a compressed version of each cached page and stores it alongside the original version. This compressed version is sent to browsers that support gzip compression (all modern browsers support this) instead of the uncompressed version. For typical HTML, XML or other text-based outputs, using the compressed version can save about 90% of the bandwidth and improve page load times. In addition, the compressed copy is saved in your cache so the CPU-intensive process of real-time compression is not required. However, if you are caching a PHP script that outputs binary data (for example JPEG or PNG images, ZIP or EXE files, PDFs, etc.) which cannot be further compressed, you should un-check this option to avoid redundant processing. For more information, see: Page Cache.
2. Click to create different cached copies according to specific values.
   This creates more than one version of the page in the cache, based on specific conditions.

Example:

The following example demonstrates how to create different copies of cached information, based on the _GET parameter 'language'.



_GET [language]

This sets Caching to create a different copy for each different value of _GET 'language' (for the content that was cached based on the rules defined in steps 1 and 2).

This example demonstrates how to use Caching for multilingual pages that handle the same content in different languages.

Caching Duration

Set the cache duration in seconds. After that time, the cache is refreshed and a newer version is created.

For example, 600 seconds is ten minutes.

This completes the last step of creating a Cache Rule.

Scroll to the bottom of the page (**Rule Management | Caching**) and click "Save" to save the rule information and "Restart PHP" to activate the rule.

To edit a rule, go to **Rule Management | Caching** and click Edit next to the rule you want to edit.

To clear the information in the cache for a specific rule, go to **Rule Management | Caching** and click Clear next to the rule you want to edit.

Creating a Copy of a Rule

The following procedure describes how to create a new rule based on an already existing rule. This option can be used when a new rule with settings similar to an existing rule needs to be created such as for different URLs with the same caching requirements.

To copy a Cache rule:

1. Go to **Rule Management | Caching**
2. Double click on the rule name on which you want to base the new rule.
3. Change the settings.
4. Click "**Save As**"

   A new Dialog will open prompting you to enter a new name for the rule.
5. Enter a name and click "**Save**".

A new rule will be added to **Rule Management | Caching.**

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 Installer includes this component as part of the installation process in Windows and Zend Server Community Edition. Download the Linux 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
- 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 /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: Windows

If you already have phpMyAdmin

When you install Zend Server, 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 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, 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 the phpMyAdmin installation is added in the Zend Server 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.

In this section:

Components

About Components

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

The components are:

- Zend Debugger - The Zend Debugger communicates with the Zend (PHP) Engine to retrieve runtime information and present it in Zend Studio for root cause analysis.
- Zend Optimizer+ - The Zend Optimizer+ component speeds up PHP execution via opcode caching and optimization.
- Zend Guard Loader - The Zend Guard Loader is used in order to run PHP scripts that are encoded with Zend Guard.
- Zend Data Cache - The Zend Data Cache component provides a set of PHP functions to improve performance, by storing data in the cache.
- Zend 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.
- Zend Monitor - The Zend Monitor component is integrated into the runtime environment and serves as an alerting and collection mechanism for early detection of PHP script problems.
- Zend Page Cache - The Zend Page Cache component is used to cache the entire output of PHP scripts, without changing the PHP code.
- ZDS (Zend Download Server) - The ZDS (Zend Download Server) component improves Apache's scalability by managing static content downloads and passing them to a dedicated process optimized for parallel downloads (not supported in Microsoft Windows).
- Zend Job Queue - The *Jobs* Component is used to streamline offline processing of PHP scripts.
- Zend Code Tracing - Code Tracing enables real-time execution flow recording in Production Environments.
- Zend Session Clustering - Scales environments to a cluster.

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.

Comprehensive caching takes the improved application responsiveness offered by Zend Server and expands it to an entire cluster – those optimizations include accelerating PHP execution, caching page content, and optimizing file and content downloads to improve the overall user experience

Zend 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.

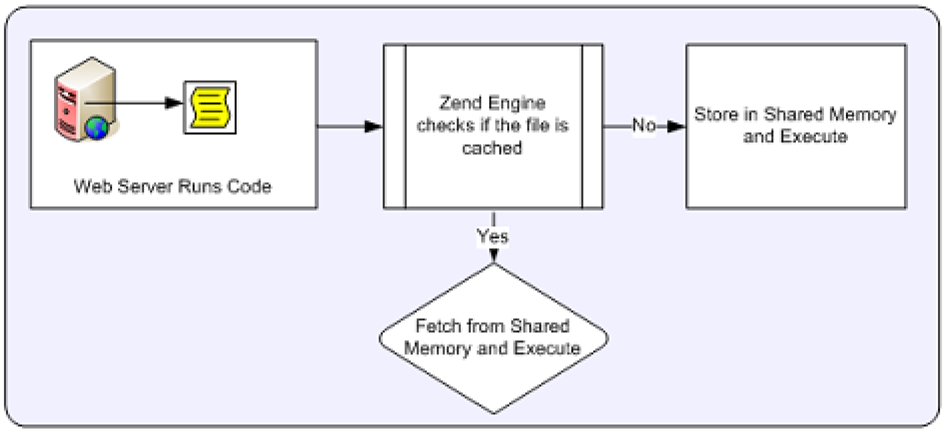
The Zend Debugger API communicates with the Zend (PHP) engine to reveal PHP runtime information such as variables, call stack and environment information. This information is then displayed and set up in Zend Studio to enable server side debugging, profiling and code coverage.

Zend 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.


Web Server Runs Code
Zend Engine checks if the file is cached
No
Store in Shared Memory and Execute
Yes
Fetch from Shared Memory and Execute

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).

Zend 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

Zend 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

Zend 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.

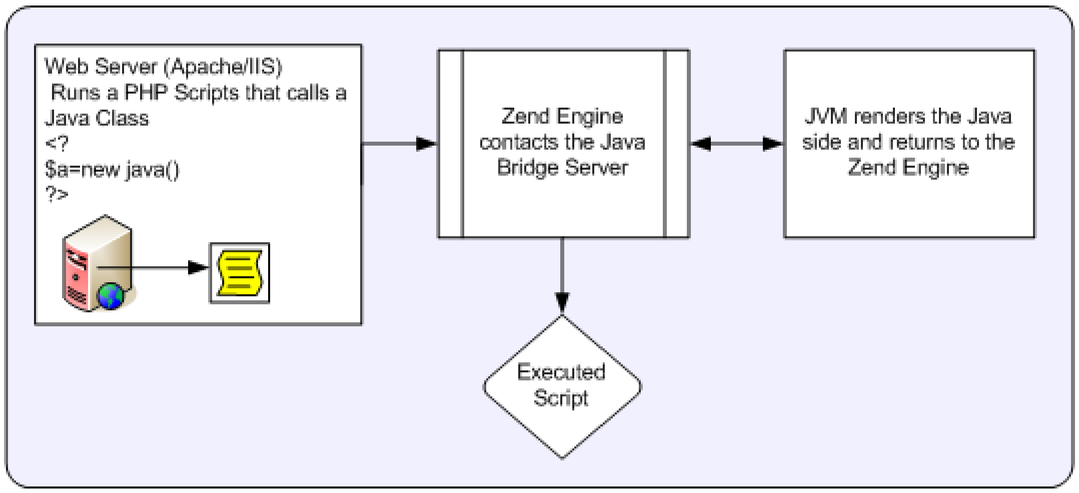
More information about JRE and the latest updates can be obtained from SUN Microsystems's website.

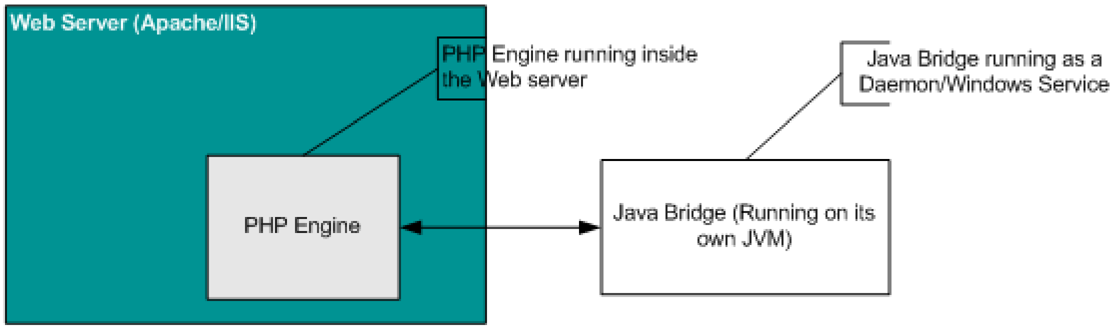
The Java Bridge PHP extension adds functions that allow you to instantiate new Java classes from inside your PHP script. Once a Java class is instantiated, the Java Bridge gets a message from the Zend Engine to execute the Java code. The Java Bridge executes the script and returns the results to the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

Zend Server 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) and the installation method format (EXE, DEB, RPM). 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:


Web Server (Apache/IIS)
Runs a PHP Scripts that calls a Java Class
<?
$a=new java()
?>
Zend Engine contacts the Java Bridge Server
JVM renders the Java side and returns to the Zend Engine
Executed Script


Web Server (Apache/IIS)
PHP Engine running inside the Web server
Java Bridge running as a Daemon/Windows Service
PHP Engine
Java Bridge (Running on its own JVM)

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.

Zend Monitor

The Zend Monitor component is integrated into the runtime environment and serves as an alerting and collection mechanism for early detection of PHP script problems.

The Zend Monitor is a Zend Server component that integrates into the PHP runtime environment and watches for various *events* such as errors, failing functions, slow scripts, database errors, etc. When an event occurs, the Zend Monitor collects and reports all the relevant debugging information. This information can then be used to perform root cause analysis.

What is an Event?

Events are governed by rules created in **Rule Management | Monitor**. Rules define the nature of an event and the parameters for capturing event related information in an application. Events are only created when the monitor component is running (**Server Setup | Components**).

By definition, an event is a collection of information that can be used to investigate what caused the rule to trigger an event. The information collected varies according to the rule type.

Aggregation

If events are triggered by the same rule and have similar characteristics – i.e., filename, URL, line etc – they are aggregated into a single issue. If they do not have similar characteristics, a separate (new) issue is created.

Inside a single issue, events are divided into groups according to when they occurred. A new group is created only if there is no activity for at least five minutes. If a new event occurs after five minutes pass, a new group is added to the issue. The new group includes all the events that occurred, as long as five minutes without activity has passed.

Changing Statuses

This section describes what happens to issues when they are created and what happens when a new event must be added to an issue after the issue's status has changed.

- New events are created with the **New** status.
- If an event is closed and a new issue occurs, the event is changed to the **Reopened** status.
- If an event is ignored and a new issue occurs, the event does not change its status. However, the system continues to collect information for the event.

Zend Page Cache

The Zend Page Cache component is used to cache the entire output of PHP scripts, without changing the PHP code.

The Zend Page Cache improves PHP application performance by caching the entire output of PHP scripts (HTML, XML, etc.), while still maintaining dynamic capabilities through an elaborate rules system. Rules are configured in the Administration Interface.

Page caching extends the concept of caching files and applies it to pages. Caching by page facilitates the ability to eliminate situations where the same file is used in multiple instances, such as when the same file is used to redirect to several pages.

When to Cache Pages

Pages should be cached when their content is stable and does not require frequent changes. You can cache any PHP generated output including, HTML, XML, and images (if the images are generated by PHP, such as graphs and charts).

Compression should be used to cache content such as HTML, XML and plain text, but is not recommended for caching binary output.

When Not to Cache Pages

Caching is not recommended for files that have constantly changing output, such as clocks, timers and database queries.

Compression should not be used for images, PDF files, .exe files, ZIP files or any other compressed binary formats.

Note:

Zend Page Cache only caches GET and HEAD HTTP requests. To comply with the HTTP RFC, POST requests are never cached.

All cached content is stored in a hashed directory structure on disk. The location is defined by the directive *zend_pagecache.save_path*.

Caching Alternatives

Web pages that contain sections that continuously change can also be cached. This partial page caching solution can be accomplished by applying the Data Cache API to the portions of code that do not change. Data caching is an intermediate solution to provide a partial performance boost that can sustain the accuracy of changing content.

To find out more about this alternative, go to Data Cache

Zend Download Server

This component is not relevant for Windows and IBM i Servers.

The ZDS (Zend Download Server) component improves Apache's scalability by taking over static content downloads and passing them to a dedicated process that is optimized for parallel downloads (not supported on Windows servers).

The ZDS enhances Apache's static download capabilities. This releases Apache processes to handle more dynamic requests. There are two ways to define which files will be serviced by the ZDS: Either via the API or via an external configuration file (mime_types).
A single ZDS process can handle substantially more concurrent downloads, compared to Apache's capabilities.

The Zend Download Server is a PHP extension that forks an HTTP server separate from Apache to be used for serving large files of given types by redirecting regular requests from your Apache to a separate server.

The Zend Download Server is a PHP extension that forks an HTTP server (separate from Apache) to serve specific types of large files, by redirecting regular requests from your Apache to a separate server.

The Zend Download Server offloads the limited Apache process activity to avoid the situation where a connection is open for a long time to serve large downloads, which in turn prevents the Apache from serving requests that can be handled immediately. The concept is similar to the express lane in a supermarket: A separate "lane" is made available for large downloads, which frees up the other lanes for smaller downloads.

Based on the settings in the mime_types file, specific file extensions are immediately identified when the request is received: These downloads are rerouted to a separate server.
This essentially creates a separate process which functions as an HTTP daemon, dedicated to downloads.

The mechanism is a PHP extension that creates a daemon to handle the rerouting when the extension loads. We also provide a mime_types file and an API for explicitly directing content (a file or buffer) to be handled by the ZDS (in parallel to using the mime_types file, depending on your preferences).

Zend Job Queue

Job Queues provide offline asynchronous processing of tasks and activities that can be run independently of the end user experience and/or assigned to a particular machine for execution.

The *Jobs* approach is used for streamlining offline processing of PHP scripts. It provides the ability to delay execution of “heavy” parts of web applications that originate from direct user interaction with a Web Server. As result it may significantly improve response time and reduce Web Server load.

Any PHP script can create a job which will be executed later, separately from a current HTTP request. The job itself is just an additional HTTP request to another PHP script that can be done on the same or different Web Server.

As opposed to “fire and forget” systems,(like cron), the *Job Queue* is a job management system that provides advanced capabilities such as:

- Keeping track of batch jobs, including their status, execution time and output
- Different schedule strategies based on time, job priorities and dependencies
- Run-time statistics
- Web-based management *GUI*

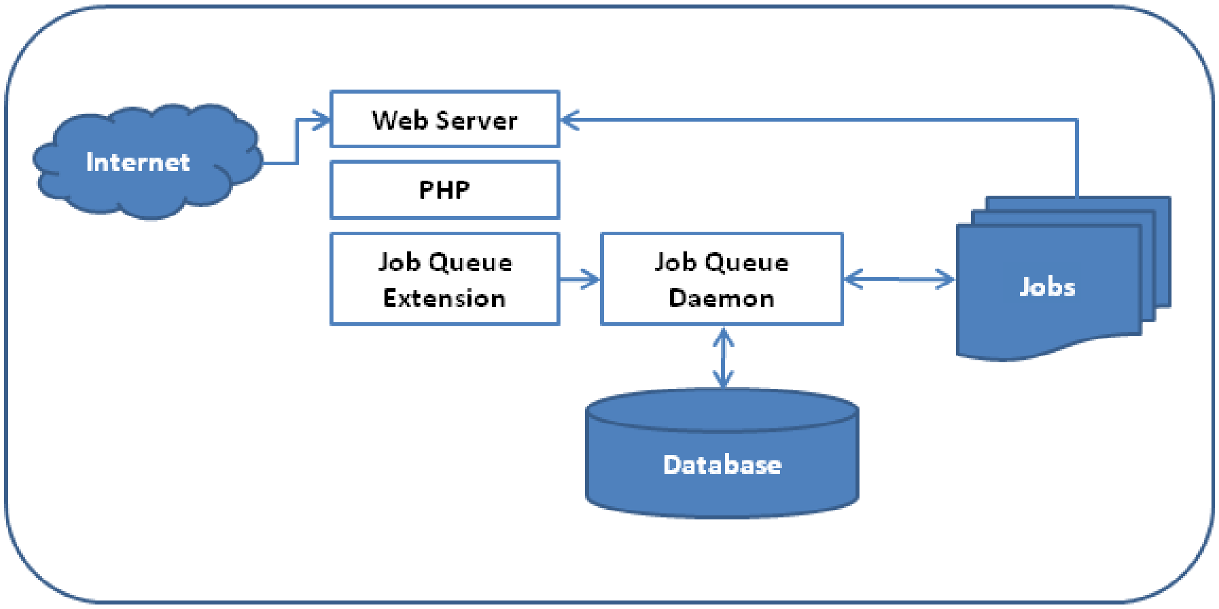
The functional diagram of the Job Queue is shown on the following picture.

Concepts and Terminology

- Job - a task defined by a URL, a set of parameters and additional properties which is to be executed by the Job Queue system. Each Job is executed once and scheduling rules can be defined to trigger periodical job creation for the same task.
- Scheduling Rule - an entity defining the periodic re-occurrence of jobs of a specific type. Each scheduling rule defines both the schedule and the job properties such as URL, parameters and priority. All jobs created by a specific scheduling rule share the same properties. this is the basis on which Recurring Jobs are created. Scheduling Rules are created either through the GUI and can also be hard-coded using an API.
- Recurring Jobs - are jobs that were generated based on a scheduling rule

Architectural Overview

The Zend job queue is comprised of three main parts, the Job Queue Daemon, the Job Queue Client and one or more servers that execute the jobs.



The Job Queue Daemon

The heart of Zend Job Queue is the Job Queue Daemon – this external service takes care of managing scheduling, queuing and executing tasks. It also communicates with the Zend Server Administration Interface to report any errors and failures.

Jobs are successfully queued once they are written to persistent storage. This makes sure that even if the daemon has to be restarted, or even in case of complete system failure, jobs are not lost.

The Job Queue Client Extension

Jobs, be they scheduled, recurring or one-time queued jobs are all passed to the Daemon by the Job Queue Client extension. This simple PHP extension provides an easy to use API for creating and managing jobs from within PHP. It communicates with the Job Queue Daemon through a TCP socket or a UNIX domain socket.

The Job Queue Client is also a part of the Zend Server Administration Interface – which, through the same extension, allows the end user to create and manage jobs from the convenience of a web GUI.

Executing Web Servers

As far as execution is concerned, running a job is no more than sending a simple HTTP request. When creating a new job through API or GUI, the user specifies the URL for this HTTP request, along with some possible optional parameters. When it's time to run the job, the Job Queue Daemon sends an HTTP request to the server specified by the URL along with the provided headers and parameters, and waits for a response.

The executing HTTP server can be any HTTP server – it might even be the same server that queued the job. This design makes it very easy to manage the system. You don't have to (but can, if needed) set up dedicated servers for handling off-line jobs. You don't have to (but can, if needed) create and manage a special PHP configuration for this environment. You can reuse existing code to a level where sometimes the difference between off-line and on-line execution is decided based on a simple request parameter.

Zend Code Tracing

Zend Code Tracing enables real-time execution flow recording in Production Environments.

Code Tracing enables deep analysis of PHP execution and flow using drill-down requests related to an Event (Monitoring Rule), or triggered manually.

The component focuses on collecting key data points such as:

- Application functions and main PHP function calls including function arguments
- High-resolution timing and memory usage of execution elements.
- Key PHP engine services such as Web server interface calls

The information collection process focuses on keeping relevant information based on either user-defined parameters that trigger *events* or a specific URL that is manually entered to be traced. For monitor events, each time an event is triggered the information is kept for further use. Otherwise, the information will be discarded to preserve disk-space. Events can also be triggered to generate a trace file for a specific URL (see Triggering a URL Trace).

Trace information is a capture of the Function Tree and enables deep tracing of the functions of the server parameters, including:

- Returns
- Memory
- Time (ms)

Workflow - The trace displayed in the Zend Server web console functions like a DVD player, showing the recorded execution history of the application. Users can follow the footsteps of a single problematic request in order to quickly pinpoint the root cause of the problem.

1. Manual Workflow - The code tracing of a single request can be generated manually through the code tracing page in Zend Server (Monitor | Code Tracing) (Figure 1). By executing a request, the full application execution is captured and stored in a trace file. The captured trace data can undergo collaborative reviewed leaving no need for developers and test engineers to pore over the symptoms of the defect and no room for misinterpretation of the events leading up to the error.
2. Event Monitoring Workflow - Code tracing in Zend Server has very low performance overhead. This therefore enables its use either while running a load test in the lab or while running in production.

By leveraging the Zend Server event monitoring mechanism, code tracing can save trace data only when a problem occurs. For example, when the performance of a checkout process in a web application drops below a predefined threshold, Zend Server can send an alert and capture the entire execution of the poorly performing request. The combination of knowing that a problem has occurred, while recording the entire execution flow, allows for quick identification and correction even before other users may notice.

Summary

One of the most appealing characteristics of this application is that the performance and memory characteristics have been designed to be suitable for production environments and excellent in-depth trace of execution.
Other advantages:

- In-depth root-cause analysis when event happens. Trace of “exact” request which had the failure (a view back in time)
- No need for risky reproduction
- Insight into additional PHP subsystems beyond the script itself such as Web Server interface
- Technology suitable for development, testing, staging and production

Zend Session Clustering

SC (Session Clustering), facilitates session management in cluster based environments. This module transparently shares session data between nodes. Session Clustering High Availability is an additional layer of functionality to for preventing information loss in case a server suddenly becomes unavailable.

Without SC, session information resides on different servers depending on where your Load Balancer routed incoming requests (at any given time) making continuity between requests difficult to follow. This is especially important in environments such as online stores that depend on continuity between request information, for example; to fill a shopping cart.

SC provides full continuity in a cluster. It doesn’t matter where the session resides because all servers share the information. This provides application transparency and an increase in application performance. SC also adds Linear Scalability providing a way to add new servers while running in order to expand your cluster such as cases where there are sudden traffic influxes. New servers added to the cluster start handling subsequent traffic immediately however, traffic that occurred before the server was added will not pass through the new server due to the fact that Load Balancers cannot recalculate distribution between nodes.

Using SC replaces the daemon that handles all incoming http requests with the SCD (Zend Session Manager) this new daemon creates a cookie if the request did not have a cookie or adds information to the cookie in the request. The information included in the cookie relates to the Cluster manager, the session key (the session's unique ID) and the backup server (only for HA). Subsequent requests will include this information (in the cookie) so the SCD will know where the session originated and which session and make sure that all new information coming from requests is managed from one master server.

Adding Extensions

This section includes information for the following Operating Systems:

- Zend Server on UNIX/Linux
- Zend Server Community Edition on UNIX/Linux/Mac
- Zend Server for IBM i

Zend Server 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 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 Restart PHP 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 .

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'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 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 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 repository called *php-5.2-source-zend-server* or *php-5.3-source-zend-server*, depending on your Zend Server'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.

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, 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 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 directory(in our example, the PHP version is 5.2.9 as it is the current stable version Zend Server 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/
```

3. Run the configure script, generated by phpize:
   $ ./configure --with-php-config=/usr/local/zend/bin/php-config
4. Run make:
   $ make
5. 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 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 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:

   Windows: *<install_path>\apache2\conf\httpd.conf*

   Linux/Tarbal: *<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.
- 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 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 '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.

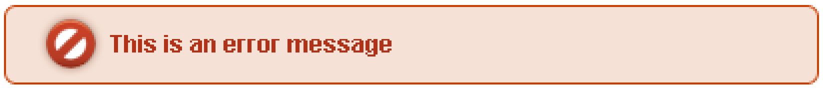
After script execution, the referenced class may be disposed.

Info Messages

Zend Server 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 message like this follow the instructions in the message.

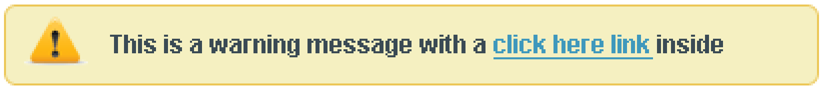

This is an error message

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 - Choosing Which Distribution to Install

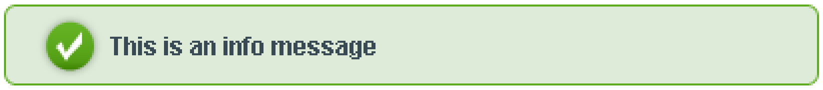
Notices

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.


This is a warning message with a click here link inside

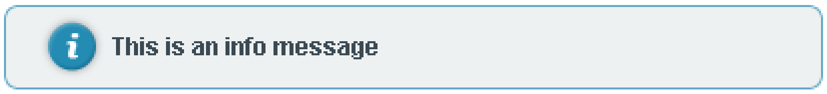
Success Messages

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).


This is an info message

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.


This is an info message

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

API Reference

Introduction

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 - Configuration Directives
- Zend Optimizer+ Directives
- Zend Optimizer+ - PHP API
- Zend Guard Loader - Configuration Directives
- Zend Guard Loader - PHP API
- Zend Data Cache - Configuration Directives
- Zend Data Cache - PHP API
- Zend Java Bridge - Configuration Directives
- Zend Java Bridge - PHP API
- The Java Exception Class
- Zend Download Server - Configuration Directives
- Zend Download Server - PHP API
- Zend Page Cache - Configuration Directives
- Zend Page Cache - PHP API
- Zend Monitor Node Daemon - Configuration Directives
- Zend Monitor - PHP API
- Zend Monitor UI extension - PHP API
- Zend Job Queue - Configuration Directives
- Zend Job Queue - PHP API
- The Zend Job Queue Class
- Zend Code Tracing - Configuration Directives
- Zend Session Clustering - Configuration Directives

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_ALL	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_ALL	Directive in order to mock up xdebug coverage
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:<br> 0 - Never. The presence of the Zend Debugger is not detected by other clients <br> 1 - Always. All clients can detect the Zend Debugger <br> 2 - Allowed Hosts. Only clients listed in zend_debugger.allow_hosts can detect the Zend Debugger<br> 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 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
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: <br> 1) Directories that contain auto generated code, like Smarty or ZFW cache.<br> 2) Code that does not work well when accelerated, due to some delayed compile time evaluation. <br> 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 elminating possible collisions between files with the same name (basename). Disablingthe directive improves performance, but may break existing applications.

Type: boolean

Default Value: 1

Available since version 4.0

zend_optimizerplus.validate_timestamps

When disabled, you must reset the Optimizer+ manually or restart the webserver for changes to the filesystem to take effect.<br> The frequancy 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.<br> 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).<br> For "debug" binaries, the default log verbosity level is 4, not 1.
Type: integer
Default Value: 1
Available since version 4.0

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 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+ features **(**where 0 is disabled and 1 is enabled).

Type: integer

Default Value: 0xfffffbbf

Available since version 4.0

The following is a list of each bit represented in the value :

- bit 0 - Enables/disables optimization step 1:
  - CSE - constants subexpressions elimination
  - Sequences of ADD_CHAR/ADD_STRING optimization
- bit 1 - Enables/disables optimization step 2:
  - Convert constant operands to expected types
  - Convert conditional jumps with constant operands
  - Optimize static BRK and CONT
- bit 2 - Enables/disables optimization step 3:
  - Convert $a = $a + expr into $a += expr
  - Convert $a++ into ++$a
  - Optimize series of JMPs
- bit 3 - Enables/disables optimization step 4:
  - PRINT and ECHO optimization
- bit 4 - Enables/disables optimization step 5:
  - block optimization (the most expensive optimization pass which perform many different optimization patterns based on CFG - control flow graph)
- bit 8- Enables/disables optimization step 9:
  - register allocation (allows re-usage of temporary variables)
- bit 9 - Enables/disables optimization step 10:
  - remove NOPs

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

Table of Contents

PHP Functions

accelerator_reset

Resets the contents of the Optimizer+ shared memory storage.<br> 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. <br> 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 <br>

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

Table of Contents

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.<br> 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 <br> 0=Do not overwrite<br> 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
zend_datacache.log_verbosity_level	integer	PHP_INI_SYSTEM	The log verbosity level [0-5]
zend_datacache.log_rotation_size	integer	PHP_INI_ALL	The maximum size of the log file before it is rotated

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_datacache.log_verbosity_level

The extension's log verbosity level. Level 1 includes very important information messages, errors and warnings. Level 2 displays notices. Greater levels (up to 5) are reserved for debug purposes

Type: integer

Default Value: 2

Available since version 4.0

zend_datacache.log_rotation_size

The maximum size of the log file before it is rotated

Type: integer

Units: MBytes

Default Value: 10

Available since version 4.0

Zend Data Cache - PHP API

Table of Contents

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, 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). <br> 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

Table of Contents

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: <br> https://, http://, file://, ftp:// <br> 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 Java Exception 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 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.
zend_dserver.use_sendfile	boolean	PHP_INI_SYSTEM	Enable use of sendfile() backend. It can significantly improve performence

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

Available since version

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: dserver.log

Available since version

zend_dserver.log_verbosity

The extension's log verbosity level. 1 - Fatal errors (ZDS goes down) <br> 2 - Errors (The currect request bails out<br> 3 - Warnings (not good, but continue)<br> 4 - Notice (important info)<br> 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

zend_dserver.use_sendfile

Enable use of sendfile() backend. It can significantly improve performence

Type: boolean

Default Value: On

Available since version

Zend Download Server - PHP API

Table of Contents

PHP Functions

zend_send_file

Outputs the contents of a file to a client using the ZDS and terminates the script.

Description

```
void zend_send_file (string $filename [ , string $mime_type [ , string
$custom_headers ]])
```

Parameters

filename
: The full path to the file for the download

mime_type
: MIME type of the file. The function probes the file, using the list in the mime_type file. If that fails, the file is treated as an application/octet-stream.

custom_headers
: User defined HTTP headers that are sent instead of the regular ZDS headers. If nothing is specified, a few essential headers are sent, anyway.

Return Value

zend_send_buffer

Ouputs the contents of a string buffer to a client using ZDS and terminates the script.

Description

```
void zend_send_buffer (string $buffer [ , string $mime_type [ , string
$custom_headers ]])
```

Parameters

buffer
The actual content for the output

mime_type
MIME type of the file. If nothing is specified, the file is treated as an application/octet-stream.

custom_headers
User defined HTTP headers that will be sent instead of regular ZDS headers. If nothing is specified, a few essential headers are sent, anyway.

Return Value

zds_get_pid

Returns the download server's process ID.

Description

```
int zds_get_pid (void)
```

Return Value

The download server's process ID.

Zend Page Cache - Configuration Directives

Configuration Directives Summary

Directive	Type	Modification Scope	Description
zend_pagecache.enable	boolean	PHP_INI_SYSTEM	Enables the Zend Page Cache extension
zend_pagecache.save_path	string	PHP_INI_SYSTEM	Location where the cache files are saved. This must point to an existing location.
zend_pagecache.dir_depth	integer	PHP_INI_SYSTEM	Depth of directory tree in which cached files are stored
zend_pagecache.log_verbosity_level	integer	PHP_INI_SYSTEM	The log verbosity level [0-5]
zend_pagecache.dependencies_file	string	PHP_INI_SYSTEM	The location of the configuration file in which caching rules are stored
zend_pagecache.compression_enable	boolean	PHP_INI_SYSTEM	Enables file compression of cached output
zend_pagecache.clean_frequency	integer	PHP_INI_SYSTEM	How often expired entries are removed from the cache. The cleaning frequency is configured in seconds
zend_pagecache.log_rotation_size	integer	PHP_INI_ALL	The maximum size of the log file before it is rotated

Configuration Directive Details

zend_pagecache.enable

Enables the Zend Page Cache extension

Type: boolean

Default Value: 1

Available since version 4.0

zend_pagecache.save_path

Location where the cache files are saved. This must point to an existing location.

Type: string

Default Value: pagecache

Available since version 4.0

zend_pagecache.dir_depth

Depth of directory tree in which cached files are stored

Type: integer

Default Value: 2

Available since version 4.0

zend_pagecache.log_verbosity_level

The extension's log verbosity level. Level 1 includes very important information messages, errors and warnings. Level 2 displays notices. Greater levels (up to 5) are reserved for debug purposes

Type: integer

Default Value: 2

Available since version 4.0

zend_pagecache.dependencies_file

The location of the configuration file in which caching rules are stored

Type: string

Default Value: pagecache_rules.xml

Available since version 4.0

zend_pagecache.compression_enable

Enables file compression of cached output. Applicable only on cached outputs larger than 1k.

Type: boolean

Default Value: 1

Available since version 4.0

zend_pagecache.clean_frequency

How often expired entries are removed from the cache. The cleaning frequency is configured in seconds

Type: integer

Units: seconds

Default Value: 300

Available since version 4.0

zend_pagecache.log_rotation_size

The maximum size of the log file before it is rotated

Type: integer

Units: MBytes

Default Value: 10

Available since version 4.0

Zend Page Cache - PHP API

Table of Contents

PHP Functions

page_cache_disable_caching

Disables output caching for the current request. This overrides any caching settings that are configured for the current request.
Available since version 4.0

Description

```
void page_cache_disable_caching (void)
```

page_cache_disable_compression

Does not allow the cache to perform compression on the output of the current request. This overrides any compression settings that are configured for this request.
Available since version 4.0

Description

```
void page_cache_disable_compression (void)
```

page_cache_remove_cached_contents

Clears cached contents for all requests that match a specific URL or regular expression

Available since version 4.0

Description

```
void page_cache_remove_cached_contents (string $URL)
```

Parameters

URL
: The URL or regular expression

page_cache_remove_all_cached_contents

Clears all cached contents

Available since version 4.0

Description

```
void page_cache_remove_all_cached_contents (void)
```

Zend Monitor - Configuration Directives

Configuration Directives Summary

Directive	Type	Modification Scope	Description
zend_monitor.enable	boolean	PHP_INI_ALL	Enables or disables the Monitor component
zend_monitor.log_verbosity	integer	PHP_INI_SYSTEM	Monitor extension log verbosity level
zend_monitor.log_rotation_size	integer	PHP_INI_ALL	The maximum size of the log file before it is rotated
zend_monitor.shm_size	integer	PHP_INI_SYSTEM	Amount of shared memory to allocate for event collection
zend_monitor.max_shm_segment_size	integer	PHP_INI_SYSTEM	The maximum size of a shared memory segment
zend_monitor.events_transport_parameter	string	PHP_INI_SYSTEM	Communication transport for event reporting to the Monitor Node
zend_monitor.report_super_globals	string	PHP_INI_ALL	PHP Super-global variables to include in event reports
zend_monitor.security_black_list	string	PHP_INI_SYSTEM	PHP Super-global values that should not be included in Zend

			Monitor events
zend_monitor.super_globals_to_secure	string	PHP_INI_ALL	PHP Super-globals to pass through the security black-list filter
zend_monitor.max_super_globals_string_len	integer	PHP_INI_ALL	The maximum length of a superglobal to include in an event report
zend_monitor.event.request_slow_exec.disabled_on_function_slow_exec_event	boolean	PHP_INI_ALL	Disable reporting of Slow Request Execution events if a Slow Function Execution event was reported
zend_monitor.event.request_slow_exec.disabled_on_high_load.threshold	integer	PHP_INI_ALL	Sets the load level to disable Slow Request Execution events for the same request that already triggered a high load event.
zend_monitor.event.request_relative_slow_exec.min_exec_time	integer	PHP_INI_ALL	The minimum request execution time for a relativity check.
zend_monitor.event.request_relative_large_mem_usage.min_mem_usage	integer	PHP_INI_ALL	The minimum request memory usage for a relativity check
zend_monitor.event.request_relative_large_output_size.min_output_size	integer	PHP_INI_ALL	The minimum

			request output size for a relativity check
zend_monitor.event.zend_error.silence_level	integer	PHP_INI_ALL	Controls PHP Error event reporting when PHP error reporting is supressed
zend_monitor.requests_statistics.warmup_requests	integer	PHP_INI_ALL	How many requests to run before deviation events are calculated (to collect data for the average). This is used to calculate averages for relative events.
zend_monitor.requests_statistics.request_lifetime	integer	PHP_INI_ALL	How long to wait (in seconds) before statistics are reset if a request is not called
zend_monitor.events_rules_xml_file_name	string	PHP_INI_SYSTEM	Event rules XML configuration file.
zend_monitor.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)
zend_monitor.event_tracing_mode	integer	PHP_INI_ALL	Determines integration with tracer mechanism in case of events that require a trace data dump
zend_monitor.aggregate_by_url_query	boolean	PHP_INI_ALL	When enabled, use the URL's query string as part of the aggregation key

Configuration Directive Details

zend_monitor.enable

Enables or disables the Monitor component. When set to On, Zend Monitor will colelct and report PHP events. When set to Off, PHP will not be monited

Type: boolean

Default Value: 1

Available since version 4.0

zend_monitor.log_verbosity

The extension's log verbosity level.

- Level 1 includes very important information messages, errors and warnings.
- Level 2 displays notices.
- Greater levels (up to 5) are reserved for debugging purposes

Type: integer

Default Value: 2

Available since version 4.0

zend_monitor.log_rotation_size

The maximum size of the log file before it is rotated

Type: integer

Units: MBytes

Default Value: 10

Available since version 4.0

zend_monitor.shm_size

Amount of shared memory to allocate for event collection. If exceeded, an error message is reported to the log

Type: integer

Default Values:

- Windows, Linux i386, Linux x86-64, Linux AMD64: 4194304
- i5/OS: 2097152

Available since version 4.0

zend_monitor.max_shm_segment_size

The maximum size of a shared memory segment

Type: integer

Default Values:

- Windows, Linux i386, Linux x86-64, Linux AMD64: 4194304
- i5/OS: 2097152

Available since version 4.0

zend_monitor.events_transport_parameter

Defines the network communication transport to be used when sending event reports from the Monitor Extension to the Monitor Node

Type: string

Default Values:

- Unix-like Systems: events.sock
- Windows: events

Available since version 4.0

zend_monitor.report_super_globals

Which PHP Super-global variables should be included event reports. You can remove Super-globals from this, if they do not contain helpful debugging information, in order to reduce event reporting overhead.
You may also add additional Super-globals if the data they contain is relevant for debugging.
Each Super-global is represented by a different character from the following list:

- G - GET (included by default)
- P - POST (included by default)
- C - COOKIE (included by default)
- R - RAW_POST_DATA (included by default)
- V - SERVER (included by default)
- F - FILES (included by default)
- S - SESSION
- E - ENV

Type: string
Default Value: PRGCVF
Available since version 3.6

zend_monitor.security_black_list

Defines a list of PHP Super-global values that will not be collected for Zend Monitor events.
Each Super-globa (from the variables defined in zend_monitor.super_globals_to_secure) value who's key is listed here, will not be saved in event reports, and the string "<BLOCKED_VALUE>" will appear instead of the actual value.
The list of keys to remove can be defined in two ways:

1. A comma-separated list of array keys to remove
2. A path to a file, prefixed by '@'. This file should include a newline-separated list of array keys to remove.

Type: string
Available since version 3.6

zend_monitor.super_globals_to_secure

Defines which PHP Super-global variables should passed through the security black-list filter prior to being included in event reports. See zend_monitor.security_black_list for details
Each Super-global is represented by a different character from the following list:

- G - GET (included by default)
- P - POST (included by default)
- C - COOKIE (included by default)
- V - SERVER (included by default)
- F - FILES
- R - RAW_POST_DATA
- S - SESSION
- E - ENV

Type: string
Default Value: PGCV
Available since version 3.6

zend_monitor.max_super_globals_string_len

When the string is passed, any characters that exceed the limit are truncated and '...' is appended to the end of the string the end, to indicate that this is a partial value.
Type: integer
Default Value: 100
Available since version 3.6

zend_monitor.event.request_slow_exec.disabled_on_function_slow_exec_event

Disable reporting of Slow Request Execution events if a Slow Function Execution event was reported in the same request. This reduces the chance of getting double slow-execution reports triggered by the same underlying cause.
Type: boolean
Default Value: 0
Available since version 4.0

zend_monitor.event.request_slow_exec.disabled_on_high_load.threshold

Sets the load level to disable Slow Request Execution events for the same request that already triggered a high load event.

Type: integer

Default Value: 0

Available since version 4.0

zend_monitor.event.request_relative_slow_exec.min_exec_time

The minimum execution time for a request to be compared to the average request execution time.

Type: integer

Units: milliseconds

Default Value: 100

Available since version 4.0

zend_monitor.event.request_relative_large_mem_usage.min_mem_usage

The minimum memory usage of a request to be compared to the average request memory usage.

Type: integer

Units: KBytes

Default Value: 100

Available since version 4.0

zend_monitor.event.request_relative_large_output_size.min_output_size

The minimum output size of a request to be compared to the average request output size.

Type: integer

Units: KBytes

Default Value: 100

Available since version 4.0

zend_monitor.event.zend_error.silence_level

Controls PHP Error event reporting when the silencing operator (@) is used or when the error_reporting level is set to 0. The values are:

- 0 - PHP warnings and errors are reported
- 1 - PHP warnings and errors are not reported
- 2 - PHP warnings and errors are reported even when error_reporting is set to 0, but not when the silencing operator (@) is explicitly used

Type: integer
Default Value: 1
Available since version 4.0

zend_monitor.requests_statistics.warmup_requests

How many requests to run before deviation events are calculated (to collect data for the average). This is used to calculate averages for relative events.
Type: integer
Default Value: 500
Available since version 4.0

zend_monitor.requests_statistics.request_lifetime

How long (in seconds) to hold statistics in memory. If a request is not called within that time period, the statistics are reset
Type: integer
Default Value: 300
Available since version 4.0

zend_monitor.events_rules_xml_file_name

The name and path to the XML file that contains the defininitons for event rules and actions. This file should not be edited manually
Type: string
Default Value: events_rules.xml
Available since version 4.0

zend_monitor.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_monitor.event_tracing_mode

Determines integration with tracer mechanism in case of events that require a trace data dump. 0 means off, 1 means on, 2 means standby.

Type: integer

Default Value: 1

Available since version 4.0

zend_monitor.aggregate_by_url_query

When enabled, use the URL's query string as part of the aggregation key

Type: boolean

Default Value: 1

Available since version 4.0

Zend Monitor Node Daemon - Configuration Directives

Configuration Directives Summary

Directive	Type	Description
zend_monitor.global_directives_ini_file	string	Global Directives ini File
zend_monitor.node_id	string	Node Id
zend_monitor.log_verbosity	integer	The Log's verbosity level
zend_monitor.log_rotation_size	integer	The maximum size of the log file before it is rotated
zend_monitor.database.type	string	Database Type (SQLITE, MYSQL, DB2)
zend_monitor.database.name	string	Name of the Database
zend_monitor.database.host_name	string	Hos-Name of the Database machine (server db only)
zend_monitor.database.port	integer	Port of the Databse (server db only)
zend_monitor.database.user	string	User-name to connect to the database (server db only)
zend_monitor.database.password	string	Password to connect to the database (server db only)
zend_monitor.events_rules_xml_file_name	string	Events Rules XML File Name. For the full path, concatenate it with zend.conf_dir value
zend_monitor.smtp_server	string	SMTP server
zend_monitor.smtp_port	integer	SMTP Port
zend_monitor.sendmail_from	string	Send Mail From email Address
zend_monitor.gui_host_name	string	Host name (and port) of the Zend Server GUI
zend_monitor.max_events_queue_size	integer	Maximum size of the Events Queue (in MB)
zend_monitor.queue_flush_interval	integer	Number of event reports to cache before flusing them into the database
zend_monitor.gui_default_emails	string	A semicolon separated list of email addresses which serves as the default list for events
zend_monitor.persistent_smtp	boolean	Use a persistent connection to the SMTP server

Configuration Directive Details

zend_monitor.global_directives_ini_file

The .ini file that contains the global directives, as defined in ZendGlobalDirectiveDD.xml

Type: string

Default Value: GLOBAL_DIRECTIVES_INI_FILE

Available since version 4.0

zend_monitor.node_id

The Node Id

Type: string

Default Value: 1

Available since version 4.0

zend_monitor.log_verbosity

The Log's verbosity level

Type: integer

Default Value: 2

Available since version 4.0

zend_monitor.log_rotation_size

The maximum size of the log file before it is rotated

Type: integer

Units: MBytes

Default Value: 10

Available since version 4.0

zend_monitor.database.type

The type of the database to be used. Choose from the following values: SQLIE, MYSQL, DB2

Type: string

Default Value: SQLITE

Available since version 4.0

zend_monitor.database.name

The name of the Database. SQLITE: file-name, MYSQL: schema-name

Type: string

Default Value: monitor

Available since version 4.0

zend_monitor.database.host_name

The host-name of the Database machine. SQLITE: N/A, MYSQL: machine-name/ip

Type: string

Available since version 4.0

zend_monitor.database.port

The port, in-which teh Database is listening on. SQLITE: N/A, MYSQL: port

Type: integer

Default Value: 3306

Available since version 4.0

zend_monitor.database.user

The user-name to connect to the DB. SQLITE: N/A, MYSQL: machine-name/ip

Type: string

Available since version 4.0

zend_monitor.database.password

The password to connect to the DB. SQLITE: N/A, MYSQL: machine-name/ip

Type: string

Available since version 4.0

zend_monitor.events_rules_xml_file_name

The name of the file that contains the XML for the Events Rules and Actions

Type: string

Default Value: events_rules.xml

Available since version 4.0

zend_monitor.smtp_server

The SMTP server to use to send event mails

Type: string

Available since version 4.0

zend_monitor.smtp_port

The SMTP Port that the SMTP server uses to send event mails

Type: integer

Default Value: 25

Available since version 4.0

zend_monitor.sendmail_from

An email address to be used as the 'from mail' for Event mails

Type: string

Available since version 4.0

zend_monitor.gui_host_name

This host name is used to build a link to an issue when the user is notified of the issue

Type: string

Available since version 4.0

zend_monitor.max_events_queue_size

The maximum size of the Received Events queue (MB). If events are received beyond the maximum size, the incoming events are dropped out

Type: integer

Units: MBytes

Default Value: 5

Available since version 4.0

zend_monitor.queue_flush_interval

Number of event reports to cache before flusing them into the database

Type: integer

Default Value: 1

Available since version 5.0

zend_monitor.gui_default_emails

A semicolon separated list of email addresses which serves as the default list for events

Type: string

Available since version 4.0

zend_monitor.persistent_smtp

Use a persistent connection to the SMTP server

Type: boolean

Default Value: 0

Available since version 4.0

Zend Monitor - PHP API

Table of Contents

Predefined Global Constants

ZEND_MONITOR_ETBM_ALL
Bitmask Representing all Event Types(for zend_monitor_event_reporting)

ZEND_MONITOR_ETBM_NONE
Bitmask Representing NO Monitoring Events (for zend_monitor_event_reporting)

ZEND_MONITOR_ETBM_CUSTOM
Custom Event-Type Bitmask(for zend_monitor_event_reporting)

ZEND_MONITOR_ETBM_FUNC_SLOW_EXEC
Function Slow Execution Event-Type Bitmask(for zend_monitor_event_reporting)

ZEND_MONITOR_ETBM_FUNC_ERR
Function Error Event-Type Bitmask(for zend_monitor_event_reporting)

ZEND_MONITOR_ETBM_REQ_SLOW_EXEC
Request Slow Execution Event-Type Bitmask(for zend_monitor_event_reporting)

ZEND_MONITOR_ETBM_REQ_REL_SLOW_EXEC
Request Relative Slow Execution Event-Type Bitmask(for zend_monitor_event_reporting)

ZEND_MONITOR_ETBM_REQ_LARGE_MEM_USAGE
Request Large Memory Usage Event-Type Bitmask(for zend_monitor_event_reporting)

ZEND_MONITOR_ETBM_REQ_REL_LARGE_MEM_USAGE
Request Relative Large Memory Usage Event-Type Bitmask(for zend_monitor_event_reporting)

ZEND_MONITOR_ETBM_REQ_LARGE_OUT_SIZE
Request Relateive Large Output Size Event-Type Bitmask(for zend_monitor_event_reporting)

ZEND_MONITOR_ETBM_ZEND_ERR
Zend/PHP Error Event-Type Bitmask(for zend_monitor_event_reporting)

ZEND_MONITOR_ETBM_JAVA_EXP
Unhandled Java Exception Event-Type Bitmask(for zend_monitor_event_reporting)

ZEND_MONITOR_ETBM_JQ_JOB_EXEC_ERROR
Job Queue - Job Execution Error Event-Type Bitmask

ZEND_MONITOR_ETBM_JQ_JOB_LOGICAL_FAILURE
Job Queue - Job Logical Error Event-Type Bitmask(for zend_monitor_event_reporting)

ZEND_MONITOR_ETBM_JQ_JOB_EXEC_DELAY
Job Queue - Job Execution Delayed Event-Type Bitmask(for zend_monitor_event_reporting)

ZEND_MONITOR_ETBM_JQ_DAEMON_HIGH_CONCURRENCY_LEVEL
Job Queue - Daemon High Concurrency Event-Type Bitmask(for zend_monitor_event_reporting)

ZEND_MONITOR_ETBM_ET_TRACER_FILE_WRITE_FAIL
Tracer - Failed to write a dump file Event-Type Bitmask(for zend_monitor_event_reporting)

ZEND_MONITOR_ET_JQ_JOB_EXEC_ERROR
Job Queue - Job Execution Error Event-Type Bitmask

ZEND_MONITOR_ET_JQ_JOB_LOGICAL_FAILURE
Job Queue - Job Logical Error Event-Type

ZEND_MONITOR_ET_JQ_JOB_EXEC_DELAY
Job Queue - Job Execution Delayed Event-Type

ZEND_MONITOR_ET_JQ_DAEMON_HIGH_CONCURRENCY_LEVEL
Job Queue - Daemon High Concurrency Event-Type

ZEND_MONITOR_ET_TRACER_FILE_WRITE_FAIL

Tracer - Failed to write a dump file Event-Type

ZEND_MONITOR_ET_ZSM_CONFIGUATION_MISMATCH
ZSM - Configuration is not matching the cluster

ZEND_MONITOR_ET_ZSM_NODE_ADDED_SUCCESSFULLY
ZSM - Node added successfully to the cluster

ZEND_MONITOR_ET_ZSM_NODE_IS_NOT_RESPONDING
ZSM - Node is not responding

ZEND_MONITOR_ET_ZSM_RESTART_FAILED
ZSM - Node removed successfully from the cluster

ZEND_MONITOR_ET_TRACER_FILE_WRITE_FAIL
ZSM - Restart failed

PHP Functions

zend_monitor_pass_error

Passes an error to the Monitor component with file and line details. This function is used in error handlers. An alternative is to use trigger_error. However, this function does not indicate the file name and line number: It only passes the error message.
Available since version 4.0

Description

```
void zend_monitor_pass_error (int $errno, string $errstr, string
$errfile, int $errline)
```

Parameters

errno
: Error code

errstr
: Error string

errfile
: Error file

errline
: Error Line

zend_monitor_custom_event

Creates a special (custom) event that is generated from your code. The information collected consists of the three following parameters: Class, Text and User Data.
Available since version 4.0

Description

```
void zend_monitor_custom_event (string $class, string $text, mixed
$user_data)
```

Parameters

class
: event type

text
: string to appear in the event

user_data
: optional. Any additional data to store with the event

zend_monitor_set_aggregation_hint

Incorporates the locations of occurrences in the script when there are events that require those location for diagnosing the reason an event occured. Only events of the same type are aggregated. The collected information is viewed in the Zend Server Administration Interface.
Available since version 4.0

Description

```
void zend_monitor_set_aggregation_hint (string $hint)
```

Parameters

hint
    aggregation hint string

Return Value

zend_monitor_event_reporting

Enables or disables the event reporting of some event types by passing a bit-mask (as done by PHP's error_reporting function), but with the constants listed above, in ZEND_MONITOR_ET*.

Note: You cannot enable events that are disabled in the Event Rules file

Available since version 4.0

Description

```
int zend_monitor_event_reporting ([ int $new_error_reporting = null ])
```

Parameters

new_error_reporting
    The new error reporting level to use. The default value is null

Return Value

The previous error_reporting level or FALSE if there is an error

Zend Monitor UI extension - PHP API

Table of Contents

PHP Functions

zend_monitor_delete_old_issues

Delete old issues from the Zend Monitor database and daemon cache

Available since version 5.0

Description

```
array zend_monitor_delete_old_issues (int $expiration_days, int
$expiration_days_closed)
```

Parameters

expiration_days
    All issues that did not reoccur for more than expiration_days will be deleted

expiration_days_closed
    All "Closed" or "Ignored" issues that did not reoccur for more than expiration_days_closed will be deleted

Return Value

List of trace files IDs related to deleted issues

Zend Job Queue - Configuration Directives

Configuration Directives Summary

Directive	Type	Modification Scope	Description
zend_jobqueue.enable	boolean	PHP_INI_SYSTEM	Enables the Job Queue
zend_jobqueue.daemon_connection_timeout	integer	PHP_INI_SYSTEM	The default connection timeout to the Job Queue Daemon
zend_jobqueue.default_binding	string	PHP_INI_SYSTEM	Default binding to the Job Queue Daemon
zend_jobqueue.log_verbosity_level	integer	PHP_INI_SYSTEM	The log verbosity level [0-5]
zend_jobqueue.log_rotation_size	integer	PHP_INI_ALL	The maximum size of the log file before it is rotated
zend_jobqueue.named_queues	string	PHP_INI_ALL	This directive provides the ability to use aliases instead of bindings for Job Queue names.

Configuration Directive Details

zend_jobqueue.enable

Enables the Job Queue

Type: boolean

Default Value: 1

Available since version 5.0

zend_jobqueue.daemon_connection_timeout

The default connection timeout to the Job Queue Daemon. It can be overridden using a parameter to ZendJobQueue constructor

Type: integer

Units: seconds

Default Value: 10

Available since version 5.0

zend_jobqueue.default_binding

Default binding to the Job Queue Daemon. It can be overridden using a parameter to ZendJobQueue constructor
Type: string
Default Values:

- unix://jobqueue.sock
- Windows: tcp://127.0.0.1:10085

Available since version 5.0

zend_jobqueue.log_verbosity_level

The extension's log verbosity level. Level 1 includes very important info messages, errors and warnings. Level 2 displays notices. Greater levels (up to 5) serve debug purposes only.
Type: integer
Default Value: 2
Available since version 5.0

zend_jobqueue.log_rotation_size

The maximum size of the log file before it is rotated
Type: integer
Units: MBytes
Default Value: 10
Available since version 5.0

zend_jobqueue.named_queues

The named queue format is a list of name => binding pairs, for example <name>:<binding>;<name>:<binding>;... Names are unique and cannot be defined twice with a different value although the same binding with different aliases is accepted.
Type: string
Available since version 5.0

Zend Job Queue - PHP API

Table of Contents

- ZendJobQueue::getApplications - Returns an array of application names known by the daemon
- ZendJobQueue::getSchedulingRules - Returns an array of all the registered scheduled rules. Each rule is represented by a nested associative array with the following properties: "id" - The scheduling rule identifier "status" - The rule status (see STATUS_* constants) "type" - The rule type (see TYPE_* constants) "priority" - The priority of the jobs created by this rule "persistent" - The persistence flag of the jobs created by this rule "script" - The URL or script to run "name" - The name of the jobs created by this rule "vars" - The input variables or arguments "http_headers" - The additional HTTP headers "schedule" - The CRON-like schedule command "app_id" - The application name associated with this rule and created jobs "last_run" - The last time the rule was run "next_run" - The next time the rule will run
- ZendJobQueue::getSchedulingRule - Returns an associative array with the properties of the scheduling rule identified by the given argument. The list of the properties is the same as in getSchedulingRule()
- ZendJobQueue::deleteSchedulingRule - Deletes the scheduling rule identified by the given $rule_id and scheduled jobs created by this rule
- ZendJobQueue::suspendSchedulingRule - Suspends the scheduling rule identified by given $rule_id and deletes scheduled jobs created by this rule
- ZendJobQueue::resumeSchedulingRule - Resumes the scheduling rule identified by given $rule_id and creates a corresponding scheduled job
- ZendJobQueue::updateSchedulingRule - Updates and reschedules the existing scheduling rule
- ZendJobQueue::getCurrentJobParams - Decodes an array of input variables passed to the HTTP job
- ZendJobQueue::setCurrentJobStatus - Reports job completion status (OK or FAILED) back to the daemon

The ZendJobQueue Class

The ZendJobQueue is a PHP class that implements a connection to the Job Queue Daemon

Available since: 5.0

Class Prototype

```
class ZendJobQueue {
/* Constants */
const int TYPE_HTTP;
const int TYPE_HTTP_RELATIVE;
const int TYPE_SHELL;
const int PRIORITY_LOW;
const int PRIORITY_NORMAL;
const int PRIORITY_HIGH;
const int PRIORITY_URGENT;
const int STATUS_PENDING;
const int STATUS_WAITING_PREDECESSOR;
const int STATUS_RUNNING;
const int STATUS_COMPLETED;
const int STATUS_FAILED;
const int STATUS_OK;
const int STATUS_LOGICALLY_FAILED;
const int STATUS_TIMEOUT;
const int STATUS_REMOVED;
const int STATUS_SCHEDULED;
const int STATUS_SUSPENDED;
const int SORT_NONE;
const int SORT_BY_ID;
const int SORT_BY_TYPE;
const int SORT_BY_SCRIPT;
const int SORT_BY_APPLICATION;
const int SORT_BY_NAME;
const int SORT_BY_PRIORITY;
const int SORT_BY_STATUS;
const int SORT_BY_PREDECESSOR;
const int SORT_BY_PERSISTENCE;
const int SORT_BY_CREATION_TIME;
const int SORT_BY_SCHEDULE_TIME;
const int SORT_BY_START_TIME;
const int SORT_BY_END_TIME;
const int SORT_ASC;
const int SORT_DESC;
const int OK;
const int FAILED;
/* Methods */
void __construct ([ string $queue ])
int createHttpJob (string $url, array $vars, mixed $options)
array getJobStatus (int $job_id)
boolean removeJob (int $job_id)
boolean restartJob (int $job_id)
boolean isSuspended (void)
static boolean isJobQueueDaemonRunning (void)
void suspendQueue (void)
void resumeQueue (void)
array getStatistics (void)
array getConfig (void)
boolean reloadConfig (void)
array getJobInfo (int $job_id)
array getDependentJobs (int $job_id)
array getJobsList (array $query, int $total)
array getApplications (void)
array getSchedulingRules (void)
array getSchedulingRule (int $rule_id)
boolean deleteSchedulingRule (int $rule_id)
boolean suspendSchedulingRule (int $rule_id)
boolean resumeSchedulingRule (int $rule_id)
boolean updateSchedulingRule (int $rule_id, string $script, array $vars, array $options)
static array getCurrentJobParams (void)
static void setCurrentJobStatus (int $completion, string $msg)
}
```

Class Constants

ZendJobQueue::TYPE_HTTP
A HTTP type of job with an absolute URL

ZendJobQueue::TYPE_HTTP_RELATIVE
A HTTP type of job with a relative URL

ZendJobQueue::TYPE_SHELL
A SHELL type of job

ZendJobQueue::PRIORITY_LOW
A low priority job

ZendJobQueue::PRIORITY_NORMAL
A normal priority job

ZendJobQueue::PRIORITY_HIGH
A high priority job

ZendJobQueue::PRIORITY_URGENT
An urgent priority job

ZendJobQueue::STATUS_PENDING
The job is waiting to be processed

ZendJobQueue::STATUS_WAITING_PREDECESSOR
The job is waiting for its predecessor's completion

ZendJobQueue::STATUS_RUNNING
The job is executing

ZendJobQueue::STATUS_COMPLETED
Job execution has been completed successfully

ZendJobQueue::STATUS_FAILED
The job execution failed

ZendJobQueue::STATUS_OK
The job was executed and reported its successful completion status

ZendJobQueue::STATUS_LOGICALLY_FAILED
The job was executed but reported failed completion status

ZendJobQueue::STATUS_TIMEOUT
Job execution timeout

ZendJobQueue::STATUS_REMOVED
A logically removed job

ZendJobQueue::STATUS_SCHEDULED
The job is scheduled to be executed at some specific time

ZendJobQueue::STATUS_SUSPENDED
The job execution is susspended

ZendJobQueue::SORT_NONE
Disable sorting of result set of getJobsList()

ZendJobQueue::SORT_BY_ID
Sort result set of getJobsList() by job id

ZendJobQueue::SORT_BY_TYPE
Sort result set of getJobsList() by job type

ZendJobQueue::SORT_BY_SCRIPT
Sort result set of getJobsList() by job script name

ZendJobQueue::SORT_BY_APPLICATION
Sort result set of getJobsList() by application name

ZendJobQueue::SORT_BY_NAME
Sort result set of getJobsList() by job name

ZendJobQueue::SORT_BY_PRIORITY
Sort result set of getJobsList() by job priority

ZendJobQueue::SORT_BY_STATUS
Sort result set of getJobsList() by job status

ZendJobQueue::SORT_BY_PREDECESSOR
Sort result set of getJobsList() by job predecessor

ZendJobQueue::SORT_BY_PERSISTENCE
Sort result set of getJobsList() by job persistence flag

ZendJobQueue::SORT_BY_CREATION_TIME
Sort result set of getJobsList() by job creation time

ZendJobQueue::SORT_BY_SCHEDULE_TIME
Sort result set of getJobsList() by job schedule time

ZendJobQueue::SORT_BY_START_TIME
Sort result set of getJobsList() by job start time

ZendJobQueue::SORT_BY_END_TIME
Sort result set of getJobsList() by job end time

ZendJobQueue::SORT_ASC
Sort result set of getJobsList() in direct order

ZendJobQueue::SORT_DESC
Sort result set of getJobsList() in reverse order

ZendJobQueue::OK
Constant to report completion status from the jobs using setCurrentJobStatus()

ZendJobQueue::FAILED
Constant to report completion status from the jobs using setCurrentJobStatus()

Class Methods

ZendJobQueue::__construct

Creates a ZendJobQueue object connected to a Job Queue daemon.

Available since version 5.0

Description

```
void ZendJobQueue::__construct ([ string $queue ])
```

Parameters

queue
: This can be one of: 1. No value specified - the default binding will be used. 2. A named queue as defined in the named queues directive - In such a case, the client will connect to the binding specified by the directive, and the application name used will be the value provided. 3. A literal binding URL - the URL will be used to connect to the daemon directly, and no application name will be defined. 4. If a string is provided which does not match a binding URL format, and has no alias defined for it, an exception will be thrown. . The default value is taken from default_binding directive

ZendJobQueue::createHttpJob

Creates a new URL based job to make the Job Queue Daemon call given $script with given $vars

Available since version 5.0

Description

```
int ZendJobQueue::createHttpJob (string $url, array $vars, mixed
$options)
```

Parameters

url
: An absolute URL of the script to call

vars
: An associative array of variables which will be passed to the script. The total data size of this array should not be greater than the size defined in the zend_jobqueue.max_message_size directive.

options
: An associative array of additional options. The elements of this array can define job priority, predecessor, persistence, optional name, additional attributes of HTTP request as HTTP headers, etc The following options are supported: "name" - Optional job name "priority" - Job priority (see corresponding constants) "predecessor" - Integer predecessor job id "persistent" - Boolean (keep in history forever) "schedule_time" - Time when job should be executed "schedule" - CRON-like scheduling command "http_headers" - Array of additional HTTP headers

Return Value

A job identifier which can be used to retrieve the job status

ZendJobQueue::getJobStatus

Retrieves status of previously created job identified by $job_id

Available since version 5.0

Description

```
array ZendJobQueue::getJobStatus (int $job_id)
```

Parameters

job_id
: a job identifier

Return Value

The array contains status, completion status and output of the job

ZendJobQueue::removeJob

Removes the job from the queue. Makes all dependent jobs fail. In case the job is in progress it will be finished but dependent jobs won't be started anyway. For non-existing jobs the function just returns false. Finished jobs are simply removed from the database

Available since version 5.0

Description

```
boolean ZendJobQueue::removeJob (int $job_id)
```

Parameters

job_id
: A job identifier

Return Value

The job was removed or not removed

ZendJobQueue::restartJob

Restart a previously executed Job and all its followers.

Available since version 5.0

Description

```
boolean ZendJobQueue::restartJob (int $job_id)
```

Parameters

job_id
: A job identifier

Return Value

If the job was restarted or not restarted

ZendJobQueue::isSuspended

Checks if Queue is suspended and returns true or false

Available since version 5.0

Description

```
boolean ZendJobQueue::isSuspended (void)
```

Return Value

A Job Queue status

ZendJobQueue::isJobQueueDaemonRunning

Checks if the Job Queue Daemon is running

Available since version 5.0

Description

```
static boolean ZendJobQueue::isJobQueueDaemonRunning (void)
```

Return Value

Return true if the Job Queue Deamon is running, otherwise it returns false

ZendJobQueue::suspendQueue

Suspends the Job Queue so it will accept new jobs, but won't start them. The jobs which were executed during call to this function will be completed
Available since version 5.0

Description

```
void ZendJobQueue::suspendQueue (void)
```

ZendJobQueue::resumeQueue

Resumes the Job Queue so it will schedule and start queued jobs.
Available since version 5.0

Description

```
void ZendJobQueue::resumeQueue (void)
```

ZendJobQueue::getStatistics

Returns internal daemon statistics such as up-time, number of complete jobs, number of failed jobs, number of waiting jobs, number of currently running jobs, etc
Available since version 5.0

Description

```
array ZendJobQueue::getStatistics (void)
```

Return Value

Associative array

ZendJobQueue::getConfig

Returns the current value of the configuration option of the Job Queue Daemon
Available since version 5.0

Description

```
array ZendJobQueue::getConfig (void)
```

Return Value

Associative array of configuration variables

ZendJobQueue::reloadConfig

Re-reads the configuration file of the Job Queue Daemon and reloads all directives that are reloadable

Available since version 5.0

Description

```
boolean ZendJobQueue::reloadConfig (void)
```

Return Value

If configuration file was loaded successfully or not

ZendJobQueue::getJobInfo

Returns an associative array with properties of the job with the given id from the daemon database

Available since version 5.0

Description

```
array ZendJobQueue::getJobInfo (int $job_id)
```

Parameters

job_id
  a job identifier

Return Value

array of job details. The following properties are provided (some of them don't have to always be set): "id" - The job identifier "type" - The job type (see TYPE_* constants) "status" - The job status (see STATUS_* constants) "priority" - The job priority (see PRIORITY_* constants) "persistent" - The persistence flag "script" - The URL or SHELL script name "predecessor" - The job predecessor "name" - The job name "vars" - The input variables or arguments "http_headers" - The additional HTTP headers for HTTP jobs "output" - The output of the job "error" - The error output of the job "creation_time" - The time when the job was created "start_time" - The time when the job was started "end_time" - The time when the job was finished "schedule" - The CRON-like schedule command "schedule_time" - The time when the job execution was scheduled "app_id" - The application name

ZendJobQueue::getDependentJobs

Returns a list of associative arrays with the properties of the jobs which depend on the job with the given identifier

Available since version 5.0

Description

```
array ZendJobQueue::getDependentJobs (int $job_id)
```

Parameters

job_id
: A job identifier

Return Value

A list of jobs

ZendJobQueue::getJobsList

Returns a list of associative arrays with properties of jobs which conform to a given query

Available since version 5.0

Description

```
array ZendJobQueue::getJobsList (array $query, int $total)
```

Parameters

query
: An associative array with query arguments The array may contain the following keys which restrict the resulting list: "app_id" - Query only jobs which belong to the given application "name" - Query only jobs with the given name "script" - Query only jobs with a script name similar to the given one (SQL LIKE) "type" - Query only jobs of the given types (bitset) "priority" - Query only jobs with the given priorities (bitset) "status" - Query only jobs with the given statuses (bitset) "rule_id" - Query only jobs produced by the given scheduling rule "scheduled_before" - Query only jobs scheduled before the given date "scheduled_after" - Query only jobs scheduled after the given date "executed_before" - Query only jobs executed before the given date "executed_after" - Query only jobs executed after the given date "sort_by" - Sort by the given field (see SORT_BY_* constants) "sort_direction" - Sort the order (SORT_ASC or SORT_DESC) "start" - Skip the given number of jobs "count" - Retrieve only the given number of jobs (100 by default)

total
: The output parameter which is set to the total number of jobs conforming to the given query, ignoring "start" and "count" fields

Return Value

A list of jobs with their details

ZendJobQueue::getApplications

Returns an array of application names known by the daemon
Available since version 5.0

Description

```
array ZendJobQueue::getApplications (void)
```

Return Value

A list of applications

ZendJobQueue::getSchedulingRules

Returns an array of all the registered scheduled rules. Each rule is represented by a nested associative array with the following properties: "id" - The scheduling rule identifier "status" - The rule status (see STATUS_* constants) "type" - The rule type (see TYPE_* constants) "priority" - The priority of the jobs created by this rule "persistent" - The persistence flag of the jobs created by this rule "script" - The URL or script to run "name" - The name of the jobs created by this rule "vars" - The input variables or arguments "http_headers" - The additional HTTP headers "schedule" - The CRON-like schedule command "app_id" - The application name associated with this rule and created jobs "last_run" - The last time the rule was run "next_run" - The next time the rule will run
Available since version 5.0

Description

```
array ZendJobQueue::getSchedulingRules (void)
```

Return Value

A list of scheduling rules

ZendJobQueue::getSchedulingRule

Returns an associative array with the properties of the scheduling rule identified by the given argument. The list of the properties is the same as in getSchedulingRule()

Available since version 5.0

Description

```
array ZendJobQueue::getSchedulingRule (int $rule_id)
```

Parameters

rule_id
: The rule identifier

Return Value

Information about the scheduling rule

ZendJobQueue::deleteSchedulingRule

Deletes the scheduling rule identified by the given $rule_id and scheduled jobs created by this rule

Available since version 5.0

Description

```
boolean ZendJobQueue::deleteSchedulingRule (int $rule_id)
```

Parameters

rule_id
: The rule identifier

Return Value

If scheduling rule was deleted or not deleted

ZendJobQueue::suspendSchedulingRule

Suspends the scheduling rule identified by given $rule_id and deletes scheduled jobs created by this rule

Available since version 5.0

Description

```
boolean ZendJobQueue::suspendSchedulingRule (int $rule_id)
```

Parameters

rule_id
    The rule identifier

Return Value

If scheduling rule was suspended or not suspended

ZendJobQueue::resumeSchedulingRule

Resumes the scheduling rule identified by given $rule_id and creates a corresponding scheduled job

Available since version 5.0

Description

```
boolean ZendJobQueue::resumeSchedulingRule (int $rule_id)
```

Parameters

rule_id
    The rule identifier

Return Value

If the scheduling rule was resumed or not resumed

ZendJobQueue::updateSchedulingRule

Updates and reschedules the existing scheduling rule

Available since version 5.0

Description

```
boolean ZendJobQueue::updateSchedulingRule (int $rule_id, string
$script, array $vars, array $options)
```

Parameters

rule_id
: The rule identifier

script
: The URL to request

vars
: The input variables

options
: The same as in createHttpJob()

Return Value

If scheduling rule was updated or not updated

ZendJobQueue::getCurrentJobParams

Decodes an array of input variables passed to the HTTP job

Available since version 5.0

Description

```
static array ZendJobQueue::getCurrentJobParams (void)
```

Return Value

The job variables

ZendJobQueue::setCurrentJobStatus

Reports job completion status (OK or FAILED) back to the daemon

Available since version 5.0

Description

```
static void ZendJobQueue::setCurrentJobStatus (int $completion, string
$msg)
```

Parameters

completion
: The job completion status (OK or FAILED)

msg
: The optional explanation message

Zend Job Queue Daemon - Configuration Directives

Configuration Directives Summary

Directive	Type	Description
zend_jobqueue.database	string	A database connection string in PDO like format
zend_jobqueue.binding	string	An address of TCP or UNIX socket to listen for requests from clients and management GUI
zend_jobqueue.max_http_jobs	integer	The maximum number of HTTP based jobs which can be executed simultaneously by single back-end server
zend_jobqueue.history	integer	The maximum time (in days) a completed, failed or removed job is kept in database.
zend_jobqueue.history_failed	integer	The maximum time (in days) a failed job is kept in database. If it is not set - the 'history' value is used.
zend_jobqueue.client_keep_alive	integer	Number of second while daemon keeps inactive connection from client.
zend_jobqueue.client_read_timeout	integer	Number of second while daemon is trying to read request from client.
zend_jobqueue.client_write_timeout	integer	Number of seconds while daemon is trying to deliver response to client.
zend_jobqueue.connection_timeout	integer	Number of seconds while daemon trying to establish a connection with back-end server
zend_jobqueue.http_job_timeout	integer	Number of seconds while URL based job must complete.
zend_jobqueue.job_restart_timeout	integer	The minimal number of milliseconds between job startups.
zend_jobqueue.http_job_retry_count	integer	Number of retries in case of HTTP job failure.
zend_jobqueue.http_job_retry_timeout	integer	The number of seconds between retries of failed HTTP jobs.
zend_jobqueue.high_concurrency_margin_allowed	integer	Report an event when the Job Queue Daemon reaches a margin between the number of running jobs and the maximum allowed
zend_jobqueue.job_time_skew_allowed	integer	Report an event when a job is "skewing" from its defined execution time
zend_jobqueue.max_message_size	integer	The maximum message size the daemon can accept from the extension
zend_jobqueue.log_verbosity_level	integer	The Log's verbosity level
zend_jobqueue.log_rotation_size	integer	The maximum size of the log file before it is rotated
zend_jobqueue.global_directives_ini_file	string	Global Directives ini File

Configuration Directive Details

zend_jobqueue.database

A database connection string in PDO like format. Relative sqlite path will be treated as relative to Zend Server data directory.

Type: string

Default Value: "sqlite:file=jobqueue.db"

Available since version 5.0

zend_jobqueue.binding

An address of TCP or UNIX socket to listen for requests from clients and management GUI

Type: string

Default Values:

- unix://jobqueue.sock
- Windows: tcp://127.0.0.1:10085

Available since version 5.0

zend_jobqueue.max_http_jobs

The maximum number of HTTP based jobs which can be executed simultaneously by single back-end server

Type: integer

Default Value: 4

Available since version 5.0

zend_jobqueue.history

The maximum time (in days) a completed, failed or removed job is kept in database. If no directive is provided time is unlimited and jobs are never deleted. Independently on this directive setting jobs may be kept forever using "persistent" option.

Type: integer

Units: days

Default Value: 7

Available since version 5.0

zend_jobqueue.history_failed

The maximum time (in days) a failed job is kept in database. If it is not set - the 'history' value is used.

Type: integer

Units: days

Available since version 5.0

zend_jobqueue.client_keep_alive

Number of second while daemon keeps inactive connection from client. In case client doesn't send any request during this time daemon closes the client's connection. (default 3600 seconds = 1 hour)

Type: integer

Default Value: 3600

Available since version 5.0

zend_jobqueue.client_read_timeout

Number of second while daemon is trying to read request from client. In case client doesn't respond in this time daemon closes the client's connection. (default 10 seconds)

Type: integer

Default Value: 30

Available since version 5.0

zend_jobqueue.client_write_timeout

Number of seconds while daemon is trying to deliver response to client. In case client doesn't respond in this time daemon closes the client's connection. (default 10 seconds)

Type: integer

Default Value: 30

Available since version 5.0

zend_jobqueue.connection_timeout

Number of seconds while daemon trying to establish a connection with back-end server

Type: integer

Units: seconds

Default Value: 30

Available since version 5.0

zend_jobqueue.http_job_timeout

Number of seconds while URL based job must complete. After timeout expiration daemons drops the connection to back-end server and sets job status to "failed" and completion status to "timeout".

Type: integer

Units: seconds

Default Value: 120

Available since version 5.0

zend_jobqueue.job_restart_timeout

The minimal number of microlliseconds between job startups.

Type: integer

Default Value: 200

Available since version 5.0

zend_jobqueue.http_job_retry_count

Number of retries in case of HTTP job failure.

Type: integer

Default Value: 10

Available since version 5.0

zend_jobqueue.http_job_retry_timeout

The number of seconds between retries of failed HTTP jobs.

Type: integer

Units: seconds

Default Value: 1

Available since version 5.0

zend_jobqueue.high_concurrency_margin_allowed

Report an event when the Job Queue Daemon reaches a margin between the number of running jobs and the maximum allowed

Type: integer

Default Value: 0

Available since version 5.0

zend_jobqueue.job_time_skew_allowed

Report an event when a job is "skewing" from its defined execution time

Type: integer

Units: seconds

Default Value: 120

Available since version 5.0

zend_jobqueue.max_message_size

The maximum message size the daemon can except from the extension

Type: integer

Units: KBytes

Default Value: 64

Available since version 5.0

zend_jobqueue.log_verbosity_level

The Log's verbosity level

Type: integer

Default Value: 2

Available since version 5.0

zend_jobqueue.log_rotation_size

The maximum size of the log file before it is rotated

Type: integer

Units: MBytes

Default Value: 10

Available since version 5.0

zend_jobqueue.global_directives_ini_file

The .ini file that contains the global directives, as defined in ZendGlobalDirectiveDD.xml

Type: string

Default Value: GLOBAL_DIRECTIVES_INI_FILE

Available since version 5.0

Zend Code Tracing - Configuration Directives

Configuration Directives Summary

Directive	Type	Modification Scope	Description
zend_codetracing.enable	boolean	ZEND_INI_PERDIR	Is tracing functionality enabled?
zend_codetracing.buffer_size	string	PHP_INI_SYSTEM	The size of the trace memory buffer
zend_codetracing.dump_format	integer	PHP_INI_ALL	The type of dump produced
zend_codetracing.max_string	integer	PHP_INI_ALL	The maximal length of the string before it is cut
zend_codetracing.max_depth	integer	PHP_INI_ALL	The maximal depth of the array preserved
zend_codetracing.max_elements	integer	PHP_INI_ALL	The maximal number of the array elements preserved
zend_codetracing.dump_file	string	PHP_INI_ALL	The prefix for the dump file names, relative to zend.data_dir.
zend_codetracing.trace_enable	boolean	PHP_INI_ALL	Tracing data collection is enabled
zend_codetracing.trace_time	boolean	PHP_INI_ALL	Timestamp collection is enabled
zend_codetracing.trace_source_lines	boolean	PHP_INI_ALL	Source file information collection is enabled
zend_codetracing.trace_internal_functions	boolean	PHP_INI_ALL	Internal functions call collection is enabled
zend_codetracing.trace_user_functions	boolean	PHP_INI_ALL	User functions call collection is enabled
zend_codetracing.trace_includes	boolean	PHP_INI_ALL	Include/require data collection is enabled
zend_codetracing.trace_arguments	boolean	PHP_INI_ALL	Function call argument collection is enabled
zend_codetracing.trace_return_values	boolean	PHP_INI_ALL	Function return value collection is enabled
zend_codetracing.trace_exceptions	boolean	PHP_INI_ALL	PHP exception data collection is enabled
zend_codetracing.trace_arrays	boolean	PHP_INI_ALL	Array contents recording is enabled
zend_codetracing.trace_write	boolean	PHP_INI_ALL	Output data (writing) collection is enabled
zend_codetracing.trace_headers	boolean	PHP_INI_ALL	Output headers collection is enabled

zend_codetracing.trace_memory_usage	boolean	PHP_INI_ALL	Memory usage data collection is enabled
zend_codetracing.trace_errors	boolean	PHP_INI_ALL	PHP error collection is enabled
zend_codetracing.trace_events	boolean	PHP_INI_ALL	Zend Monitor event recording is enabled
zend_codetracing.always_dump	boolean	PHP_INI_ALL	Trace data is always persisted
zend_codetracing.dump_on_segv	boolean	PHP_INI_ALL	Trace data is persisted if fatal signal (like Segmentation Fault) happens
zend_codetracing.max_freq	integer	PHP_INI_SYSTEM	Minimum amount of seconds between successive dumps.
zend_codetracing.log_file	string	PHP_INI_SYSTEM	Log file for the Code Tracing module (relative to the log dir).
zend_codetracing.log_verbosity	integer	PHP_INI_SYSTEM	Logging verbosity level.
zend_codetracing.override_functions	string	PHP_INI_SYSTEM	List of overriding functions.
zend_codetracing.max_concurrent_trace_buffers	integer	PHP_INI_ALL	Number of concurrent trace buffer allocated
zend_codetracing.log_rotation_size	integer	PHP_INI_ALL	The maximum size of the log file before it is rotated

Configuration Directive Details

zend_codetracing.enable

Is tracing functionality enabled?

Type: boolean

Default Value: 1

Available since version 4.0

zend_codetracing.buffer_size

The size of the trace memory buffer

Type: string

Units: Bytes

Default Value: 1M

Available since version 4.0

zend_codetracing.dump_format

The type of dump produced

Type: integer

Default Value: 2

Available since version 4.0

zend_codetracing.max_string

The maximal length of the string before it is cut

Type: integer

Units: Bytes

Default Value: 48

Available since version 4.0

zend_codetracing.max_depth

The maximal depth of the array preserved

Type: integer

Default Value: 2

Available since version 4.0

zend_codetracing.max_elements

The maximal number of the array elements preserved

Type: integer

Default Value: 10

Available since version 4.0

zend_codetracing.dump_file

The prefix for the dump file names, relative to zend.data_dir.

Type: string

Default Value: codetracing/dump

Available since version 4.0

zend_codetracing.trace_enable

Tracing data collection is enabled

Type: boolean

Default Value: 0

Available since version 4.0

zend_codetracing.trace_time

Timestamp collection is enabled

Type: boolean

Default Value: 1

Available since version 4.0

zend_codetracing.trace_source_lines

Source file information collection is enabled

Type: boolean

Default Value: 1

Available since version 4.0

zend_codetracing.trace_internal_functions

Internal functions call collection is enabled

Type: boolean

Default Value: 1

Available since version 4.0

zend_codetracing.trace_user_functions

User functions call collection is enabled

Type: boolean

Default Value: 1

Available since version 4.0

zend_codetracing.trace_includes

Include/require data collection is enabled

Type: boolean

Default Value: 1

Available since version 4.0

zend_codetracing.trace_arguments

Function call argument collection is enabled

Type: boolean

Default Value: 1

Available since version 4.0

zend_codetracing.trace_return_values

Function return value collection is enabled

Type: boolean

Default Value: 1

Available since version 4.0

zend_codetracing.trace_exceptions

PHP exception data collection is enabled

Type: boolean

Default Value: 1

Available since version 4.0

zend_codetracing.trace_arrays

Array contents recording is enabled

Type: boolean

Default Value: 1

Available since version 4.0

zend_codetracing.trace_write

Output data (writing) collection is enabled

Type: boolean

Default Value: 1

Available since version 4.0

zend_codetracing.trace_headers

Output headers collection is enabled

Type: boolean

Default Value: 1

Available since version 4.0

zend_codetracing.trace_memory_usage

Memory usage data collection is enabled

Type: boolean

Default Value: 1

Available since version 4.0

zend_codetracing.trace_errors

PHP error collection is enabled

Type: boolean

Default Value: 1

Available since version 4.0

zend_codetracing.trace_events

Zend Monitor event recording is enabled

Type: boolean

Default Value: 1

Available since version 4.0

zend_codetracing.always_dump

Trace data is always persisted

Type: boolean

Default Value: 0

Available since version 4.0

zend_codetracing.dump_on_segv

Trace data is persisted if fatal signal (like Segmentation Fault) happens

Type: boolean

Default Value: 1

Available since version 4.0

zend_codetracing.max_freq

Minimum amount of seconds between successive dumps.

Type: integer

Default Value: 1

Available since version 4.0

zend_codetracing.log_file

Log file for the Code Tracing module (relative to the log dir).

Type: string

Default Value: codetracing.log

Available since version 4.0

zend_codetracing.log_verbosity

Logging verbosity level.

Type: integer

Default Value: 0

Available since version 4.0

zend_codetracing.override_functions

List of overriding functions. Can contain function names, wildcards (prefix_*) and references to files (@filename). Entries are separated by comma. File contains one entry per line, which can be function name or wildcard.

Type: string

Default Value: @ZEND_PREFIX/etc/functions.txt

Available since version 4.0

zend_codetracing.max_concurrent_trace_buffers

Number of concurrent trace buffer allocated

Type: integer

Default Value: 5

Available since version 4.0

zend_codetracing.log_rotation_size

The maximum size of the log file before it is rotated

Type: integer

Units: MBytes

Default Value: 10

Available since version 4.0

Zend Session Clustering - Configuration Directives

Configuration Directives Summary

Directive	Type	Modification Scope	Description
zend_sc.enable	boolean	PHP_INI_SYSTEM	Enables the Session Clustering
zend_sc.scd_port	integer	PHP_INI_SYSTEM	the port the sc daemon is listening to
zend_sc.use_unix_sockets	boolean	PHP_INI_SYSTEM	Communication with mod cluster: 0 TCP, 1 Unix socket
zend_sc.unix_socket_filename	string	PHP_INI_SYSTEM	unix socket filename to communicate with daemon
zend_sc.lock_timeout	integer	PHP_INI_SYSTEM	used when retrying to access locked sessions
zend_sc.log_rotation_size	integer	PHP_INI_SYSTEM	size of log before it's rotated
zend_sc.log_verbosity_level	integer	PHP_INI_SYSTEM	The log verbosity level [0-5]

Configuration Directive Details

zend_sc.enable

Enables the Session Clustering

Type: boolean

Default Value: 0

Available since version 4.0

zend_sc.scd_port

the port the sc daemon is listening to

Type: integer

Default Value: 10062

Available since version 4.0

zend_sc.use_unix_sockets

Communication with mod cluster: 0 TCP, 1 Unix socket

Type: boolean

Default Value: 1

Available since version 4.0

zend_sc.unix_socket_filename

unix socket filename to communicate with daemon

Type: string

Default Value: scd.sock

Available since version 4.0

zend_sc.lock_timeout

used when retrying to access locked sessions

Type: integer

Units: seconds

Default Value: 1

Available since version 4.0

zend_sc.log_rotation_size

size of log before it's rotated

Type: integer

Default Value: 10

Available since version 4.0

zend_sc.log_verbosity_level

The extension's log verbosity level. Level 1 includes very important info messages, errors and warnings. Level 2 displays notices. Greater levels (up to 5) server debug purposes only.

Type: integer

Default Value: 2

Available since version 4.0

Web API Reference Guide

About

The Zend Server Web API allows automation of the management and deployment of Zend Server and Zend Server Cluster Manager, and allows integration with other Zend or 3rd party software.

The Web API Reference Guide includes information about:

- Generic Request/Response Format
- API Versioning
- Authentication and Message Verification
- Data Types
- Available API Methods

Generic Request/Response Format

This section describes the generic formatting of all Zend Server Web API requests and responses, regardless of the specific method used.

All Web API HTTP requests and response content will be encoded using UTF-8 character encoding.

This section includes:

Request Format

About

The request format defines the required format for each request sent to Zend Server. The request format for all Zend Server Web API requests are formatted as described in this page, regardless of the specific method used.

Request Method, URL, and Headers

Web API HTTP requests use HTTP GET for read-only API calls, and HTTP POST for all state changing API calls.

The request URL is different for each action, and is in one of the following formats:

- Zend Server - Where <ACTION> is the action to perform (e.g. “disableServer”):

```
http://example.com:10081/ZendServer/Api/<ACTION>
```

- Zend Server Cluster Manager - Where <ACTION> is the action to perform (e.g. “disableServer”):

```
http://example.com:10081/ZendServerManager/Api/<ACTION>
```

All HTTP requests must include the following HTTP headers:

- **Date** - Contains the current date and time in the GMT time zone, in the format specified by the HTTP RFC for date fields (e.g. “Wed, 07 Jul 2010 17:10:55 GMT”). This value is used to verify the authenticity of the request, and is expected to be in sync with the server time (within 30 seconds).
- **User-agent** - The user agent string is logged by the server and used for message authenticity verification. It cannot be empty.
- **Host** - The HTTP host header is expected to be present and is used for message authenticity verification.
- **X-Zend-Signature** - The API key name and calculated request signature which is used to authenticate and validate the request. See Authentication and Message Verification for additional information on calculating the signature.

In addition, you should send the Accept HTTP request header to designate your supported API version(s). If the Accept header is missing, the server will fall back to the default API version. This is described in detail in API Versioning Negotiation.

For POST requests, including any parameters or payload, clients must set the Content-type header to either "application/x-www-form-urlencoded" or "multipart/form-data", depending on the payload. You must specify the size of the request body as required by the HTTP/1.1 protocol, that is by using the Content-length header, the Content-transfer-encoding: chunked header or simply by closing the connection.

Passing Request Parameters

API methods that require passing parameters may be passed in the following forms:

- For GET requests, parameters are passed in the URL query part (following the '?') in a URL-encoded format, similar to how HTML forms sent using the GET method are encoded.
- For POST requests, parameters should be passed in the request body, encoded using either the "application/x-www-form-urlencoded" (as specified by the HTML 4.01 standard) or "multipart/form-data" (as specified in RFC-2388) encoding methods.
- For some methods (namely methods that may transfer large amounts of binary data), the 'multipart/form-data' encoding method must be used.

Refer to specific method documentation for a list of required and optional parameters.

Examples

Example

The following is an example of a call to the (obviously fake) “makePizza” method (some lines are broken for readability):

```
POST /ZendServerManager/Api/makePizza HTTP/1.1
Host: zscm.local
Date: Sun, 11 Jul 2010 13:16:10 GMT
User-agent: Zend_Http_Client/1.10
Accept: application/vnd.zend.serverapi+xml;version=1.0
X-Zend-Signature: bob.at.example.com;
7f0db29a3d82a81ec6f5387f5aae96e295530b4c8acf2074488185902dc900f4
Content-type: application/x-www-form-urlencoded
Content-length: 100

style=thinCrust&extraCheese=TRUE&extras%5B0%5D=pepperoni&extras%5B1%
5D=onion&extras%5B2%5D=pineapple
```

The request above is for the “makePizza” method, with the following parameters: style, extraCheese, extras.

The following example shows a call to a read-only “getPizzaStatus” method:

```
GET /ZendServerManager/Api/getPizzaStatus?pizzaId=53 HTTP/1.1
Host: zscm.local|
Date: Sun, 11 Jul 2010 13:16:10 GMT
User-agent: Zend_Http_Client/1.10
Accept: application/vnd.zend.serverapi+xml;version=1.0
X-Zend-Signature: bob.at.example.com;
 02dcbf4cb338a0a8b807c83a84a7888929f5c06491105d6752f290da47a24619
```

Notice that the ‘pizzaId’ parameter is passed as part of the URL’s query string.

Response Format

About

API HTTP response messages use standard HTTP response codes to designate high-level status (success, failure, etc.) and simple XML payload in the response body to provide additional method specific data or specific error messages.

HTTP Response Codes

Zend Server Web API operations will return standard HTTP response codes in order to indicate overall success or failure. In case of an error, the HTTP status code may further indicate the nature of the problem, and a specific error code contained in the response body will indicate the specific nature of the error.

The following HTTP response codes may be used:

- **200 OK** - The operation has completed successfully.
- **202 Accepted** - The operation was accepted and is being processed, but is not complete yet.
- **4xx** - HTTP status codes between 400 and 499 are used to designate a client-side error. For example, a missing request parameter or an authentication error.
- **5xx** - HTTP status codes between 500 and 599 are used to designate a server-side error. For example, a temporary locking issue or an unexpected error in the server operation.

Additional information about HTTP response codes for error responses is available in Error Responses, as well as in the documentation for each method.

HTTP Response Headers

The following HTTP response headers will be included in API responses:

- **Content-type** - Unless stated otherwise, these will be “application/vnd.zend.serverapi+xml; version=<API version>”. For more information about API versions see API Versioning Negotiation.

HTTP Response Body

The Web API HTTP response body will almost always contain an XML document of the following format:

```
<?xml version="1.0" encoding="UTF-8"?>
<zendServerAPIResponse
  xmlns=“http://www.zend.com/server/api/1.0”>

  <requestData>
    <apiKeyName>angel.eyes</apiKeyName>
    <method>clusterGetServerStatus</method>
  </requestData>

  <responseData>
    [response data here...]
  </responseData>
</zendServerAPIResponse>
```

All API responses are enclosed in <zendServerAPIResponse> tags, and contain two sections: <requestData> which includes some reference information about the request; and <responseData> which includes the response data.

The content of the <responseData> section differs depending on the specific method called. Refer to the method's specific documentation for additional information.

In error responses, the <responseData> section is replaced with an <errorData> section. See Error Responses for additional information.

Some Web API methods will not return an XML document in case of a successful operation - specifically, methods that export large amounts of binary data such as the configurationExport method. In such cases, this is specifically indicated in the method's documentation. Note that you can check the value of the Content-type response header in order to know in advance what kind of content to expect in the response.

Error Responses

In a response representing an error, the <responseData> XML section is replaced with the <errorData> XML section, which has the following format:

```
<errorData>
  <errorCode>serverDoesNotExist</errorCode>
  <errorMessage>A server with the specified ID does not exist in the
cluster</errorMessage>
</errorData>
```

Where:

- <errorCode> is a short alphanumeric constant string that represents the specific error.
- <errorMessage> is a human readable, native language explanation of the error.

In addition, some error responses may include additional elements in the <errorData> container, with additional information relevant to the specific error.

Generic Error Codes

The following generic error responses are possible for any operation:

HTTP Code	Error Code	Description
400	missingHttpHeader	A required HTTP header is missing.
400	unexpectedHttpMethod	An unexpected HTTP method where GET is used but POST is expected.
400	invalidParameter	One or more request parameters contains invalid data.
400	missingParameter	The request is missing a required parameter.
400	unknownMethod	An unknown Zend Server API method.
400	malformedRequest	The server is unable to understand the request.
401	authError	An authentication error occurred because of an unknown key or invalid request signature.
401	insufficientAccessLevel	The user is not authorized to perform this action.
401	timeSkewError	The request timestamp deviates too much from the server time.
405	notImplementedByEdition	The method is not implemented by this edition of Zend Server.
406	unsupportedApiVersion	The API version is not supported by this version of Zend Server.
500	internalServerError	An unexpected error occurred on the server side.
500	serverNotConfigured	This Zend Server installation was not yet initialized (the user did not go through the initial setup wizard).
500	serverNotLicensed	Zend Server does not have a valid license, which is required to perform this operation.

Refer to the documentation of a specific method for details about additional possible errors specific to each method.

API Versioning Negotation

As you perform an API call, it should specify its currently used API version as part of the Zend Server API media type in the Accept HTTP header. For example, sending a request using API version 3.0 should include the following Accept header in the request:

```
Accept: application/vnd.zend.serverapi+xml;version=3.0
```

If the server supports the specified API version, it will handle the request and respond in the appropriate format, matching the specified API version. The response format and API version will be specified using the Content-type response header:

```
Content-type: application/vnd.zend.serverapi+xml;version=3.0
```

If the server is not compatible with the API version being used, the server will return an HTTP 406 Not Acceptable”response, with supported version content types listed as part of the <errorData> XML.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<zendServerAPIResponse
  xmlns=”http://www.zend.com/server/api/1.0”>

  <requestData>
    <apiKeyName>angel.eyes</apiKeyName>
    <method>clusterGetServerStatus</method>
  </requestData>

  <errorData>
    <errorCode>unsupportedApiVersion</errorCode>
    <errorMessage>
      Client API version not supported by this server
    </errorMessage>
    <supportedApiVersions>
      application/vnd.zend.serverapi+xml;version=1.0,
      application/vnd.zend.serverapi+xml;version=1.1,
      application/vnd.zend.serverapi+xml;version=2.0
    </supportedApiVersions>
  </errorData>
</zendServerAPIResponse>
```

You can then choose to switch to a different API version, or give up and issue a failure message to the end user.

Note:

The Accept request header, while highly recommended, is optional. If you do not specify the Accept header, the server will fall back to using the oldest API version supported by the server.

Authentication and Message Verification

API request authentication is done by creating a digital signature of some request parameters using an account-specific secret key. This signature, as well as the key name is then sent in the custom X-Zend-Signature HTTP header.

The server will compare this signature with the expected signature (calculated based on the same key and parameters as known to the server) and will only authorize the request if the signatures match.

Note:

This authentication and validation method does not contradict the use of HTTPS to encrypt the communication channel, which is recommended but not required.

This section includes the information on the following:

- Generating API Keys
- Signing API Requests

Generating API Keys

API keys are generated using the Zend Server/Zend Server Cluster Manager GUI. When generating a new key, you must specify a name for this key. This may be a username or a group name, which is used to identify the key and to tell the server which key to use when attempting to authenticate a request.

Valid key names may be composed only of “token” characters as defined by RFC-2616, with the addition of the whitespace character and the ‘@’ character. This allows all printable US-ASCII characters (ASCII characters 0x20 to 0x7e) with the exception of the following characters:

```
( ) < > , ; : \ " / [ ] ? = { }
```

In addition, key names may not begin or end with a whitespace character.

The specific API key also determines the access level granted when using this key.

Note:

Zend Server API keys must be kept secret and immediately revoked if there is any chance that they have been compromised.

Signing API Requests

Importance of the Date Header

The value of the Date HTTP header is used as part of the request signing process to enforce the temporary state of signed requests. For this reason, the system clock on the client and server sides must be synchronized, up to an allowed time skew of ±30 seconds.

If the server receives an API request with a Date header value that represents more than 30 seconds of time difference (either before or after the server clock), the request will not be accepted.

The X-Zend Signature HTTP Header

In order to send authenticated API requests you are required to send the X-Zend-Signature HTTP header with each request. It must be in the following format:

```
X-Zend-Signature: <key name>; <signature>
```

Where <key name> is replaced with the key name, and <signature> is replaced with the calculated request signature.

There can be any number of whitespace characters before or after the separating semicolon.

Example:

```
X-Zend-Signature: Arch Stanton;
  a0f9b1d61e21e796d78dccdf1352f23cd328...
```

Note:

The signature is expected to be 64 characters long, and is cut here for readability purposes.

Calculating the Request Signature

The request signature is a 64 digit long hexadecimal number with digits a-f in lower case, calculated using the following method:

1. Concatenate the following values in order, separated by a colon (:), into a single string:
    a. The exact value of the Host HTTP header. In most cases this will be a string in the form "<host>:<port>". In some cases the colon and port are omitted. In any case, if the port is included in the Host header sent in the request, it must be included in the generated string.
    b. The Request URI, which is the path part of the full request URL, without the query string or host name.
    c. The exact value of the User-Agent request header.
    d. The exact value of the Date request header.
2. Hash the generated string with the HMAC/SHA-256 function using the secret API key to obtain the request signature.

Example

When sending the following API request:

```
POST /ZendServer/Api/findTheFish HTTP/1.1
Host: zscm.local:10081
User-agent: Zend_Http_Client/1.10
Accept: application/vnd.zend.serverapi+xml;version=1.0
Date: Sun, 11 Jul 2010 13:16:10 GMT
Content-type: application/x-www-form-urlencoded
Content-length: 19
lookInCupboard=TRUE
```

Using a key named “angel.eyes” with the following value:

```
9dc7f8c5ac43bb2ab36120861b4aeda8f9bb6c521e124360fd5821ef279fd9c7
```

The request parameters to be signed, concatenated into a string is:

```
zscm.local:10081:/ZendServer/Api/findTheFish:Zend_Http_Client/1.10
: Sun, 11 Jul 2010 13:16:10 GMT
```

From this value, an HMAC/SHA-256 signature will be calculated using the API key. For example using the hash_hmac() PHP function:

```
785be59b7728b1bfd6495d610271c5d47ff0737775b09191daeb5a728c2d97c0
```

The final request, including the added X-Zend-Signature header, is (lines are broken for readability):

```
POST /ZendServer/Api/findTheFish HTTP/1.1
Host: zscm.local:10081
User-agent: Zend_Http_Client/1.10
Accept: application/vnd.zend.serverapi+xml;version=1.0
Date: Sun, 11 Jul 2010 13:16:10 GMT
Content-type: application/x-www-form-urlencoded
Content-length: 19
X-Zend-Signature: angel.eyes;
  785be59b7728b1bfd6495d610271c5d47ff0737775b09191daeb5a728c2d97c0
lookInCupboard=TRUE
```

The server then proceeds to generate the same signature, based on the same data and same secret key. If the two signatures match, the request will be accepted.

Data Types

The following section describes the different data types used for request parameters and response data enclosed in XML. The specific API methods documented in Available API Methods refer to the data types defined here.

The data types described here are:

- Request Data Types
- Response Data Types

Request Data Types

Request data may be encoded into several primitive types. Since all data is eventually represented as UTF-8 strings, these types mostly define what characters are considered valid for data of a specific type. Additional validation rules may apply for specific parameters.

- **Boolean** - A case insensitive Boolean value, represented as either "TRUE" or "FALSE".
- **Integer** - An integer (whole number).
- **String** - A string of characters.
- **TimeStamp** - The time and date represented in RFC-882/RFC-1123 format (e.g. "Sun, 06 Nov 1994 08:49:37 GMT"). The time and date must always be represented in the GMT time zone, even if the server or client uses a different default time zone.
- **Array** - An array of values. Arrays are encoded by adding square brackets with an incrementing 0-based index number to the parameter name. For example, the array parameter fruits = ("apple", "orange", "banana") is to be represented as follows:

```
fruits[0]=apple&fruits[1]=orange&fruits[2]=banana
```

Since request parameter names must be URL-encoded, the above parameter will actually be sent as:

```
fruits%5B0%5D=apple&fruits%5B1%5D=orange&fruits%5B2%5D=banana
```

Response Data Types

Response data types are represented in XML format. This allows for more complex object types to be represented in the response data.

The following response types are possible in addition to the basic types defined for request data types (Boolean, Integer, String, TimeStamp).

Each complex type is represented as an XML element, with properties represented as sub-elements.

XML element names always use camelCase notation (first character is lower case).

The following is a list of the available response data types:

- messageList
- serverInfo
- serversList
- systemInfo
- licenseInfo

messageList

A list of 0 or more messages.

Parameter	Type	Count	Description
Info	String	0+	An info-level message (may appear 0 or more times).
warning	String	0+	A warning-level message (may appear 0 or more times).
error	String	0+	An error-level message (may appear 0 or more times).

serverInfo

An object representing a single server with information about the server.

Parameter	Type	Count	Description
id	Integer	1	The server ID.
name	String	1	The server name.
address	String	1	The server address as an HTTP URL.
status	String	1	The server status, which may be one of the following values: ▪ OK ▪ shuttingDown ▪ startingUp ▪ pendingRestart ▪ restarting ▪ misconfigured ▪ extensionMismatch ▪ daemonMismatch ▪ notResponding ▪ disabled ▪ removed ▪ unknown
messageList	messageList	1	A list of messages reported by this server, which can be empty if there are no messages to show.

serversList

A list of servers.

Parameter	Type	Count	Description
serverInfo	serverInfo	0+	The server information (may appear more than once).

systemInfo

Generic information about the system being accessed.

Parameter	Type	Count	Description
status	String	1	The global status information, which can be one of the following: ▪ OK - The system is operational. ▪ notLicensed - The system is not licensed. In Zend Server Cluster Manager, this means the Zend Server Cluster Manager is not licensed, but the nodes may be licensed and operating. ▪ pendingRestart - The system is pending a PHP restart. In Zend Server Cluster Manager this will never be set.
edition	String	1	The Zend Server edition, which can be one of the following: ▪ ZendServer ▪ ZendServerClusterManager ▪ ZendServerCommunityEdition
zendServerVersion	String	1	The full version of Zend Server (e.g. “5.0.4”).
supportedApiVersions	String	1	A comma-separated list of the supported content types/versions of the Zend Server Web API.
phpVersion	String	1	The full PHP version (e.g. “5.3.3”).
operatingSystem	String	1	A string identifying the operating system.
serverLincenseInfo	licenseInfo	1	Information about the Zend Server license. If it is running in a cluster, it will contain the node license information.
managerLicenseInfo	licenseInfo	1	Information about the Zend Server Cluster Manager license.
messageList	messageList	1	A list of messages reported by this server, which is empty if there are no messages to show.

licenseInfo

Information about a Zend Server or Zend Server Cluster Manager license.

Parameter	Type	Count	Description
status	String	1	The licensing status, which can be one of the following: ▪ notRequired - This edition does not require this license type. ▪ OK - The server/cluster is licensed and working. ▪ invalid - The license is invalid. ▪ expired - The license has expired. ▪ serverLimitExceeded - The Zend Server Cluster Manager server limit has been exceeded.
orderNumber	String	1	The license order number, which is empty if there is no license.
validUntil	TimeStamp	1	The license expiration date, which is empty if there is no license.
serverLimit	Integer	1	For a Zend Server Cluster Manager license, this is the number of servers allowed by the license. For a license other than Zend Server Cluster Manager, the value is always 0.

Available API Methods

The following section describes the specific operations that can be performed using the Web API. Each method carries a different operation or is designed to retrieve specific information from Zend Server or Zend Server Cluster Manager. You an use the information described for each method together with the structure defined in Generic Request/Response Format to execute the available API methods.
The following information is described for each method:

- Required permissions - The API key access level required to perform this action.
- HTTP method - Defines whether HTTP GET or POST should be used for this action.
- Supported by editions - Lists the Zend Server editions that can perform this action.
- Request parameters - The list of required and optional parameters accepted by this action. The type of each parameter corresponds to one of the types defined in Request Data Types.
- Expected response code - Lists the HTTP response code that can be returned in case of a successful operation.
- Response type - Defines the response type expected in case of successful operation. This corresponds to one of the types defined in Response Data Types.
- Possible action specific error codes - Lists the HTTP response code and error code that can be returned in case of an unsuccessful operation. Every method can receive any of the generic error codes in addition to its possible action specific error codes. For more information see Error Responses.
- Response format - This is only defined in methods which return a response format different from the generic response format.

The methods described here are:

- Server and Cluster Management Methods
- Configuration Management Methods

Server and Cluster Management Methods

The following is a list of the available methods used to manage your server and/or cluster:

- getSystemInfo
- clusterGetServerStatus
- clusterAddServer
- clusterRemoveServer
- clusterDisableServer
- clusterEnableServer
- restartPHP

getSystemInfo

Use this method to get information about the system, including the Zend Server edition and version, PHP version, licensing information, etc. This method is produces similar output on all Zend Server systems, and is future compatible.

Required Permissions: read

HTTP method: GET

Supported by Editions: All

Request Parameters: This method has no request parameters.

Expected Response Code: 200 OK

Response Type: systemInfo

Possible Action Specific Error Codes: This method has no action-specific error codes.

Example:

Request:

```
GET /ZendServerManager/Api/getSystemInfo
```

Response:

```
<?xml version="1.0" encoding="UTF-8"?>
<zendServerAPIResponse xmlns="http://www.zend.com/server/api/1.0">
  <requestData>
    <apiKeyName>angel.eyes</apiKeyName>
    <method>getSystemInfo</method>
  </requestData>
  <responseData>
    <systemInfo>
      <status>OK</status>
      <edition>
        ZendServerClusterManager
      </edition>
      <zendServerVersion>6.0.1</zendServerVersion>
      <supportedApiVersions>
        application/vnd.zend.serverapi+xml;version=1.0,
        application/vnd.zend.serverapi+xml;version=1.1,
        application/vnd.zend.serverapi+xml;version=2.0
      </supportedApiVersions>
      <phpVersion>5.4.1</phpVersion>
      <operatingSystem>Linux</operatingSystem>
      <serverLicenseInfo>
        <status>OK</status>
        <orderNumber>ZEND-ORDER-66</orderNumber>
        <validUntil>Sat, 31 Mar 2012 00:00:00 GMT</validUntil>
        <serverLimit>0</serverLimit>
      </serverLicenseInfo>
      <managerLicenseInfo>
        <status>serverLimitExceeded</status>
        <orderNumber>ZEND-ORDER-66</orderNumber>
        <validUntil>Sat, 31 Mar 2012 00:00:00 GMT</validUntil>
        <serverLimit>10</serverLimit>
      </managerLicenseInfo>
      <messageList />
    </systemInfo>
  </responseData>
</zendServerAPIResponse>
```

An example response for the same request sent to a Zend Server Community Edition Machine would be:

```
<?xml version="1.0" encoding="UTF-8"?>
<zendServerAPIResponse xmlns="http://www.zend.com/server/api/1.0">
  <requestData>
    <apiKeyName>angel.eyes</apiKeyName>
    <method>getSystemInfo</method>
  </requestData>
  <responseData>
    <systemInfo>
      <status>OK</status>
      <edition>
        ZendServerCommunityEdition
      </edition>
      <zendServerVersion>6.0.1</zendServerVersion>
      <supportedApiVersions>
        application/vnd.zend.serverapi+xml;version=1.0,
        application/vnd.zend.serverapi+xml;version=1.1,
        application/vnd.zend.serverapi+xml;version=2.0
      </supportedApiVersions>
      <phpVersion>5.4.1</phpVersion>
      <operatingSystem>Linux</operatingSystem>
      <serverLicenseInfo>
        <status>notRequired</status>
        <orderNumber />
        <validUntil />
        <serverLimit>0</serverLimit>
      </serverLicenseInfo>
      <managerLicenseInfo>
        <status>notRequired</status>
        <orderNumber />
        <validUntil />
        <serverLimit>0</serverLimit>
      </managerLicenseInfo>
      <messageList />
    </systemInfo>
  </responseData>
</zendServerAPIResponse>
```

clusterGetServerStatus

Use this method to get the list of servers in the cluster and the status of each one. On a Zend Server Cluster Manager with no valid license, this operation fails. This operation causes Zend Server Cluster Manager to check the status of servers and return fresh, non-cached information. This is different from the Servers List tab in the GUI, which may present cached information. Users interested in reducing load by caching this information should do it in their own code.

Required Permissions: read

HTTP method: GET

Supported by Editions: Zend Server Cluster Manager

Request Parameters:

Parameter	Type	Required	Description
servers	Array	No	A list of server IDs. If specified, the status is returned for these servers only. If not specified, the status of all the servers is returned.

Expected Response Code: 200 OK

Response Type: serversList

Possible Action Specific Error Codes:

HTTP Code	Error Code	Description
404	noSuchServer	One or more of the provided server IDs does not exist in the cluster.
405	notImplementedByEdition	This method is only available on Zend Server Cluster Manager.
500	serverNotLicensed	Zend Server Cluster Manager is not licensed.

Example:

Request (URI broken for readability):

```
GET /ZendServerManager/Api/clusterGetServerStatus?
  servers%5B0%5D=12&servers%5B1%5D=15
```

Response:

```
<?xml version="1.0" encoding="UTF-8"?>
<zendServerAPIResponse xmlns="http://www.zend.com/server/api/1.0">

  <requestData>
    <apiKeyName>angel.eyes</apiKeyName>
    <method>clusterGetServersStatus</method>
  </requestData>

  <responseData>
    <serversList>
      <serverInfo>
        <id>12</id>
        <name>www-01</name>
        <address>https://www-01.local:10082/ZendServer</address>
        <status>OK</status>
        <messageList />
      </serverInfo>
      <serverInfo>
        <id>15</id>
        <name>www-02</name>
        <address>https://www-02.local:10082/ZendServer</address>
        <status>pendingRestart</status>
        <messageList>
          <warning>This server is waiting a PHP restart</warning>
        </messageList>
      </serverInfo>
    </serversList>
  </responseData>
</zendServerAPIResponse>
```

clusterAddServer

Add a new server to the cluster. On a Zend Server Cluster Manager with no valid license, this operation fails.

Required Permissions: full

HTTP method: POST

Supported by Editions: Zend Server Cluster Manager

Request Parameters:

Parameter	Type	Required	Description
serverName	String	Yes	The server name.
serverUrl	String	Yes	The server address as a full HTTP/HTTPS URL.
guiPassword	String	Yes	The server GUI password.
propagateSettings	Boolean	No	Propagate this server's current settings to the rest of the cluster. The default value is "FALSE".
doRestart	Boolean	No	Initiate a PHP restart on the cluster after adding the server. The default value is "FALSE".

Expected Response Code:

- 200 OK - The operation was successful.
- 202 Accepted - The server was added successfully, but setting propagation failed. (This can only happen if propagateSettings was set to "TRUE".)

Response Type: serverInfo with information about the just-added server.

Possible Action Specific Error Codes:

HTTP Code	Error Code	Description
500	cantConnectToServer	Zend Server Cluster Manager is unable to connect to the specified server URL.
500	invalidServerResponse	An invalid or unexpected response from a new server.
400	wrongPassword	The provided GUI password is incorrect.
400	alreadyConnected	The server is already a member of a cluster (not necessarily the current cluster).
503	temporarilyLocked	The server cannot be added because a cluster member is in graceful startup/shutdown mode.
500	noActiveServers	The server cannot be added because all servers in the cluster are disabled or unreachable.
500	serverNotLicensed	Zend Server Cluster Manager is not licensed.
405	notImplementedByEdition	This method is only available on Zend Server Cluster Manager.

Example:

Request (headers removed for clarity):

```
POST /ZendServerManager/Api/clusterAddServer
serverName=www-05&serverURL=https://www-05.local:10081/ZendServer&
guiPassword=somepassword&doRestart=TRUE
```

Response:

```
<?xml version="1.0" encoding="UTF-8"?>
<zendServerAPIResponse xmlns="http://www.zend.com/server/api/1.0">

  <requestData>
    <apiKeyName>angel.eyes</apiKeyName>
    <method>clusterAddServer</method>
  </requestData>

  <responseData>
    <serverInfo>
      <id>25</id>
      <name>www-05</name>
      <address>https://www-05.local:10082/ZendServer</address>
      <status>OK</status>
      <messageList />
    </serverInfo>
  </responseData>
</zendServerAPIResponse>
```

clusterRemoveServer

This method removes a server from the cluster. The removal process may be asynchronous if Session Clustering is used. If this is the case, the initial operation will return an HTTP 202 response. As long as the server is not fully removed, further calls to remove the same server should be idempotent. On a Zend Server Cluster Manager with no valid license, this operation fails.

Required Permissions: full

HTTP method: POST

Supported by Editions: Zend Server Cluster Manager

Request Parameters:

Parameter	Type	Required	Description
serverId	String	Yes	The server ID
force	Boolean	No	Force-remove the server, skipping graceful shutdown process. Default is FALSE

Expected Response Code:

- 200 OK - The server removal process was completed successfully. This status is expected if there is no need to perform a graceful shutdown process, or if the Force option was set to "TRUE".
- 202 Accepted - The removal process has started but not completed yet. The user may want to check the server status within a few seconds using the clusterGetServerStatus method to verify that the operation was completed.

Response Type: serverInfo with the status of the server being removed. The status is expected to be either *shuttingDown* or *removed*.

Possible Action Specific Error Codes:

HTTP Code	Error Code	Description
404	noSuchServer	There is no server with the provided server ID.
500	cantConnectToServer	Zend Server Cluster Manager is unable to connect to the specified server.
500	invalidServerResponse	An invalid or unexpected response from the server.
503	temporarilyLocked	The server cannot be removed because another server in the cluster is performing a graceful startup/shutdown.
405	notImplementedByEdition	The method is not implemented by this edition of Zend Server.
500	serverNotLicensed	Zend Server Cluster Manager is not licensed.

Example:

Request (headers removed for clarity):

```
POST /ZendServerManager/Api/clusterRemoveServer
serverId=5
```

Response:

```
HTTP/1.0 202 Accepted
<?xml version="1.0" encoding="UTF-8"?>
<zendServerAPIResponse xmlns="http://www.zend.com/server/api/1.0">

  <requestData>
    <apiKeyName>angel.eyes</apiKeyName>
    <method>clusterRemoveServer</method>
  </requestData>

  <responseData>
    <serverInfo>
      <id>5</id>
      <name>www-02</name>
      <address>https://www-02.local:10082/ZendServer</address>
      <status>shuttingDown</status>
      <messageList />
    </serverInfo>
  </responseData>
</zendServerAPIResponse>
```

clusterDisableServer

This method disables a cluster member. This process may be asynchronous if Session Clustering is used. If this is the case, the initial operation returns an HTTP 202 response. As long as the server is not fully disabled, further calls to this method are idempotent. On a Zend Server Cluster Manager with no valid license, this operation fails.

Required Permissions: full
HTTP method: POST
Supported by Editions: Zend Server Cluster Manager
Request Parameters:

Parameter	Type	Required	Description
serverId	String	Yes	The server ID

Expected Response Code:

- 200 OK - The process was completed successfully. This status is expected if there is no need to perform a graceful shutdown process.
- 202 Accepted - The disabling process has started but was not completed yet. You can check the server status within a few seconds using the clusterGetServerStatus method to verify that the operation is complete.

Response Type: serverInfo with the status of the server being disabled. The status is either *shuttingDown* or *disabled*. On a Zend Server Cluster Manager with no valid license, this operation fails.
Possible Action Specific Error Codes:

HTTP Code	Error Code	Description
404	noSuchServer	There is no server with the provided server ID.
500	cantConnectToServer	Zend Server Cluster Manager is unable to connect to the specified server.
500	invalidServerResponse	An invalid or unexpected response from the server.
503	temporarilyLocked	The server cannot be disabled because another server in the cluster is performing a graceful startup/shutdown.
405	notImplementedByEdition	The method is not implemented by this edition of Zend Server.
500	serverNotLicensed	Zend Server Cluster Manager is not licensed.

Example:

Request (headers removed for clarity):

```
POST /ZendServerManager/Api/clusterDisableServer
serverID=5
```

Response:

```
HTTP/1.0 200 OK
<?xml version="1.0" encoding="UTF-8"?>
<zendServerAPIResponse xmlns="http://www.zend.com/server/api/1.0">

  <requestData>
    <apiKeyName>angel.eyes</apiKeyName>
    <method>clusterDisableServer</method>
  </requestData>

  <responseData>
    <serverInfo>
      <id>5</id>
      <name>www-02</name>
      <address>https://www-02.local:10082/ZendServer</address>
      <status>disabled</status>
      <messageList />
    </serverInfo>
  </responseData>
</zendServerAPIResponse>
```

clusterEnableServer

This method is used to re-enable a cluster member. This process may be asynchronous if Session Clustering is used. If this is the case, the initial operation will return an HTTP 202 response. This action is idempotent, and running it on an enabled server will result in a 200 OK response with no consequences. On a Zend Server Cluster Manager with no valid license this operation fails.

Required Permissions: full

HTTP method: POST

Supported by Editions: Zend Server Cluster Manager

Request Parameters:

Parameter	Type	Required	Description
serverId	String	Yes	The server ID

Expected Response Code:

- 200 OK - The server was enabled successfully. This status appears if the server is not re-joining the cluster after performing a graceful shutdown and has no sessions to reclaim.
- 202 Accepted - The process started but has not completed yet. You can check the server status within a few seconds using the clusterGetServerStatus method to verify that the operation is complete.

Response Type: serverInfo with the status of the server being *enabled*. Status is expected to be either *startingUp* if the server is in the process of re-joining the cluster, or any other active status (*OK, pendingRestart, misconfigured, extensionMismatch, daemonMismatch, notResponding*) if the process was completed.

Possible Action Specific Error Codes:

HTTP Code	Error Code	Description
404	noSuchServer	There is no server with the provided server ID.
500	cantConnectToServer	Zend Server Cluster Manager is unable to connect to the specified server.
500	invalidServerResponse	An invalid or unexpected response from the server.
503	temporarilyLocked	The server cannot be disabled because another server in the cluster is performing a graceful startup/shutdown.
405	notImplementedByEdition	The method is not implemented by this edition of Zend Server.
500	serverNotLicensed	Zend Server Cluster Manager is not licensed.

Example:

Request (headers removed for clarity):

```
POST /ZendServerManager/Api/clusterEnableServer
serverID=5
```

Response:

```
HTTP/1.0 200 OK
<?xml version="1.0" encoding="UTF-8"?>
<zendServerAPIResponse xmlns="http://www.zend.com/server/api/1.0">

  <requestData>
    <apiKeyName>angel.eyes</apiKeyName>
    <method>clusterEnableServer</method>
  </requestData>

  <responseData>
    <serverInfo>
      <id>5</id>
      <name>www-02</name>
      <address>https://www-02.local:10082/ZendServer</address>
      <status>pendingRestart</status>
      <messageList />
    </serverInfo>
  </responseData>
</zendServerAPIResponse>
```

restartPHP

This method restarts PHP on all servers or on specified servers in the cluster. A 202 response in this case does not always indicate a successful restart of all servers. Use the clusterGetServerStatus command to check the server(s) status again after a few seconds.

Required Permissions: full

HTTP method: POST

Supported by Editions: All

Request Parameters:

Parameter	Type	Required	Description
servers	Array	No	A list of server IDs to restart. If not specified, all servers in the cluster will be restarted. In a single Zend Server context this parameter is ignored.
parallelRestart	Boolean	No	Sends the restart command to all servers at the same time. The default value is "FALSE".

Expected Response Code: 202 Accepted

Response Type: serversList with the status of all servers to which the restart command was requested (i.e. the servers provided in the servers parameter or all servers if no servers are specified).

Possible Action Specific Error Codes:

HTTP Code	Error Code	Description
404	noSuchServer	One or more of the provided server IDs does not exist. In this case, no servers are restarted.
500	restartFailed	Restarting at least some of the servers failed. This response is only possible when working with a cluster.

Example:

Request (headers removed for clarity):

```
POST /ZendServerManager/Api/restartPhp
servers%5B0%5D=1&servers%5B1%5D=2
```

Response:

```
HTTP/1.0 202 Accepted
<?xml version="1.0" encoding="UTF-8"?>
<zendServerAPIResponse xmlns="http://www.zend.com/server/api/1.0">

  <requestData>
    <apiKeyName>angel.eyes</apiKeyName>
    <method>restartPhp</method>
  </requestData>

  <responseData>
    <serversList>
      <serverInfo>
        <id>1</id>
        <name>www-01</name>
        <address>https://www-01.local:10082/ZendServer</address>
        <status>restarting</status>
        <messageList />
      </serverInfo>
      <serverInfo>
        <id>2</id>
        <name>www-02</name>
        <address>https://www-02.local:10082/ZendServer</address>
        <status>restarting</status>
        <messageList />
      </serverInfo>
      <serverInfo>
        <id>3</id>
        <name>www-03</name>
        <address>https://www-03.local:10082/ZendServer</address>
        <status>OK</status>
        <messageList />
      </serverInfo>
    </serversList>
  </responseData>
</zendServerAPIResponse>
```

Configuration Management Methods

The following is a list of the available methods used to manage your Zend Server or Zend Server Cluster Manager configuration:

- The configurationExport Method
- The configurationImport Method

The configurationExport Method

Export the current server/cluster configuration into a file.

Required Permissions: read

HTTP method: GET

Supported by Editions: All

Request Parameters: This method has no request parameters

Expected Response Code: 200 OK

Response Format: A successful call to the configurationExport method results in an HTTP response with the configuration snapshot file in the response body.

The content type for the configuration snapshot file is “application/vnd.zend.serverconfig”. In addition, the response includes a Content-disposition header, specifying a suggested file name for the configuration snapshot file.

Note:

This is different from most Web API calls where the content type is expected to be “application/vnd.zend.serverpi+xml; version=…” and the response body payload is expected to be in XML format.

In case of an error, a regular error response will be returned containing an <errorData> element as defined for other Web API methods.

Possible Action Specific Error Codes:

HTTP Code	Error Code	Description
500	exportFailed	Creating a configuration snapshot failed.

Example:

Request (headers removed for clarity):

```
GET /ZendServerManager/Api/configurationExport
```

Response (not all headers are shown):

```
HTTP/1.0 200 OK
Content-type: application/vnd.zend.serverconfig
Content-disposition: attachment;
  filename=”ZendServerConfig-20101123.zcfg”

[...binary data follows...]
```

The configurationImport Method

Import a saved configuration snapshot into the server.

Required Permissions: full

HTTP method: POST

Supported by Editions: All

Request Parameters: Because this method contains a file upload, parameters are encoded using the ‘multipart/form-data’ content type.

Parameter	Type	Required	Description
configFile	File	Yes	The configuration snapshot file to import. Content-type for the file must be "application/vnd.zend.serverconfig".
ignoreSystemMismatch	Boolean	No	If set to TRUE, configuration must be applied even if it was exported from a different system (other major PHP version, Zend Server version or operating system). The default value is FALSE.

Expected Response Code: 200 OK

Response Type: serversList with information about affected servers (one server in Zend Server, all cluster members in Zend Server Cluster Manager)

Possible Action Specific Error Codes:

HTTP Code	Error Code	Description
500	importFailed	Importing the configuration snapshot failed.
409	systemMismatch	The system type, PHP version or Zend Server version from which the configuration snapshot was exported does not match the current system. This error can be overridden if the ignoreSystemMismatch parameter is set to TRUE.

Example:

Request (some headers removed for clarity):

```
POST /ZendServerManager/Api/configurationImport
Content-type: multipart/form-data, boundary=--bla-bla-bla--

----bla-bla-bla--
Content-disposition: form-data; name=ignoreSystemMismatch
TRUE
----bla-bla-bla--
Content-disposition: form-data; name=”configFile”;
  filename=”mySavedConfig.zcfg”
Content-type: application/vnd.zend.serverconfig
[...binary data follows...]
----bla-bla-bla----
```

Response:

```
<?xml version="1.0" encoding="UTF-8"?>
<zendServerAPIResponse
  xmlns="http://www.zend.com/server/api/1.0">

  <requestData>
    <apiKeyName>angel.eyes</apiKeyName>
    <method>configurationImport</method>
  </requestData>

  <responseData>
    <serversList>
      <serverInfo>
        <id>12</id>
        <name>www-01</name>
        <address>https://www-01.local:10082/ZendServer</address>
        <status>pendingRestart</status>
        <messageList>
          <warning>This server is waiting a PHP restart</warning>
        </messageList>
      </serverInfo>
      <serverInfo>
        <id>15</id>
        <name>www-02</name>
        <address>https://www-02.local:10082/ZendServer</address>
        <status>pendingRestart</status>
        <messageList>
          <warning>This server is waiting a PHP restart</warning>
        </messageList>
      </serverInfo>
    </serversList>
  </responseData>
</zendServerAPIResponse>
```

Zend Server Best Practices

Introduction

Welcome to the Zend Server 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 .
- The Security section lists several additional security precautions you can take to secure your Zend Server 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

What's in Performance

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

This document includes information on the following performance issues:

- Optimizing Zend Server 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 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 (optional component).
Monitor	Identifies performance issues	Turn off temporarily, only for performance testing reasons. Not recommended to remove this component however it is	

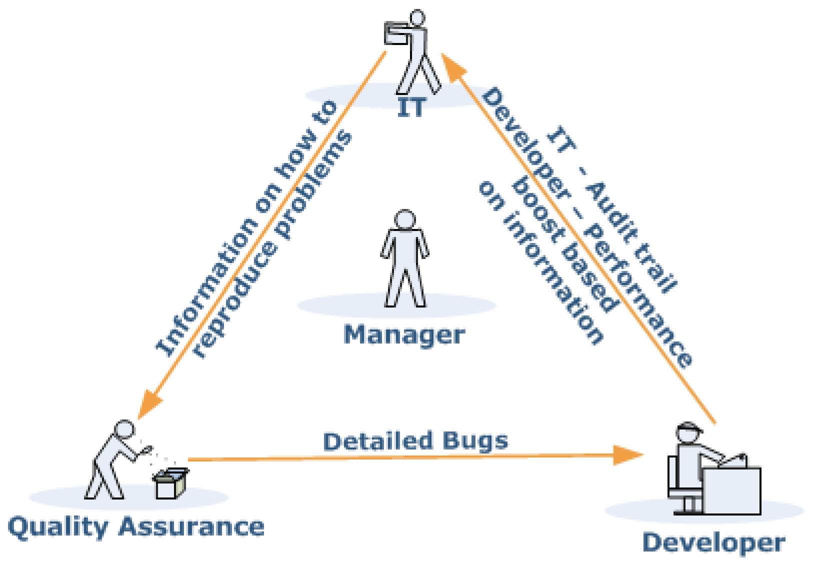
Component	Description	Turn Off	Comment
		best to configure accordingly see " Working with Monitoring"	
Page Cache	A URL based HTML output cache for PHP scripts	Always If you are not using URL based Caching.	If you decide not to use this component.
ZDS (Zend Download Server)	Passing heavy download requests 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 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).


IT
Information on how to reproduce problems
IT - Audit trail
Developer – Performance boost based on information
Manager
Detailed Bugs
Quality Assurance
Developer

Using Zend Server 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 (Restart PHP) 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 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 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.
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 are not using

Name	Recommended Value	Zend Server Default	Description
			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.
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.
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

Name	**Recommended Value**	**Zend Server Default**	**Description**
			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.
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

What's in Security

In the Security section, you will find information on how to configure and optimize the Zend Server 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 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.
- Monitor Security Blacklist - How to hide certain values that are not relevant for diagnostics for security reasons.

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 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 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 .

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:
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.
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

Name	**Default**	**Optimal Value**	**Description**
			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.

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 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 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.

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

Monitor Security Blacklist

When capturing event context, Zend Monitor will save the values of all PHP super-globals (POST, GET, SESSION etc.) in the event database, and will present these values in event reports. Sometimes, it is unwise to store and present some values - for example, passwords or credit card numbers and other private information.

Zend Monitor allows you to filter out some super-global values by black-listing their keys in the Zend Monitor Security Blacklist. Filtered values will not be stored in the *events* database and will not be included in event reports (details) - instead, the string "<BLOCKED_VALUE>" will be presented.

Why Configure Security Settings?

The primary reason for securing information is to prevent the storage, handling and distribution of sensitive information such as user names, passwords and credit card numbers. This information is collected as part of the Zend Server diagnostic process. However, in most cases the context, and not the value, is important to understanding why the Event occurred. Therefore, sensitive information omitted from the Event Detail collection process.

An additional reason for using the Security Blacklist is to prevent inadvertently sending sensitive information by e-mail when using the Event Action option that automatically sends Event Details via email.

Black-listing sensitive information may be required in order to confirm with some security standards such as PCI Certification.

This procedure describes how to manually define a blacklist of keys that should not be collected, stored or displayed in Event reports.

To manually define a security Blacklist:

1. In Zend Server, go to **Server Setup | Components**.
2. Locate the **Zend Monitor** component in the table and click on the **Directives** link.
   This will guide you directly to the directives related to this component in **Server Setup | Directives**.
3. Locate the directive *zend_monitor.security_black_list.*
4. Add as a value any keys that should be blocked using a comma to separate between them.
5. Click "Save Changes"
6. Restart by clicking .

Each of the keys that you added to the list will now be replaced by '<BLOCKED_VALUE>'

Example:

For example, to blacklist HTTP passwords passed to PHP by the web server through the *$_SERVER['PHP_AUTH_PW']* super-global variable, I would add it as follows:

zend_monitor.security_black_list=PHP_AUTH_PW

Any subsequent keys added would be added using a comma to separate them as follows:

zend_monitor.security_black_list=PHP_AUTH_PW,cc_number,secret_token

Please note that if the same key exists in several super-globals (for example, if "cc_number" exists in both $_POST and $_SERVER) it will be removed from all of them. You can configure the list of super-global variables you want to secure using this feature by setting the value of the *zend_monitor.super_globals_to_secure* directive.

Development

What's in Development

In the Development section, you will find information on how to use Zend Server 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 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.
- Advanced Diagnostics with Zend Server - This section presents suggestions to help diagnose problems by event rules.

Working with Zend Framework

Zend Framework users who deploy Zend Server 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,** : <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 to Run a Zend Framework Application

The following procedure describes how to configure Zend Server 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 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 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 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 is installed.
- **Linux**:
  If you installed Zend Server 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 using the generic Tarball package - /usr/local/ zend /apache2/conf/httpd.conf.
If you changed the location of your Zend Server 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 is installed.

Profiling

Profiling code with Zend Server is the process of using Zend Studio to analyze code execution performance.

To access this option from Zend Server go to **Monitor |** ***Events***, select an event from the list and **click** on the event's **ID** number to view the details page.

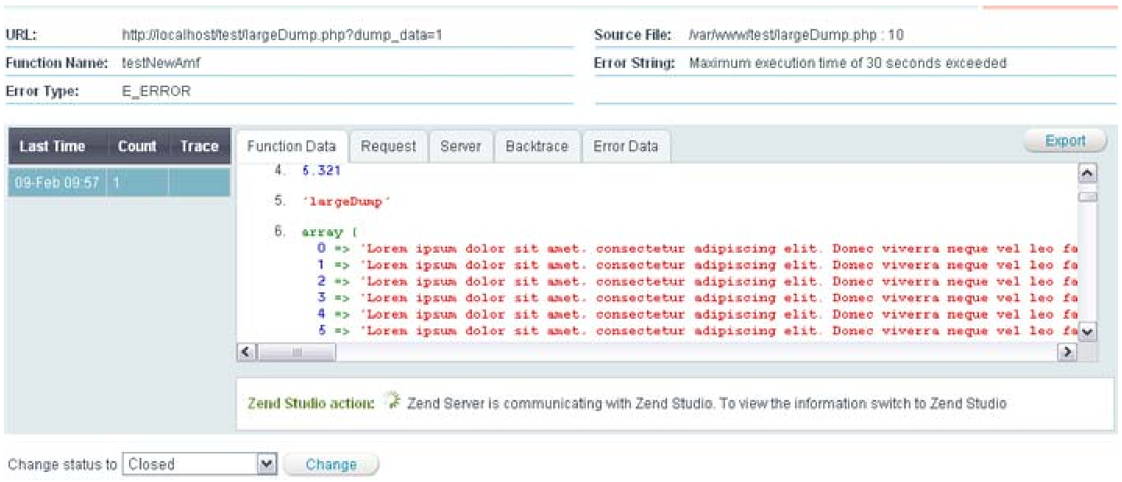
This procedure describes how to profile an event from inside Zend Server. Profiling from inside an event provides an additional diagnostics layer in order to investigate why a specific event was triggered.

Before following this procedure make sure that Zend Server and Zend Studio are running and properly setup.

To Profile an event:

1. Open an event by going to **Monitor | Events**, selecting an event from the list and **clicking** on the event's **ID** number to view the details page.
2. In the Zend Studio diagnostics area click Profile Event

   This action will run on the server defined in ▼ Settings. By default, the settings are set to run diagnostic actions on the originating server (the server on which the event was created). You can change the settings to run on a different server.



URL:	http://localhost/test/largeDump.php?dump_data=1	Source File:	/var/www/test/largeDump.php : 10
Function Name:	testNewAmf	Error String:	Maximum execution time of 30 seconds exceeded
Error Type:	E_ERROR		

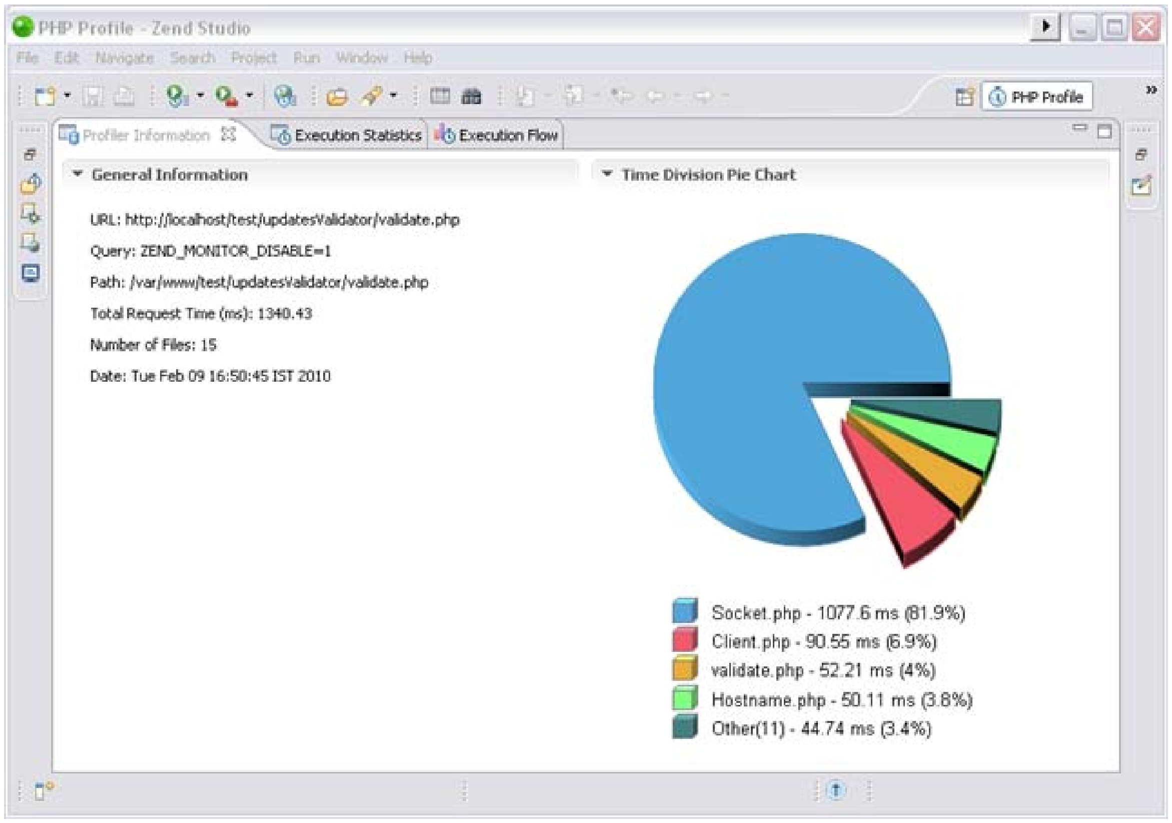
Last Time	Count	Trace
09-Feb 09:57	1	

```
4.  6.321

5.  'largeDump'

6.  array (
      0 => 'Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec viverra neque vel leo fa
      1 => 'Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec viverra neque vel leo fa
      2 => 'Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec viverra neque vel leo fa
      3 => 'Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec viverra neque vel leo fa
      4 => 'Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec viverra neque vel leo fa
      5 => 'Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec viverra neque vel leo fa
```

Zend Studio action: Zend Server is communicating with Zend Studio. To view the information switch to Zend Studio

Change status to: Closed

3. The information will be transferred to the Zend Studio PHP Profile preference where you can run profile and edit the file.



URL: http://localhost/test/updatesValidator/validate.php

Query: ZEND_MONITOR_DISABLE=1

Path: /var/www/test/updatesValidator/validate.php

Total Request Time (ms): 1340.43

Number of Files: 15

Date: Tue Feb 09 16:50:45 IST 2010

🟦 Socket.php - 1077.6 ms (81.9%)
🟥 Client.php - 90.55 ms (6.9%)
🟧 validate.php - 52.21 ms (4%)
🟩 Hostname.php - 50.11 ms (3.8%)
⬛ Other(11) - 44.74 ms (3.4%)

If this did not work see if one of these troubleshoot options help

Advanced Diagnostics with Zend Server

The information contained in an Issue (**Monitor |** ***Events***) is geared towards analyzing and resolving all sorts of problems that are common to running Web Applications in PHP.

Based on the type of rule on which triggered an issue you can immediately begin to start solving the issue.
The first step is to make sure that the issue was genuinely generated.
To do this consider one of the following:

1. Is this a real error or should the Monitor Rule parameters be modified (thresholds, functions list, etc.)?
2. Can I use the monitor API to solve this problem (i.e. identify the problem as a known issue to be ignored, so that no additional events will be added to the issue)?
3. Is the detected behavior accepted behavior for the specific situation (time, script, load etc.) that should be ignored.
4. Use the Administration Interface to manage issues by changing the Status to **Ignored**. All events that happen and are aggregated to that issue will still be monitored but the issue will stay ignored.
5. Use URL Tracing to collect further information.

Once you have established that the issue represents a real problem, you can start to handle the issue.
Use the links below to drill down by type of rule for suggestions that can help diagnose problems.
Rule names in this page are in their basic form without the severity or reference to absolute and relative.

Click on a Rule name (Link) to view diagnostics information by rule: Custom Event | Slow Function Execution | Function Error | Slow Query Execution | Slow Request Execution | High Memory Usage | Inconsistent Output Size | PHP Error | Java Exception | Database Error.

Event Rules

Custom Event

***Description**: When the API function 'zend_monitor_custom_event' is called from inside PHP code, it generates an event. When enabled an event will be generated and displayed in the Monitor* | *Events.*

Information Collected:

The most important details are:

- Function Name - As displayed in the Issue's General Details
- Function Arguments - The arguments of the function that triggered the event are listed in the Function Data tab.
- user_data - if specified in the function call that triggered the event it will display additional user defined data.

In most cases, these details alone should be enough to indicate what happened to trigger an event.

Applicable Diagnostic Actions:

Click on a link to see how to perform each action. Tools are listed in order of relevance to helping solve the event:

1. Run the Debugger

Possible Causes and Solutions:

Custom Events are defined by users according to specific requirements. These events can only be triggered by specific code in a specific application and therefore it is only possible to say that the application's logic triggered the event.
There are certain circumstances where applying a Custom Event can be useful:

1. When handling logging routines and exceptions by adding a call to '*zend_monitor_custom_event*' with the data that has been logged or held in the exception.
2. In logical closure situations. For example when handling inputs that require monitoring the value of the input (for different actions) to prevent inputting values that are not acceptable for the business logic. Adding the function '*zend_monitor_custom_event*' will give you the ability to trigger an event when an unacceptable value is passed and also provide the necessary backtrace information to understand how the value got there.
3. More...

Slow Function Execution

***Description**: When a specific function in your code runs slowly, Zend Monitor identifies it as an event worth reporting. The "Slow Function Execution" Rule contains the following monitoring definitions, runtime duration and a list of functions that should be monitored.*

Information Collected:

The most important details are:

- Function Name - As displayed in the Issue's General Details
- Function Arguments - The arguments of the function that triggered the event are listed in the Function Data tab.

In most cases, these details alone should be enough to indicate what happened to trigger an event.

Applicable Diagnostic Actions:

Click on a link to see how to perform each action tools are listed in order of relevance to helping solve the event:

- Run the Profiler
- Open code in editor
- Run the Debugger
- Use URL Tracing to collect further information about this function in action.

Possible Causes and Solutions:

Queries to a DB (database) - long, elaborate and complicated DB queries may take a long time and make the function appear to take a long time to execute.
There are many ways to speed up DB queries such as:

- Revise the SQL query itself - make it simpler
- Improve the structure of your DB tables
- Use RDBMS features that can improve speed such as indexes, prepared statements, stored procedures etc…
  All these are only suggested possible causes and each event. Developers have to analyze each occurrence to understand the specific reasons behind the slow execution time.

Long running actions - Some actions triggered by a function can, by definition take a long time. Examples of long running actions can be using a function to run code from the command line or remote access queries with Web services or searching for files in a directory. In most cases, these uses of a function cannot be refined and the best action is to ignore these issues when they occur.
False Positives - Sometimes functions run slowly. Not all functions that run slowly are indicative of a problem in your code or environment and they may be no indication of unacceptable behavior. If this is the case, remove the function from the Rule's "Watched Functions" list or set issues triggered by this function to ignored.

Function Error

***Description**: When a specific function in your code returns FALSE, it generates an event. The "Function Error" Rule contains a list of monitored functions (i.e. functions that when returning FALSE will trigger an event).*

Information Collected:

The most important details are:

- Function Name - As displayed in the Issue's General Details
- Function Arguments - The arguments of the function that triggered the event are listed in the Function Data tab.
- Backtrace – identify what happened before the error happened that may have caused a problem such as incorrect data or problematic input data.

In most cases, these details alone should be enough to indicate what triggered the event.

Applicable Diagnostic Actions:

Click on a link to see how to perform each action tools are listed in order of relevance to helping solve the event:

- Run the Debugger
- Open code in editor
- Redefine database queries
- View information in the Logs
- Run the Profiler

Possible Causes and Solutions:

Generally, logic errors trigger Function Error *events* such as queries that you expect to return something and they do not and not PHP errors.
If you have a function that you need to return FALSE as an acceptable value, remove the function from the monitored functions list.

- Internal PHP functions – If you are using internal PHP functions, the best reference for investigating the problem is use the PHP manual (The Zend Controller's Search option provides quick access to searching the PHP Manual.
- You can also check your logs to see if they show PHP error information logged the same time the function error occurred.
- User define functions – If it is a user defined function, running the debugger will help find out if the function is running complicated actions that are causing the function to fail. Using Breakpoints while debugging, will further pinpoint the problematic area in the code.
- False Positives - Sometimes functions are supposed to return FALSE. Not all functions that return FALSE are indicative of a problem in your code or environment and they may be no indication of unacceptable behavior. If this is the case, remove the function from the Rule's "Event Condition" list or set the status of issues triggered by this function to ignored.

Note:
Removing a function from the rule, affects all instances of the function. Before removing the function from the list, make sure the function does not require monitoring wherever it is used.

Slow Query Execution

Description*: When a specific query runs longer then a specified duration it generates an event. The "Slow Query Execution" Rules contains a list of functions that trigger events either severe or normal.*

Information Collected:

The most important details are:

- URL - As displayed in the Issue's General Details
- Request data - Listed in the Request.

In most cases, these details alone should be enough to indicate what happened to trigger an event.

Applicable Diagnostic Actions:

Click on a link to see how to perform each action tools are listed in order of relevance to helping solve the event:

- Run the Profiler
- Open code in editor
- Run the Debugger
- View information in the Logs
- Run general diagnostics on the machine's performance in terms of memory and CPU usage.
- Use URL Tracing to collect further information about this function in action.

Possible Causes and Solutions:

Slow queries in general indicate that there is a performance problem and they generally appear when there is a heavy load, which in turn causes Web applications to perform poorly.

Check the following:

Bottlenecks caused by un-optimized queries or multiple queries, such as un-joined queries, slow queries and multiple short queries. Use the Profiler to see how many queries are running at the same time or set thresholds to find queries that take a long time to run.

Check to see if a different CPU intensive process was running in the background taking up the CPU and making everything run slowly.

Look at the amount of calls to the function - are there too many? add breakpoints and run again to manually trace per query what happened. Possible Solution: cache the information using a Data Cache API as a PHP array or use the Page Cache to cache the presentation layer.

Profiler results - analyze the profiler results and isolate functions/areas that consume the majority of the time and analyze each function/area code separately to isolate the possible cause of the problem.

Slow Request Execution

Description: *When a specific request runs longer then a specified duration it generates an event. The "Slow Request Execution" Rules contains the durations that trigger events either severe or normal and either relative to a specific value (absolute) or to a percentage (relative measurement per URL).*

Information Collected:

The most important details are:

- URL - As displayed in the Issue's General Details
- Request data - Listed in the Request.

In most cases, these details alone should be enough to indicate what happened to trigger an event.

Applicable Diagnostic Actions:

Click on a link to see how to perform each action tools are listed in order of relevance to helping solve the event:

- Run the Profiler
- Open code in editor
- Run the Debugger
- View information in the Logs
- Run general diagnostics on the machine's performance in terms of memory and CPU usage.
- Use URL Tracing to collect further information about this function in action.

Possible Causes and Solutions:

Slow requests in general indicate that there is a performance problem and they generally appear when there is a heavy load, which in turn causes Web applications to perform poorly.

Check the following:
Bottlenecks caused by un-optimized queries or multiple queries, such as un-joined queries, slow queries and multiple short queries. Use the Profiler to see how many queries are running at the same time or set thresholds to find queries that take a long time to run.
Check to see if a different CPU intensive process was running in the background taking up the CPU and making everything run slowly.
Look at the amount of calls to the function - are there too many? add breakpoints and run again to manually trace per query what happened. Possible Solution: cache the information using a Data Cache API as a PHP array or use the Page Cache to cache the presentation layer.
Profiler results - analyze the profiler results and isolate functions/areas that consume the majority of the time and analyze each function/area code separately to isolate the possible cause of the problem.

High Memory Usage

***Description**: When a specific PHP request consumes more than the specified amount of memory it generates an event. The “High Memory Usage” Rules contain the memory consumption setting (i.e. the amount of memory the request has to consume to trigger an event).*

Information Collected:

The number of occurrences is the best indicator. A single occurrence can be disregarded (change the Status of the Issue to Closed) only if it occurs multiple times it is worth investigating. Closed *events* that continue to receive new activity open automatically.

Applicable Diagnostic Actions:

Click on a link to see how to perform each action tools are listed in order of relevance to helping solve the event:

- Use URL Tracing to collect further information about memory consumption.
- Open code in editor to see if the script can be improved
- View information in the Logs

Possible Causes and Solutions:

- Database Functions - Some queries may return large record sets and should be refined.
- Iterative functions – functions that include many ‘foreach’ loops may cause excessive memory usage and should be reviewed to see if less memory can be consumed.
- False Positives - Sometimes requests consume large amounts of memory. Not all requests that consume lots of memory are indicative of a problem in your code or environment and they may be no indication of unacceptable behavior. If this is the case, set issues triggered by this function to ignored.

Inconsistent Output Size

***Description**: When a specific PHP request's output size deviates from the average by the percentage specified (measured per URL) it generates an event. The "Inconsistent Output Size" Rule contains the output size's deviation percentage (i.e. the amount of output the request has to generate in order to trigger an event).*

Information Collected:

The output size in the Group Details helps to analyze the nature of the event. For example if the output size is 0 you can determine that nothing was outputted and try to understand why.
Generally, PHP errors and Function errors are generated at the same time as the Inconsistent Output Size error, which can provide further information.

Applicable Diagnostic Actions:

Click on a link to see how to perform each action tools are listed in order of relevance to helping solve the event:

- Run the Debugger
- View information in the Logs
- Open code in editor
- Run the Profiler

Possible Causes and Solutions:

Small Output Sizes - It is important to look at the results that show a smaller output size to help identify why this output was so large. This usually indicates that the requested output was not fully generated.
Possible Solution - Look for related PHP errors and Function Errors and then view the PHP and Web server logs for further details.

PHP Error

Description*: When a specific PHP error is reported, it generates an event. The "PHP Error" Rule contains a list of monitored PHP error types. This event type complements the error_reporting settings in your php.ini by reporting specific errors even if they are set to disabled in your php.ini..*

Information Collected:

The most important details are:

- Function Name - As displayed in the Issue's General Details
- Error Data - Listed in the Error Data tab.
- Backtrace – to investigate what function calls were executed just before the error was reported.
- In most cases, these details alone should be enough to indicate what happened to trigger an event.

Applicable Diagnostic Actions:

Click on a link to see how to perform each action tools are listed in order of relevance to helping solve the event:

- Run the Debugger
- Open code in editor
- Redefine database queries
- View information in the Logs

Possible Causes and Solutions:

Syntax/Parse Errors - missing or incorrect syntax in code that is found during PHP compilation
Fatal Runtime Errors - such as E_WARNING and E_ERROR - indicate that there was a call to an undefined function or that you did not load a specific extension or when classes and Functions are defined twice. Possible Solution: Open code and view Line and function that triggered the error.
Uncaught Exceptions - generate fatal errors, with a complete backtrace to trace the reason why the PHP error was reported.
Runtime Warnings - The code did not run as expected. Normally a notice is displayed and the code continues to run in an unexpected manner or the code will crash. Possible solutions: check your code for the following wrong DB QUERIES missing FILES, STREAMS functions that are performing DIV BY ZERO

Java Exception

***Description**: When Java code called through the Java Bridge fails due to an uncaught Java exception, it generates an error. The "Uncaught Java exception" Rule determines if Uncaught Java Exceptions are reported or not.*

Information Collected:

The most important details are:

- Function Name - As displayed in the Issue's General Details
- Error Data - Listed in the Error Data tab including the Java Stack and Java error string.

In most cases, these details alone should be enough to indicate what happened to trigger an event.

Applicable Diagnostic Actions:

Click on a link to see how to perform each action tools are listed in order of relevance to helping solve the event:

- Open code in editor
- Run the Debugger
- Run the Profiler
- Redefine database queries
- View information in the Logs

Possible Causes and Solutions:

Based on the exception type, investigate the Java code and fix to stop generating the event – all relevant information should be collected in the issue.
False Positives - Sometimes developers write code to intentionally trigger an Uncaught Java Exception. Not all triggered exceptions are indicative of a problem in your code or environment and they may be no indication of unacceptable behavior. If this is the case, set issues triggered by this function to ignored.

Database Error

***Description**: When a specific watched database function returns FALSE, it generates an event. The "Database Error" Rule contains a list of monitored functions (i.e. functions that when returning FALSE will trigger an event).*

Information Collected:

The most important details are:

- Function Data - Listed in the Function Data tab.
- Function Name - As displayed in the Issue's General Details
- The error type will show if it is an SQL query error, shell code error, Java code error

Applicable Diagnostic Actions:

Click on a link to see how to perform each action tools are listed in order of relevance to helping solve the event:

- Open code in editor
- Run the Debugger
- View information in the Logs

Possible Causes and Solutions:

Possible causes:
Check that the connection to the DB is working. Possible Solution: verify the connection data is correct (address, user name, passwords etc.) and manually re-establish the connection.
External code problems such as malformed non-PHP code (code that can trigger a PHP error). Solution: Fix SQL code.
Function errors should to help understand the problem according to the content of the error.
Use the debugger to find the code/query that failed.
Incorrect database queries can trigger the event. Solution: redefine database queries and verify that the correct query syntax is used.

Deployment to Production

What's in Deployment to Production

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

This document includes information on:

- Deploying Code with Zend Server - This section presents suggestions on how to best deploy your PHP code to run with Zend Server 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 developer's 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

Where to Put the Code?

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

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 it's 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 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

Whats in 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 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 .

By default, Zend Server 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 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 .

By default, Zend Server 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

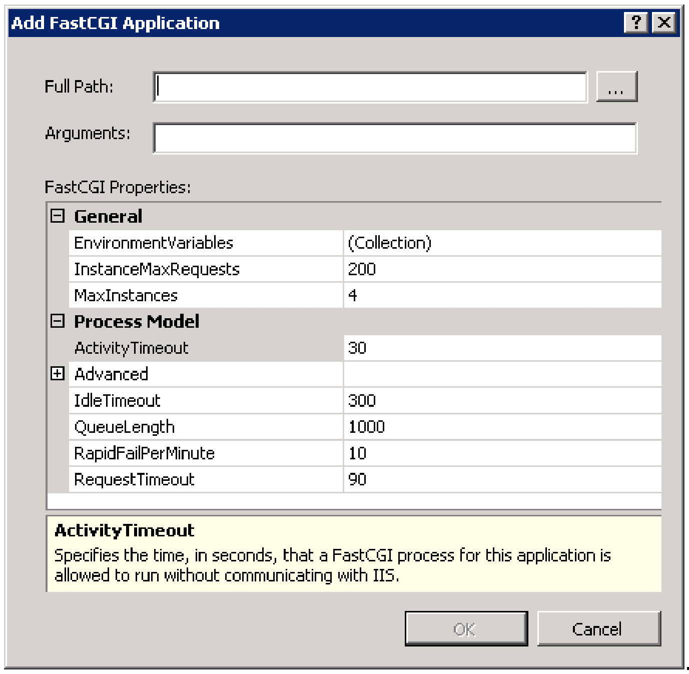
Once installed, you can administer your FastCGi settings from the Internet Information Services (IIS) Manager.

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 FastCGI Settings 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:



General	
EnvironmentVariables	(Collection)
InstanceMaxRequests	200
MaxInstances	4
Process Model	
ActivityTimeout	30
Advanced	
IdleTimeout	300
QueueLength	1000
RapidFailPerMinute	10
RequestTimeout	90

ActivityTimeout
Specifies the time, in seconds, that a FastCGI process for this application is allowed to run without communicating with IIS.

.

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 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>" ***requestTimeou***t="<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 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

What's in the Troubleshoot

Welcome to the Zend Server 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.

Cant find what you are looking for? We want to know!
Send a mail to documentation@zend.com asking about an error message or a usability issue and we will make a troubleshoot item and add it here.

This document includes information on the following issues:

All operating systems

- Zend Server Exception Caught - When the port settings are not configured as expected
- License Not Working - Your new license does not activate the features
- Zend Controller Cannot Login - Zend Controller does not start as expected
- Zend Controller Cannot Run Benchmark - There is an issue with the URL you are trying to test
- Error: Failed to Communicate with Zend Studio - The communication with Zend Studio has failed
- Changing the Component's Log Directory - Configuration options for advanced users
- Log File Permissions - handle connection permission errors to Apache logs

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.

License Not Working

This issue is relevant for all operating systems.

Problem: While running Zend Server in Community Edition, I enter a new license and nothing happens.

Expected Result: Entering a valid license should reactivate Zend monitor, Zend Page Cache and Zend Download Server.

Solution: Click Restart Server to make sure the license change is applied.

Still doesn't Help: Try to manually Restart your PHP from the command line or go to the Zend Support Center - http://www.zend.com/en/support-center/ for information about our support options.

Support Tool

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 on a supported browser from the Supported Browser List.

Supported Browser List

The following table lists the browsers that run Zend Server .

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 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 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 to read the Apache error log on Debian Linux is provided below:

To enable Zend Server 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

Zend Server Exception Caught

Installing Zend Server 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 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:

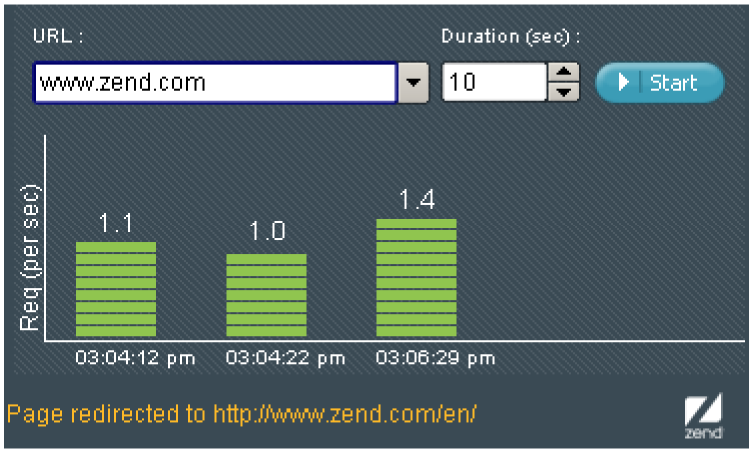
1. Change the Administration Interface's port setting as follows:
   Go to <install_path>/gui/application/data /usr/local/zendsvr/ httpd/conf/httpd.conf
2. Open the file zend-server.ini. In the section called "userServer", set the URL to the new port number. Change the port number for directive “Listen”.
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
- IBM i Apache conf file: usr/local/zendsvr/apache2/conf/httpd.conf

Zend Controller Cannot Run Benchmark

The following message may appear after you enter a URL into the Zend Controller's benchmark: "Page redirected to ..."



Page redirected to http://www.zend.com/en/

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 you try to run the Zend Controller and a message is displayed in the Zend Controller stating that it cannot log in.

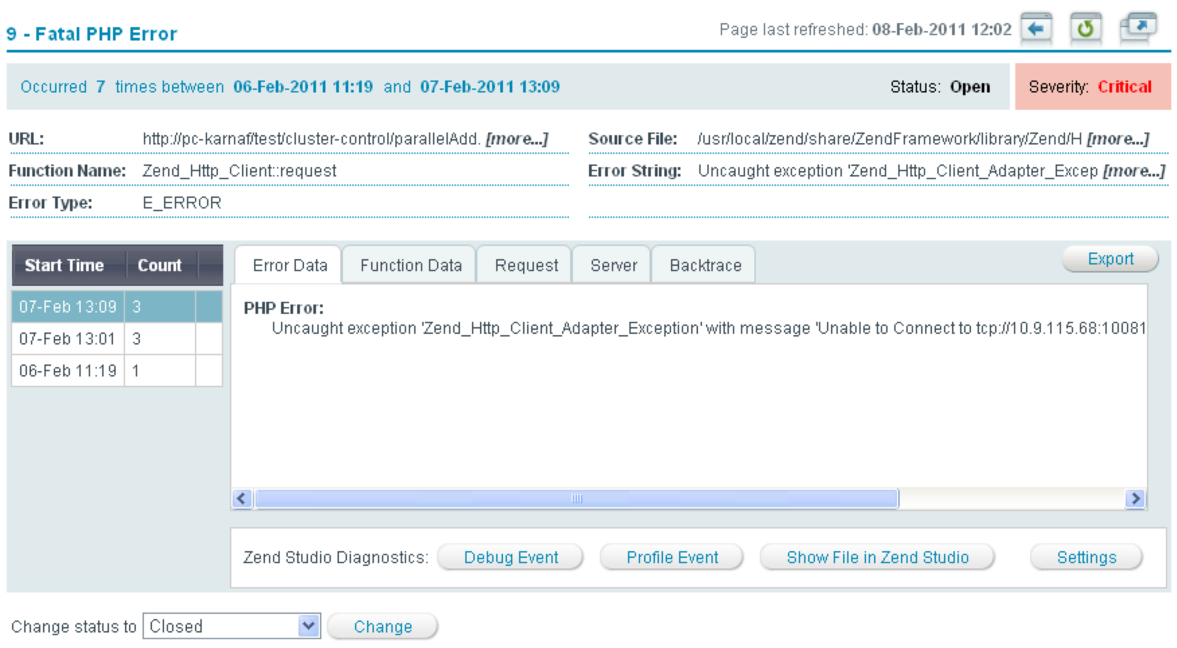
Possible causes:

1. You have not yet logged in to Zend Server for the first time and therefore your password has not been defined.
   Log in to Zend Server 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 .

Error: Failed to Communicate with Zend Studio

The following error message appears in Zend Server when using the Zend Studio Diagnostics that are available from the **Monitor |** ***Events*** **| Event Details** page.

Failed to communicate with Zend Studio. Go to the online help's 'troubleshoot' section to find out how to fix the connection



9 - Fatal PHP Error

Page last refreshed: 08-Feb-2011 12:02

Occurred 7 times between 06-Feb-2011 11:19 and 07-Feb-2011 13:09

Status: Open

Severity: Critical

URL: http://pc-karnaf/test/cluster-control/parallelAdd. *[more...]*

Function Name: Zend_Http_Client::request

Error Type: E_ERROR

Source File: /usr/local/zend/share/ZendFramework/library/Zend/H *[more...]*

Error String: Uncaught exception 'Zend_Http_Client_Adapter_Excep *[more...]*

Start Time	Count
07-Feb 13:09	3
07-Feb 13:01	3
06-Feb 11:19	1

PHP Error:
Uncaught exception 'Zend_Http_Client_Adapter_Exception' with message 'Unable to Connect to tcp://10.9.115.68:10081

Change status to Closed

This error message can be caused by a several possible problems:

When running diagnostics on an alternate server:

1. The Zend Debugger is not running on the alternate server.
   Solution - Make sure that the Zend Debugger is running and available on the alternate server by going to the Zend Server Administration Interface and in **Server Setup | Components** check that the Zend Debugger is turned on.
2. The connection parameters in **Server Setup | Monitor** are not the same as the settings in Zend Studio's Debugger preferences. (IP address, Port and if you are using SSL).
   Solution - Check the settings in Zend Studio for Eclipse. For instructions go to: http://files.zend.com/help/Zend-Studio-Eclipse/zend-studio-eclipse.htm and make sure the Zend Studio for Eclipse debug settings are the same as defined in Zend Server .
3. The Zend Studio IP address is not allowed to debug on the alternate server.
   Solution - Go to your Administration Interface and make sure that the Zend Studio IP address that appears in **Server Setup | Monitor** is an allowed host to debug - the setting should be in the alternate server's Zend Server Administration Interface under **Server Setup | Debugger**.

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

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

After installing Zend Server, 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 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 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).

Windows: Zend Server not Loading

This Item is only relevant for Windows.

If for any reason, you cannot load Zend Server 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:

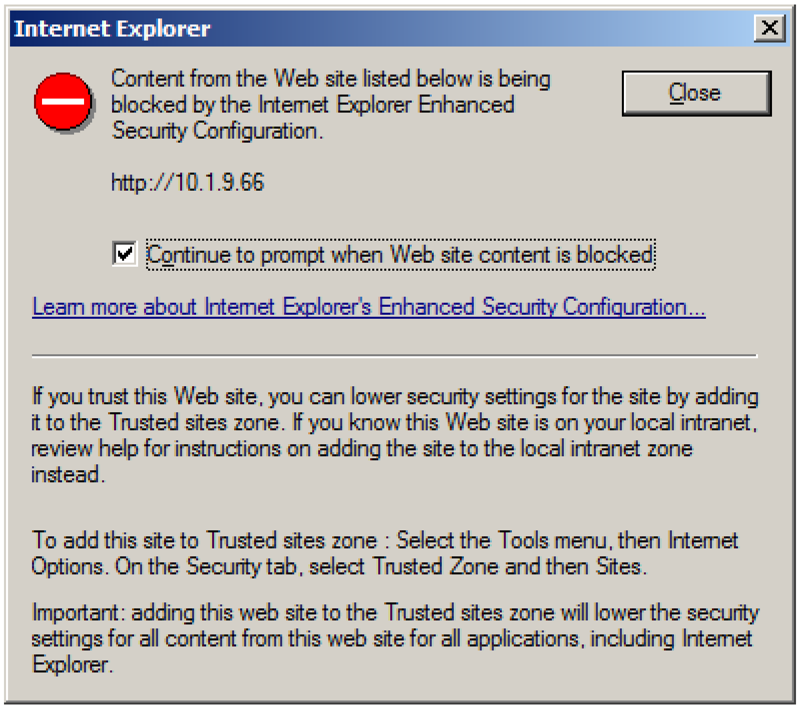
1. Run the installer file or go to Start | Control Panel | Add or Remove Programs | Zend Server and select Modify to run the installer
2. Click next to complete the repair process and Finish to close the Installer

You should now be able to run Zend Server. 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 for the first time, you may encounter an Internet Explorer system message stating that Zend Server was blocked (see image below).



Content from the Web site listed below is being blocked by the Internet Explorer Enhanced Security Configuration.

http://10.1.9.66

☑ Continue to prompt when Web site content is blocked

Learn more about Internet Explorer's Enhanced Security Configuration...

If you trust this Web site, you can lower security settings for the site by adding it to the Trusted sites zone. If you know this Web site is on your local intranet, review help for instructions on adding the site to the local intranet zone instead.

To add this site to Trusted sites zone : Select the Tools menu, then Internet Options. On the Security tab, select Trusted Zone and then Sites.

Important: adding this web site to the Trusted sites zone will lower the security settings for all content from this web site for all applications, including Internet Explorer.

This is a security message prompting you to add Zend Server to the trusted sites zone.

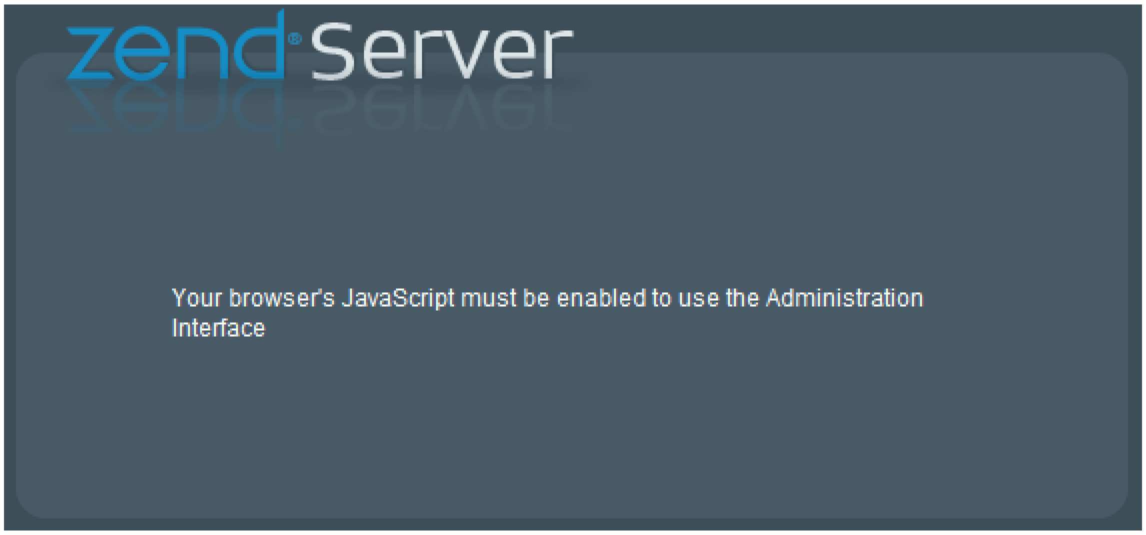
This procedure describes how to add Zend Server to the trusted sites zone in Internet Explorer 7 running on Windows 2008 Server.

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 as a trusted site.
5. Click **Close** and then **OK** to save the changes and close the dialog.

Zend Server 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:


Your browser's JavaScript must be enabled to use the Administration Interface

This also indicates that Zend Server 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.