Zend Platform™

User Guide

Zend Platform V3.0 for i5/OS

By **Zend Technologies, Inc.**

www.zend.com

Zend The php Company

Zend Platform User Guide Disclaimer

The information in this document is subject to change without notice and does not represent a commitment on the part of Zend Technologies Ltd. No part of this manual may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or information storage and retrieval systems, for any purpose other than the purchaser's personal use, without the written permission of Zend Technologies Ltd.

All trademarks mentioned in this document, belong to their respective owners.

© 1999-2007 Zend Technologies Ltd. All rights reserved.

Zend Platform User Guide for i5/OS issued November 2007.

DN: ZP-I5UG-191107-3.0-005
Product Version: 3.0.3a for i5

Preface

Zend Platform is an extremely diverse, runtime-environment management platform. As such, a greater understanding of the underlining concepts is required to benefit from the Zend Platform capabilities and features. This User Guide reflects these concepts by providing a workflow driven description of Zend Platform's features.

Part One: Introduction to Zend Platform is an introduction describing the background and architectural design of Zend Platform along with who should read this guide and how to maximize the benefits of deploying Zend Platform in your environment.

Part Two: Administration and Configuration, describes the initial tasks that should be done in order to customize Zend Platform to any given environment.

Part Three: Performance Management Server, describes the features and functionality included with the "Performance Management Server". This part includes functional descriptions of the performance management components, which include: Platform (Management Console), PHP Intelligence, Performance and Configuration.

Part Four: Enterprise Server, describes the features and functionality included with the "Enterprise Server". This part includes functional descriptions of Enterprise grade components, which include: Job Queues and SNMP Traps.

Part Five: Integration Server, describes the features and functionality included with the "Integration Server". This part includes functional descriptions of Integration components, which include: Zend Platform's Java Bridge and advanced reporting facilitated by Actuate's BIRT Reporting Tool.

Part Six: Reference Information, provides additional reference information. This part includes an API and Directives list, Tutorials and Appendixes.

Audience

Zend Platform is responsible for providing solutions to the challenges faced by different stakeholders in the organization; therefore, this guide is suited for Managers, System Administrators and Developers.

Managers will learn how to utilize this solution to support PHP lifecycle management by streamlining the PHP application lifecycle across development and production.
This kind of development method is responsible for shortening release cycles by integrating the working environment.

Managers can benefit from knowing how to utilize Zend Platform with their testing staff to obtain a means for detecting and pinpointing run-time problems throughout the production lifecycle while having the safety of knowing that when run-time problems are located, testing staff will have a complete audit trail to help resolve the issue.

From a usability aspect, delivering a product on time is only part of the equation. Find out how to improve user experience and increase performance, up-time and customer satisfaction through using Zend Platform.
From the PHP aspect, know why Zend Platform considerably improves PHP execution while maintaining and synchronizing PHP configurations.

From an organizational standpoint, find out how to leverage existing investments in Java applications as well as reducing costs on hardware.

System Administrators will be able to understand how to detect problems with PHP Intelligence to pinpoint run-time problems with detailed information. Learn how to control configuration by synchronizing configuration of '.ini' files accurately across PHP servers and improve performance by getting more out of each PHP server (less servers doing the same job).

Developers can learn how to integrate with Zend Studio to improve quality by quickly identifying problems and reducing the testing cycle. They can also benefit from the PHP/Java Integration Bridge to reuse code by utilizing existing code.

Table of Contents

Part I: Introduction to Zend Platform **8**

About Zend Platform for i5/OS **9**

Navigation 9

Central Control Center 9

Standard and Enterprise Servers 10

Zend Platform Overview **11**

Environments 11

Architecture 13

Part II: Administration and Configuration **18**

Cluster Management **19**

Server Management 20

Group Management 21

VHost Management 21

Restricting Access to Virtual Hosts 21

PHP Intelligence **22**

Configuring Event Triggers 22

Why Configure Event Triggers? 23

Filtering Event Triggers 23

Defining Event Triggers 24

Choosing and Defining Event Triggers 25

Define Event Actions 33

Performance **37**

Configuring Performance 37

File Compression 49

Blacklists 50

Configuring the Zend Download Server (ZDS) 51

Configuration Tab **56**

Tunneling (Communication Settings) 56

Studio Settings 61

PHP Configuration 63

Users and Groups **66**

User Management 66

Adding and Editing Users 67

Adding and Editing Groups 68

Passwords 70

Licenses **71**

License Management 71

Part III: Performance Management Server **74**

The Problem Resolution Lifecycle **75**

The Problem Resolution Lifecycle 75

Implementing the Problem Resolution Lifecycle 76

Creating Events 77

Finding Events that Interest You 78
PHP Intelligence 79
System Health 79
Event List 81
Event Details 84
Database Maintenance 93
Graphs 94
Performance 96
Overview 96
Performance Lifecycle 98
Implementing the Performance Lifecycle 100
Event Trigger Settings and Analysis 104
Performance Optimization Tools 106
Tuning 111
Accelerator Performance Level Descriptions 112
Tuning Zend Platform for Optimal performance on i5 OS 113
Web Services 114
Introduction 114
System Requirements for Web Services 114
General Tasks 115
Get/Set Actions 116
Add/Remove Servers Actions 117
Event Handling 118

Part IV: Enterprise Server 120
Job Queues 121
Introduction 121
Job Queues 122
Job Queue Tab (Job Management) 124
Queues 124
Jobs 127
Job Queue Settings 132
Creating Jobs 134
Zend Download Server (ZDS) 135
Configuring the Zend Download Server (ZDS) 135

Part V: Integration Server 143
Java Bridge 144
About Zend's Java Bridge Technology 144
Java Status Page 147
Common Tasks 150
Usability Issues 155
BIRT Reports 158
About BIRT Reports 158
The BIRT Reports Tab 158
Setting-Up the BIRT Report Engine 159
Zend Platform BIRT Report Examples 159

Part VI: Reference Information **165**

Zend Platform APIs and Directives **165**

APIs and Directives 177

Zend Platform Built-In Services and Extensions **185**

About 185

Setup Tool 185

Services 186

Extensions 190

Tutorials **192**

Integrating Existing and Legacy Applications 192

Calling an EJB on Websphere from PHP 195

Partial and Preemptive Page Caching 197

Appendixes **204**

Appendix A – Troubleshooting Zend Platform 204

Appendix B – Configuration Check List 206

Appendix C – Performance Lifecycle Check List 206

Appendix D - Event Aggregation Mechanism 207

Appendix E – Zend Platform Support 209

Appendix F – zend.ini Configuration Settings 211

Appendix G - Network Port Requirements 213

Appendix H - About SNMP 215

Index **221**

PART I: INTRODUCTION TO ZEND PLATFORM

IN THIS CHAPTER…

NAVIGATION
CENTRAL CONTROL CENTER
STANDARD AND ENTERPRISE SERVERS
OVERVIEW
ENVIRONMENTS
ARCHITECTURE
CENTRAL SERVER
NODES
CENTRAL-NODE COMMUNICATION
PLATFORM ADMINISTRATION A SINGLE POINT OF ACCESS

Zend Platform is a complete runtime environment for managing and maintaining mission critical and enterprise PHP applications from a single, centralized location.

This environment consists of cluster management; performance management, monitoring, detection and recovery; and Java integration.

Zend Platform improves both the end user experience and IT productivity by combining cluster and performance management; automated monitoring and detection capabilities; and powerful Java Integration capabilities into one integrated environment.

Zend Platform provides the PHP-enabled enterprise with the ability to:

- Manage every aspect of PHP from a single, Web-based interface
- Quickly drill-down to critical issues to resolve and optimize
- Create user defined thresholds and error values
- Configure servers from a remote management station and to perform controls at a click of a button
- Monitor performance improvement with Code Acceleration, Content Caching and File Compression
- The Zend Download Server
- Integrate with Java system elements over Platform’s fully implemented PHP/Java Bridge.

About Zend Platform for i5/OS

Zend Platform is the only robust PHP production environment that ensures your applications run smoothly at all times.

Designed for IT personnel and businesses that require industrial-strength applications in highly reliable production environments, Zend Platform offers high performance and scalability to provide your customers with the best possible Web experience and response time.

Zend Platform uniquely guarantees application up time and reliability through enhanced PHP monitoring and immediate problem resolution that removes the troubleshooting guesswork out of the equation and replaces it with peace-of-mind.

You spent time and money developing your state-of-the-art PHP application, now it is time to ensure its up and running.

Navigation

Zend Platform is a browser-based application. The general layout of functionality is in a tabbed view where each tab represents a unique functionality.

- **Platform** - Management functionality
- **PHP Intelligence** - Monitoring and event generation capabilities
- **Performance** - Performance enhancement tools
- **Job Queues** - Streamline offline processing
- **Integration** - Incorporate a Java environment to enrich your applications

The tab colors indicate the server type that is determined by the license type you have. The pale blue tabs belong to the Performance Management Server and the darker blue tabs belong to the Enterprise and Integration servers.

Central Control Center

Zend Platform handles clusters and standalone servers. As such, users can navigate freely between the central server and nodes.

Users stay on the central server until they select a tab that prompts to select a server; as soon as a server is selected, subsequent actions and settings will be applied to the selected server only.

A status bar showing the date, time and login name will also display the name of the server on which the user is currently working.

When no server name appears, you are on the central server.

Standard and Enterprise Servers

Zend Platform is distributed as either a Performance Management Server or an Enterprise Server. Choosing the appropriate server depends on your organization's requirements.

The following table lists the different servers and their respective functionality:

Feature	**Performance Management**	**Enterprise**	**Comments**
Platform	✓	✓	This includes the Dashboard, Server Status indicator, User and license Management.
PHP Intelligence	✓	✓	This includes the System Health overview, Event management and the Graph generator.
Performance	✓	✓	This includes performance management features: Code Acceleration, Dynamic Content Caching, File Compression, Updating Virtual Hosts and testing URLs.
Configuration	✓	✓	This includes the advanced configuration features for configuring your PHP directly from Platform Administration and enabling the connectivity with Zend Studio to provide a complete development lifecycle.
Job Queues		✓	Improve response time during interactive web sessions and utilizing unused resources.
Integration		✓	This includes Java Bridge connectivity and integration with business intelligence reporting using Actuate's BIRT reporting system.

Zend Platform Overview

Zend Platform is a central management solution and run-time environment for:

- Configuration Management - Platform's architecture provides full control of the PHP application platform, including performance management settings, event thresholds, etc. allowing administrators to set up groups of multiple identical servers via:
  - Remote server configuration.
  - Clone configurations or parts of configurations from one server to another or from one server to an entire group of servers.
- Performance Management * - Platform is equipped with three management modules for tracking and improving speed and responsiveness of Web applications. These include Code Acceleration, Dynamic Content Caching and File Compression.
- PHP Intelligence - Platform features new technology that detects and recovers crashes, whether they occur in PHP itself, the database software, or your own application. The integrated suite of monitoring, detection and recovery features allows users to drill down to critical issues and optimizations quickly and easily.
- Job Queues - Zend Platform's Job-Queue provides PHP production environments with a standard approach to streamline offline processing. A Job-Queue server is services the Job Queue that provides the ability to reroute and delay the execution of processes that are not essential during user interaction with the Web Server.
- PHP/Java Integration - The Platform 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.

Environments

A typical environment for running any Web application consists of three basic components: Web servers for running the Web application, a load-balancer to handle traffic and a Firewall to protect form unauthorized entry into the hosting network.

Zend Platform, once introduced to this kind of an environment becomes a control environment for web server activity.

In an environment where a single web server manages activity, Zend Platform resides on the web server to provide system health and analysis information.

Moreover, environments that include several web servers, as clusters servicing a single Web application or a collection of clusters servicing different Web applications, Zend Platform serves as a single control center for system health information, cluster management and runtime process optimization.

The Zend Platform system diagram below, demonstrates where Zend Platform components typically reside in the PHP- enabled enterprise.

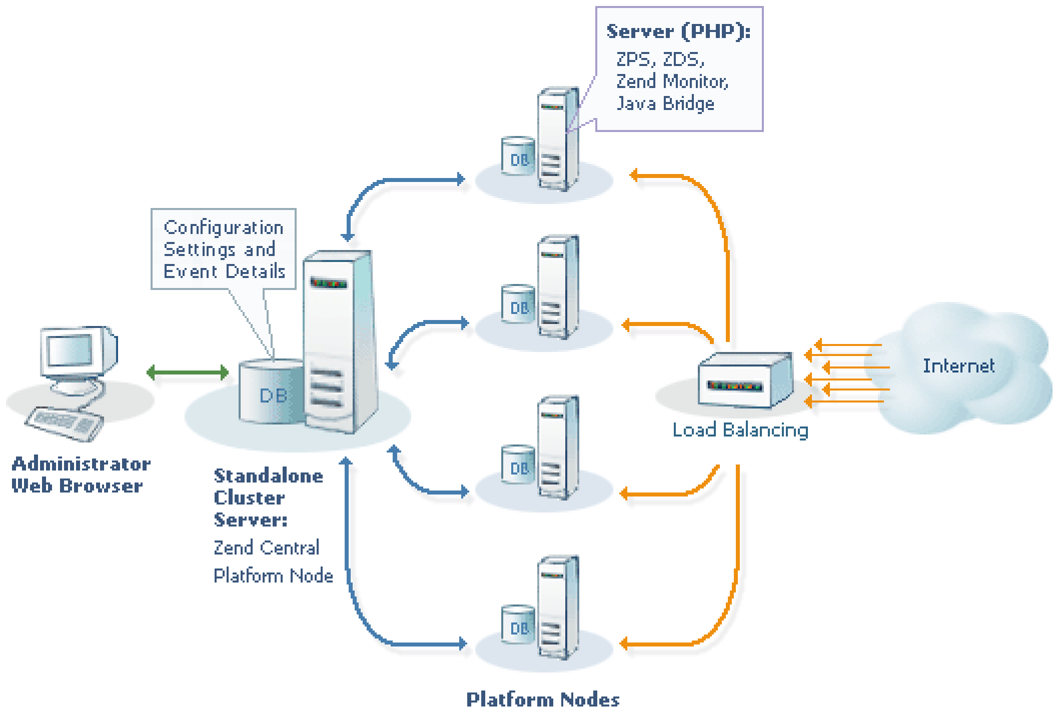


Figure 1 - Zend Platform System Diagram

The system diagram illustrates the following points:

- Zend Platform's Standalone Cluster Server is installed on a Web server.
- The System Administrator controls all Platform Central functions. Providing the ability to work with Platform from a single workstation using a standard Web Browser.
- Nodes host resident PHP-based services that fill requests from the Web.
- Load Balancing directs requests to available servers in the web farm.

Note:
Platform Server and the Platform nodes are separate entities; therefore, it is important to configure Firewall and security devices to allow communication between the nodes and the Platform Server. Identify which ports are in use and if necessary, open these ports on your Firewall.To read more about working with Firewalls and Nat go to "Configuring Zend Studio Tunneling settings".

Architecture

Zend Platform is a complete environment that provides rich functionality by interacting with the existing PHP in a simple and generic way. Zend Platform is a non-intrusive extension to an existing environment with minimal overhead that helps obtain enhanced performance and reliability.
Zend Platform extends the Zend Engine with the organization's execution environment, providing the platform on which to base Web services, business to commerce applications, content-management, Intranet and business-to-business applications.

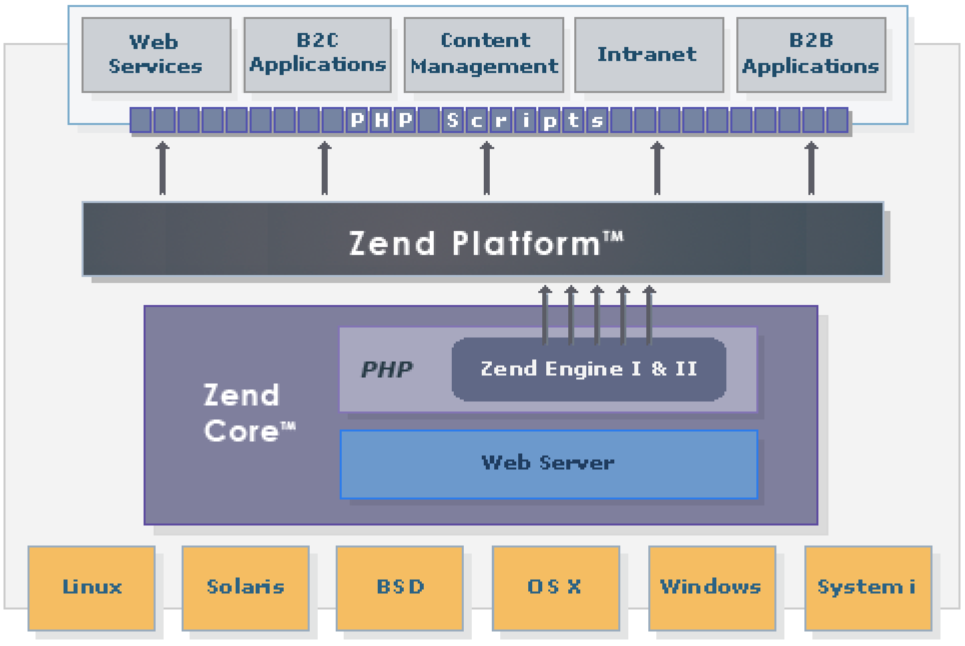


Figure 2 - Zend Platform and the PHP-enabled Enterprise

Zend Platform consists of two deployed components the Central Server (consists of a Server + Node) and the Node component.
Zend's Central Server is a central management component for governing node configurations and script performance information. The Central Server can be deployed as a standalone Zend Platform environment for a single server and for this reason, contains fully functioning node components. However, the prominent application for Zend Platform is multiple server/cluster based environments. Zend Central provides a single point of access and control for multiple nodes.
Nodes are web servers that run with Apache and service a PHP application. The Zend Platform components are installed on the node to report script, database and system activity to the Central Server. Each node installation also includes a debugger that is integrated with Zend Studio extended code management features such as profiling, debugging and correcting code directly on a node.
In essence, similar components are installed on the Central Server and the Nodes since the Central Server also performs as a node. However, the Central Server and the Node Components employ different modules for their overall activity.

Central Server

The Central Server provides the necessary functionality for handling event information, node management and performance monitoring.

No matter how many nodes are registered in the cluster, from the users point of view Zend Central provides an efficient and useful single point of entrance. Zend Central resides on the Central Server and is in charge of displaying Platform Administration for Central Server and Node configuration. Zend Central is the main communication component for collecting, storing, configuring and receiving information from the nodes.

Communication is carried out via regular TCP/IP communication and event information is stored in a dedicated database. Zend Central governs the PHP application performance and monitoring features including configurations for nodes, PHP and event collection.

The following illustration is a representation of Zend Platform Server components:

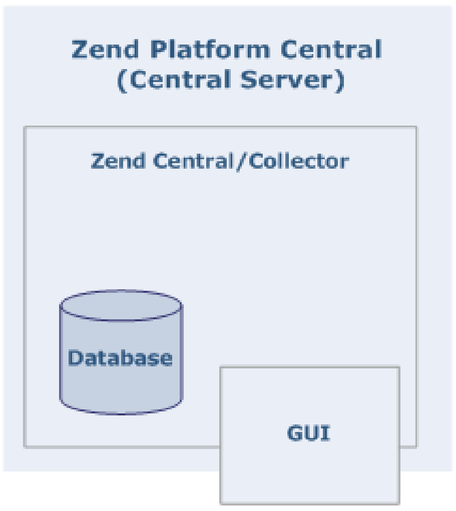


Figure 3 - Zend Platform Server Components

The Central Server is a central management component for managing and configuring nodes. The Central server component is installed once. All subsequent installations are for node components that are registered to this server in the installation process. Standalone environments base on one server only require the central component that also includes all the node components necessary for working in a single server environment.

The installation includes three main components:

1. Zend Central that includes information collection and functionality: Zend Performance, PHP Intelligence and the Java Bridge.
2. The Database is the main repository for event information collected from all registered nodes.
3. One of the main components of Zend Central is the Collector. This component collects and aggregates information from nodes in the cluster that is displayed in the Zend Platform PHP Intelligence module. The collector collects and aggregates information according to configurations applied to a single server or to several servers (grouped servers).

Nodes

The Nodes are the web servers that run PHP. Nodes are the individual servers that service a Web application and a collection of nodes is a cluster.
The central server governs clusters.

The following components need to be on each Node:

- Basic:
  - A supported operating system (Linux, Unix, Mac, i5/OS and Windows)
  - A Supported Web Server (Apache, IIS)
- PHP:
  - PHP version 4 or 5
- Zend Products
  - Zend Platform
  - Zend Download Server
  - Zend Java Bridge
  - Zend Optimizer
  - Zend Debugger

Nodes have to be registered with the Central Server in order to enable communication between the Node and the Central Server. There are two ways to register a Node to the Central Server: through the installation process or by manually registering the Node.
Zend Platform Nodes consist of several components that report information to the Central Server and provide debug capabilities for PHP scripts residing on a node.

- A Collector Component for transferring event information to the central
- Debug Infrastructure for debugging live pages directly from a node (This option is supported by Zend Studio)

The following illustration is a representation of Zend Platform Node components:

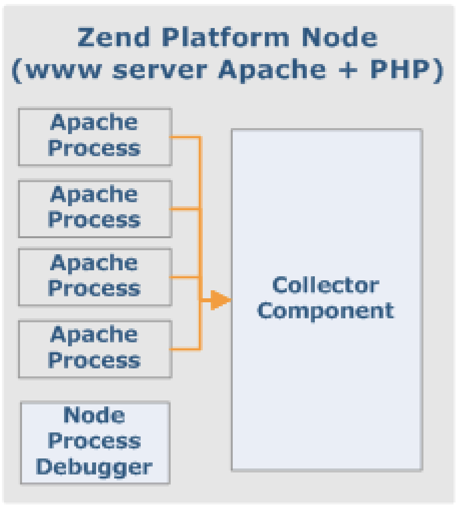


Figure 4 - Zend Platform Node Components

The Collector component listens to the running processes and collects event information (For more on Events go to “Configuring Events”), to be reported to the Central Server over a regular TCP/IP connection using SSL. However, only if the node has the appropriate certificate indicating that it is part of the cluster will the Central Server agree to receive event information from a node's collector. The type of information the Collector listens to and collects is event information determined by Event Rules that are configured on the Central Server. Event information is sent to the Central server where it is aggregated according to event type (more about event aggregation can be found in Appendix D - "Event Aggregation Mechanism").

The Debugger Infrastructure is enabled via the Zend Studio/Zend Platform Communication Tunnel that is geared to work in development and production environments. With the appropriate configuration, the Debugger Infrastructure can work through Firewalls or NAT devices that may be positioned between the Node and Zend Studio (more about Firewall traversal can be found in "Configuring Communication with Zend Studio". The Debugger Infrastructure provides full lifecycle support for editing debugging, profiling and deploying code by enabling to view and edit Event source code in the Zend Studio development environment. This provides Zend Studio users with access the remote debugger via the same communication tunnel that routs full-duplex traffic over HTTP. The Debugger Infrastructure utilizes the Communication Tunnel, ensuring that multiple servers can be debugged through the same Communication Tunnel at once.

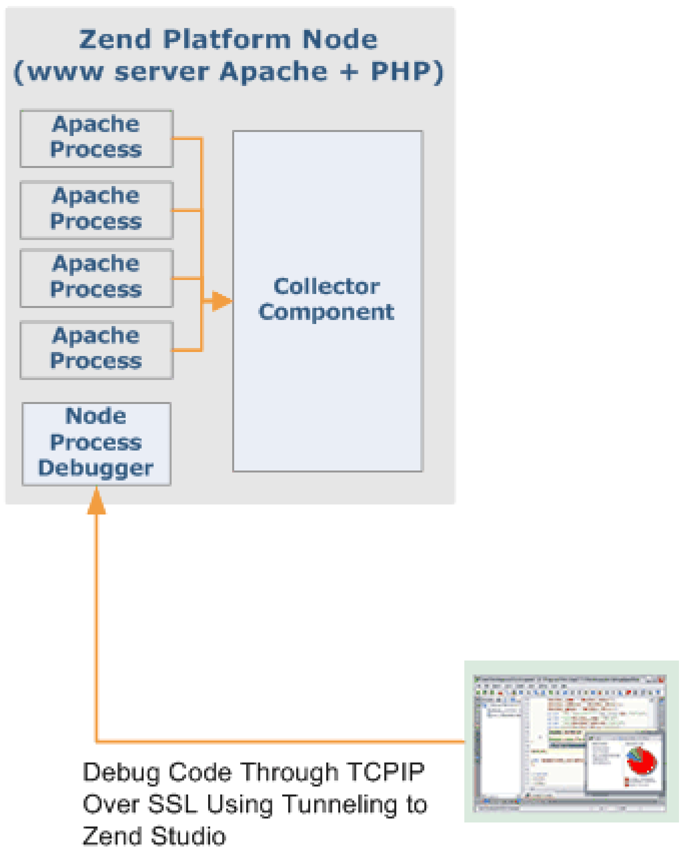


Figure 5 - Communication with Zend Studio

Central-Node Communication

Traffic between the Central Server and Node clusters mostly occurs from the nodes to the central server with the nodes reporting event information through the collector component to Zend Central. However, Zend Platform has a Server Status feature that periodically checks the availability of each Node in the cluster and provides up to date information regarding the components installed on the nodes.

The following diagram illustrates the communication between the Central Server and Nodes in a Cluster:

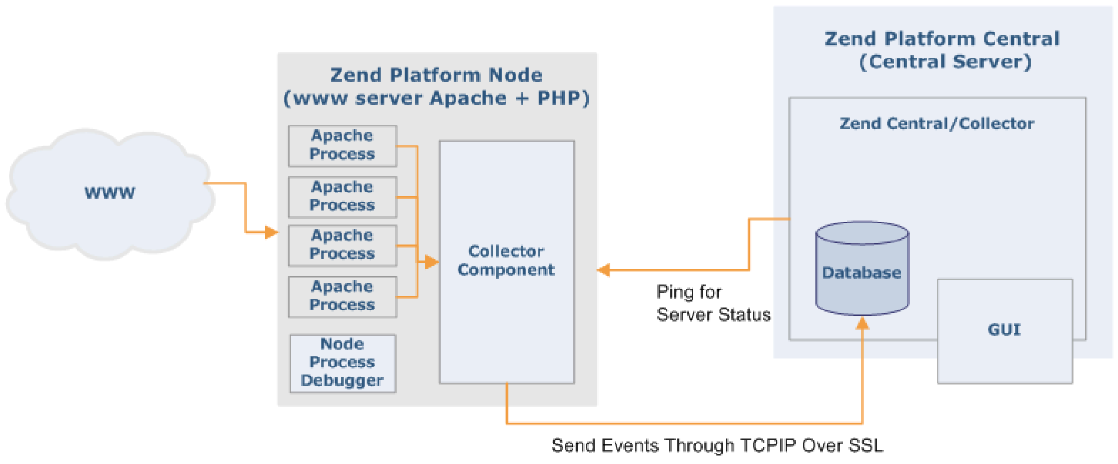


Figure 6 - Central Node Communication

Platform Administration, a Single Point of Access

Zend Platform’s sophisticated architecture enables to use the Central Server as a single point of access for node availability and configuration, enabling to configure node settings and behavior from the Central Server itself. This connectivity is achieved by the addition of Platform Administration components on the Nodes as well as on the Central Server in the installation process. In this process the Central Server’s URL is specified to the Nodes as a central control unit and from that point onwards, access and read write permissions to nodes, can be established from the Central Server.

PART II: ADMINISTRATION AND CONFIGURATION

The Zend Platform installation produces an out-of-the-box fully functioning version of Zend Platform. This installation includes basic default settings for monitoring events and code acceleration. At this stage Zend Platform already generates events and improves code generation. However, to benefit from Full-Power Cluster Management, Development integration with Zend Studio, Audit Trails, and much more, it is necessary to tune Zend Platform’s performance settings to suit your individual work environment.
In this chapter, each configuration task is detailed by module in a chronological order beginning from the initial configuration tasks to configurations that may rely on other settings.

The configuration actions addressed in this chapter are listed below:

- Cluster Management - Add the servers that you want Zend Platform to control. Each server should be added and then grouped to create a cluster environment to be treated as a single entity in terms of event collection.
- PHP Intelligence
- Configure Event Triggers - customize the Event Triggers to suit your working environment. The Zend Platform installation comes ready with default configurations; however, it is recommended that a person with an understanding of the environments settings and performance standards, configure Event Triggers accordingly.
- Configure Event Actions - once Event Triggers are configured the next logical step is to determine the actions and action rules that can be applied to Events generated according to Event Triggers.
- Performance - Adjust performance requirements as a way to benefit from Zend Platform's advanced performance features.
- Configuration
- Configure Studio Server / Tunneling - Zend Platform’s tight integration with the Zend Studio IDE provides an efficient means for improving the development lifecycle. Environments that contain security precautions such as firewalls and NAT can set up Zend Platform to provide a secure means for obtaining integration with Zend Studio without compromising an organization's security measures.
- Configuring PHP Settings - configure your PHP and Zend products directly from Zend Platform.
- Users and Groups - Grant different levels of permissions to different users provides a means for controlling actions performed in the environment and for enforcing work procedures. This is the last step to customizing Zend Platform to your working environment.
- License Management - Manage licenses for the central server and all nodes belonging to the cluster.
- Password Administration - Manage and change System passwords.

Cluster Management

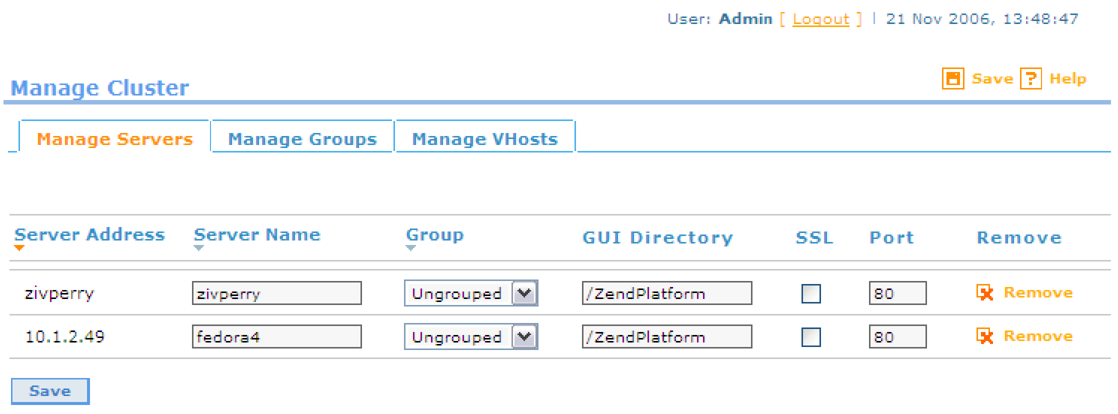
IN THIS CHAPTER…

SERVER MANAGEMENT
GROUP MANAGEMENT
VHOST MANAGEMENT
RESTRICTING ACCESS TO VIRTUAL HOSTS

Zend Platform manages clusters to make them available and manageable from a single location – the Central Server. Zend Platform treats clusters as a single unit for monitoring and management purposes. Moreover, through the Central Server each node in the cluster can be individually accessed. (The Central Server aggregates events originating from different servers; however, they include an identifier for each node on which the Event occurred).
The installation process (and later on the Setup Tool) is used for adding servers to the cluster to become Nodes belonging to the Central Server. Once users add a server, the server's settings can be applied and modified using the Central Server.

The following cluster setting options are available from: Platform | Cluster Management:

- Manage Servers - Configure, delete and define servers.
- Manage Groups - Group servers together for event reporting and configuration purposes.
- Manage VHost (Virtual Hosts) - Manually delete and define Virtual Hosts



Server Address	Server Name	Group	GUI Directory	SSL	Port	Remove
zivperry	zivperry	Ungrouped	/ZendPlatform	☐	80	Remove
10.1.2.49	fedora4	Ungrouped	/ZendPlatform	☐	80	Remove

Figure 7 - Manage Clusters Dialog

i5 / OS Note:
This view displays the central server only, Cluster management is not relevant for this environment.

Define Event Triggers once the servers have been configured and grouped.

Server Management

To access this tab go to Platform | Cluster Management and select Manage Servers.
The Manage Servers tab provides options for configuring and defining settings for servers added to Zend Platform using Zend Platforms Setup Tool.
Only servers installed with the Zend Platform Setup Tool will appear in the Manage Server tab.

To add a server to Zend Platform so that it appears in the Manage Clusters screen:

1. Run the Setup Tool (Please see the Zend Platform Installation Guide for details on Node installation).
2. Zend Platform automatically identifies registered servers and displays them in the Manage Servers tab.
3. The installation script sets the Server Name and users can now define the new server's settings.

Server settings are defined from Platform | Cluster Management | Manage Servers.

The Server management settings are as follows:

- Server Address - The actual hosts address (not editable).
- Server Name - The server’s name for identification and all references to the server from Zend Platform.
- Group - Designate a server to an existing group (new groups are added to the list from the Manage Groups tab).
- GUI Directory - States the location of the server’s Platform Administration Installation.
- SSL - Check the box if the server uses SSL.
- Port - Specifies the port with which the specific server works.
- Remove - Removes the server from the database (unregistered) and deletes all events related to the removed server.

These settings should only be changed if changes that may affect these settings, occurred since the node installation.

Removing a Server

1. To remove servers go to: Platform | Cluster Management and selecting Manage Servers.
2. Select a server from the list and click "Remove.
3. Manage Servers will remove the server from Zend Platform and all functionality will be disconnected.

Attempting to remove a server while another user is working on the server (through Zend platform), will activate a prompt message asking the user to select another server. Active Pop-up Blockers may interfere with this action causing a notification message to appear asking the user to actively select the "Select Server" option. This message will only appear once furthermore, deactivate all Pop-up Blockers when using Zend Platform or allow Pop-ups from the Zend Platform URL.

Group Management

The Group Management tab provides options grouping servers together for event reporting and configuration purposes.

Groups are created:

- To aggregate Events across nodes (only if the nodes are running the same Web application).
- To facilitate handling and managing groups of servers.

Note:
Groups should only be aggregated when the PHP application on all servers in the group is identical.

To create a new group:

1. Give the Group a name in the "Add a new group field" and press Add.
   A new group will be added to the list below.
2. If you want to aggregate all events that occur on the servers associated with the specific group select the 'Aggregated' option.

VHost Management

The Manage VHosts tab provides a way to manually define Virtual Hosts.
In general, virtual hosts are automatically added based on Event activity. However, Virtual Hosts only appear in the lists after an event is generated for a specific virtual host.
To ensure that all Virtual Hosts can be visible, an additional option has been added to manually add Virtual Hosts. This option allows users to create the actual virtual host list for any given server.
Virtual hosts can be added or deleted from this tab.
Virtual hosts are added per server and deleted in one of two ways:

1. Per virtual host name - do not select a specific server name before adding or deleting the virtual host.
2. Per virtual host for a specific server - select a specific server name before adding or deleting the virtual host.

When deleting a virtual host the database will permanently delete all events related to the deleted virtual host.

Restricting Access to Virtual Hosts

Defining Virtual Hosts provides a way to prevent certain users from gaining access to information regarding certain Virtual Hosts. Restricting access to a Virtual Host by user name is an additional level of authorization that is more precise than granting permissions per server.

To restrict user permissions by virtual host:

1. Go to Platform | User Management.
2. Select the User (who should be denied permissions) and click "Edit".
   The Edit User Wizard opens (see Adding and Editing Users for complete instructions on the Edit User Wizard).
3. In step two make sure the check box next to "No Server Restriction" is left unchecked.
4. Select the check box next to the virtual hosts that should be granted access.
   This will grant access to the selected virtual hosts only
5. Click "Finish" to close the Edit User Wizard and return to the User Management Screen.

PHP Intelligence

IN THIS CHAPTER…

CONFIGURING EVENT TRIGGERS
WHY CONFIGURE EVENT TRIGGERS?
FILTERING EVENT TRIGGERS
DEFINING EVENT TRIGGERS
CHOOSING AND DEFINING EVENT TRIGGERS
CUSTOM EVENTS
DEFINE EVENT ACTIONS

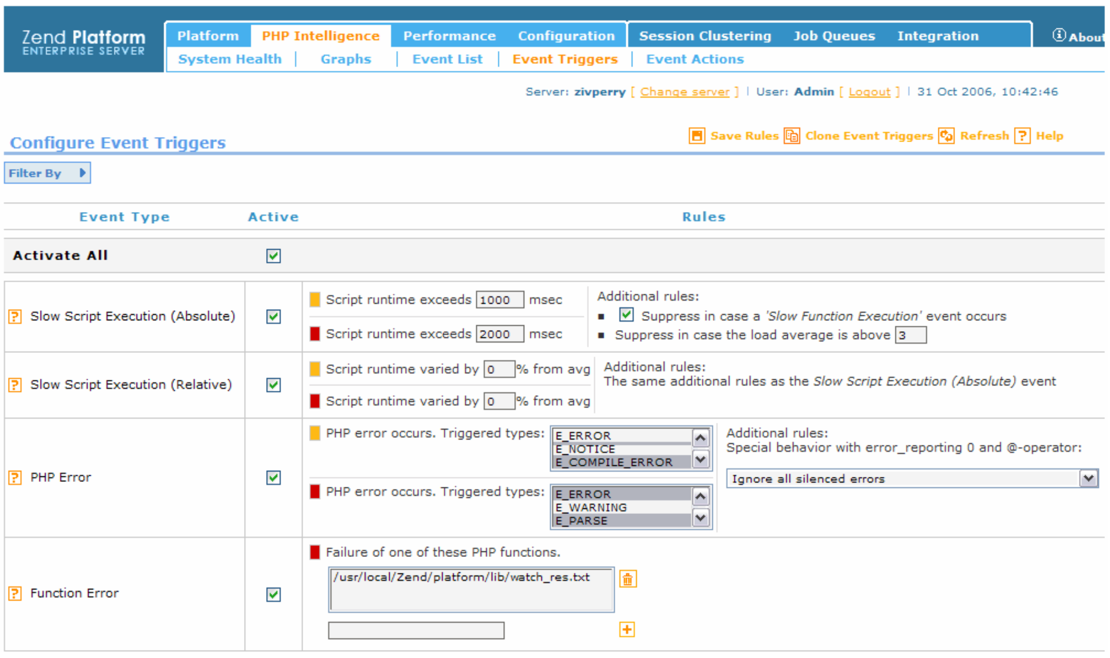
Configuring Event Triggers

Customize Event triggers to suit your working environment. Zend Platform comes ready with default configurations. However, a person with an understanding of the environment's settings and performance standards should construct the Event Triggers to suit each unique environment. Event Triggers define the conditions under which events are captured by the monitoring system (PHP Intelligence).

To Configure Event Triggers, go to PHP Intelligence | Event Triggers or use the Shortcut from Platform | Dashboard.

Users are prompted to select a node before entering the Event configuration screen as all configurations are made to a selected node. The top bars of screens display the name of the node, no name means the user is working directly on the Central Server.

For example, the image below displays the following text: Server name "zivperry". This means that the user is no longer working on the Central Server but working directly on the node (in our case a node aliased zivperry).



Server: zivperry [Change server] | User: Admin [Logout] | 31 Oct 2006, 10:42:46

Configure Event Triggers

Event Type	Active	Rules	
Activate All	✓		
Slow Script Execution (Absolute)	✓	🟧 Script runtime exceeds 1000 msec 🟥 Script runtime exceeds 2000 msec	Additional rules: ■ ✓ Suppress in case a *'Slow Function Execution'* event occurs ■ Suppress in case the load average is above 3
Slow Script Execution (Relative)	✓	🟧 Script runtime varied by 0 % from avg 🟥 Script runtime varied by 0 % from avg	Additional rules: The same additional rules as the *Slow Script Execution (Absolute)* event
PHP Error	✓	🟧 PHP error occurs. Triggered types: E_ERROR, E_NOTICE, E_COMPILE_ERROR 🟥 PHP error occurs. Triggered types: E_ERROR, E_WARNING, E_PARSE	Additional rules: Special behavior with error_reporting 0 and @-operator: Ignore all silenced errors
Function Error	✓	🟥 Failure of one of these PHP functions. /usr/local/Zend/platform/lib/watch_res.txt	

Figure 8 - Configure Event Triggers (Partial List)

The Event Triggers screen is used for defining and modifying Event Triggers to monitor events on a specific node. The table is used to define the conditions under which an event will be captured by the monitoring system.

The possible actions on this screen are:

- Configure Event Triggers for a specific server.
- View Event Triggers currently defined for the node.
- Filter the view of events displayed in the “Define Event Triggers” table.

To configure Event Triggers:

1. Click “Event Triggers” in the “Configuration and Management Tools” list of shortcuts. The "Select Server to Configure" screen opens.
2. Select a server from the “Server Tree.”
3. Click "Select" to open the “Event Triggers” screen for the selected server.

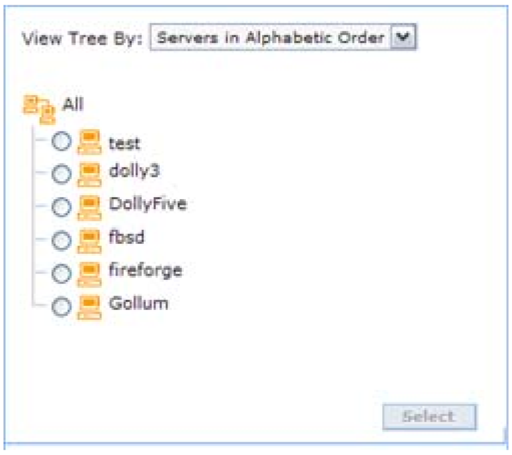


Figure 9 - Select Server to Configure

Why Configure Event Triggers?

Event Triggers are an essential tool for pinpointing bottlenecks in Web applications. Events not only indicate that one of the thresholds was breached they also collect information relevant to the occurrence to provide a full audit-trail for diagnostics.
In terms of the outcome, these thresholds can be directly translated into performance issues the end user may encounter. Therefore, the more Events resolved the better the application will run.
By using Event Triggers, scripts can be monitored to identify with precision the number of milliseconds or percentage it takes to execute a script. This identification is based on parameters that you can determine as acceptable performance thresholds.

Filtering Event Triggers

Zend Platform is equipped with 12 types of Events for monitoring performance and script execution. The default Event Trigger display is a non-filtered view that shows all the available Alerts. A filter is provided to allow displaying a selection of events by type.

To filter events:

1. Click "Filter By" to expand the filter list.
2. Use the two drop-down fields to select the Events to display by:
   - Events from – The area where the event originated (script, database, web server, etc.)
   - Event Types – Filter view to display Events according to their Event Type (The selection changes according to the area chosen in the “Events From” field).
3. Click "Go" to filter the view.

Defining Event Triggers

The fields that make up the Define Event Triggers table are:

- Event Type - The type of event that, under the rules defined, will produce an alert in the monitoring system.
- Active - When enabled for a specific event, Zend Monitor (node) will report alerts when they occur (This gives the user the right to disable an event for a particular server).
- Rules - Defines the conditions under which an event will produce a report. For example, (Red) Script Runtime Exceeds 500 Seconds means that the system will generate a critical (red) event—for Slow Script Execution (Absolute) type events, when meeting the condition (> 500 sec.).

Note:
The user defines the thresholds for both the moderate and severe events. Some events have only one level of severity (like "function error").

- To define whether Zend Monitor will report a specific event, enable/disable the event in the Active column of the Define Event Triggers table.
- To save the changes to Event Trigger definitions, click Save Rules.
  The database will update with the changes.

Each event type has its own advantages and characteristics. See the Chapter on Choosing and Defining Event triggers for more information about each Event type.

Note:
Events marked as “Performance Monitoring Events” have a special role in optimizing web application performance.

Choosing and Defining Event Triggers

Each Event Type has its own advantages and characteristics. Listed below are the different Event Types, their descriptions and recommended usage.

Note:
Events marked as “Performance Monitoring Events” have a special role in optimizing web application performance. This topic is addressed in the Chapter on "Implementing the Performance Lifecycle".

The following is a list of event types, click on the event name for more information about a specific event:

- Slow Script Execution Absolute - Generates an event when executing a script exceeds defined limits.
- Slow Script Execution Relative - Generates an event when script execution is lower or higher than the average script execution time.
- PHP Error - PHP Errors are used to identify all types of PHP errors. This type of event is useful in QA processes to identify problems that may have slipped through the cracks during production.
- Function Error - Generate a severe event when an error in one of the specified PHP Functions occurs
- Slow Function Execution - Identify bottlenecks within functions.
- Excess Memory Usage (Absolute and Relative) - Identify when scripts are using excess memory that can hinder the application's ability to perform.
- Database Error - Report database-related function fails.
- Slow Query Execution - Identify database performance slow queries that can directly influence Web server performance.
- Inconsistent Output Size - Verify the page is rendering the same output to the client each time.
- Load Average - Monitor the overall health of processes running on the server.
- Custom Event - Generate an event whenever the API function monitor_custom_event() is called from a PHP script.

Slow Script Execution Absolute

This is a performance-monitoring event.
Absolute Slow Script Execution is used to generate an event when executing a script exceeds defined limits. This function is used to maintain performance standards.
Default parameters are 500 msec for moderate, 2000 msec for severe alerts.

Additional Rules:

- Suppress in case a "Slow Function Execution" event occurs. Selecting this option ignores "Slow Script Execution" events caused by a slow function. This is to prevent double reporting, as PHP Intelligence will report these events as "Slow Function Execution" events.

- Suppress in case the load average is above X - Selecting this option ignores events that occur when the average number of active processes waiting for CPU time is above x active processes (3 active processes is the default value).

Note:
These additional rules are applied to the Absolute and Relative Slow Script Execution event types.

Slow Script Execution Relative

This is a performance-monitoring event.
Relative Slow Script Execution generates an event when script execution is lower or higher than the average script execution time. Parameters should be set to a certain percentage for moderate and severe alerts.
The default values for this event type are set to 0. To generate events, configure these settings to a value that suits required script run-time.

Additional Rules:

- Suppress in case a "Slow Function Execution" event occurs. Selecting this option ignores "Slow Script Execution" events caused by a slow function. This is to prevent double reporting, as PHP Intelligence reports these events as "Slow Function Execution" events.
- Suppress in case the load average is above x - Selecting this option ignores events that occur when the average number of active processes waiting for CPU time is above x active processes (3 active processes is the default value).

Note:
These additional rules are defined in the Absolute Slow Script execution cell.

Relative Events:

Event definitions are based on relative values i.e. percentage. Relative values are set according to warm-up settings, default value of 500 requests. If necessary, modify the default value by changing the zend_monitor.warmup_requests directive in the zend.ini.

PHP Error

PHP Errors identify all types of PHP errors such as:

- Hard errors that cause stops in page execution.
- Warnings that interrupt the end user experience.
- Notices that could lead to larger problems.

This type of event is useful in QA processes to identify problems that may have slipped through the cracks during production.

Description:

Used to generate severe or moderate events on selected PHP errors, when they occur, and identify real-time failures for given users.
To select a PHP Error Level, scroll through the selection and use CTRL for multiple selections. The trigger type list is the same; therefore severe event selection takes priority over moderate event selection.

Additional Rules:

Event reporting for PHP errors can be changed by setting error reporting to 0 or using the silence operator @.

There are three options for activating Additional Rules:

1. Always Report Errors - Ignore the error-reporting setting and the silence operator and report all PHP errors.
2. Report errors that match the error-reporting criteria - Ignore all PHP errors silenced using either the error-reporting setting or the silence operator.
3. Report any errors not silenced with the operator @ - Ignore the error-reporting setting and only ignore errors silenced with the silence operator.

Function Error

Functions return Function Errors and therefore offer specific information about the root of the error that does not always arise from PHP errors.
QA and Production use this Event for identifying run-time events, as opposed to PHP errors that identify code-related/syntactical events.
Function Errors can prove to be invaluable to an organization as they provide a different perspective on problems (view the outside problems through the eyes of PHP). Despite the fact that the code may be running okay, this Event indicates what other outside problems (i.e. network, database, web services, file system etc.) you may have, based on the PHP function's behavior. Issues like these used to be difficult to reproduce however with the complete audit trail and full problem context, Function Errors can be easily reproduced to a level of accuracy that mirrors the actual time of occurrence.

Description:

Generate a severe event when an error in one of the specified PHP Functions (built-in or user-defined) fails (returns a FALSE value).
To add a function, enter the name into the + field and press Add (+).

There are three ways to monitor PHP functions:

1. Specify the function name, object methods can also be used (for example, bar::foo).
2. Use wild cards (*) to specify a range of function names for example myFunc_* will select all functions beginning with myFunc_.
3. Specify the full path to a file containing a list of functions, each in a new line.

Note:
Database related functions are directed and reported as Database Errors (see the "Database Error" event type).

Watched Functions File Event Types

The Watched Functions file can add Function Error and Slow Function Execution event types (PHP Intelligence | Event Triggers) by entering a function in the field and pressing Add or specifying the full path to a file containing a list of functions.
When users apply the Watched Functions file to the "Function Error" Event Type, the functions included in the file will be monitored and an Event Details screen will be generated.

Slow Function Execution

This is a performance-monitoring event.

Slow Function Execution identifies bottlenecks within functions providing a more granular approach than finding bottlenecks in pages.

This type of event is useful in the production process for pinpointing performance bottlenecks by watching functions that the user specifies.

Slow Function Execution events provide a different perspective on problems (view outside problems through the eyes of PHP). Despite the fact that the code may be running okay, this Event indicates what other outside problems (i.e. network, database, web services, file system etc.) you may have, based on the PHP function's behavior. This Event is also useful for catching pure PHP functions that are performing slowly.

Description: Generates an event when function execution exceeds the setting defined in the rule. The default values are, 500 msec for moderate, 1000 msec for severe alerts. This applies to the functions selected in the additional rules section.

Additional Rules:

Generate events for specified PHP functions (built-in or user-defined).

There are three ways monitor functions:

1. Specify the function name, object methods can also be used (for example, bar::foo).
2. Use wild cards (*) to specify a range of function names for example mysql_* will select all functions beginning with mysql_.
3. Specify the full path to a file containing a list of functions, each in a new line.

Note:
Database related functions reported as Slow Query Execution events (see the "Slow Query Execution" event type).

When applying a Watched Functions file to "Slow Function Execution" events, the functions included in the file are monitored and Event Reports are generated when the function execution exceeds the values defined to trigger a moderate or severe event.

Excess Memory Usage (Absolute and Relative)

This is a performance-monitoring event.

(Absolute – a customer configured hard number; Relative – a customer configured percentage)

Excess Memory Usage events identify when scripts are using excess memory that can hinder the application's ability to perform.

Production environments mainly use this event type but QA can also benefit from monitoring by KB or percentage of memory used by a script to execute.

Description:

- Excess Memory Usage (Absolute) - Generates an event when memory for PHP script execution uses more than a set amount of KB for moderate events and severe events.
- Excess Memory Usage (Relative) - Generates an event when memory use for PHP script execution is above or below average, a set percent for moderate and severe events.

Note:

Both Event Types are only active if the PHP is compiled with memory limit. (Compile the PHP, with the configure switch "--enable-memory-limit".

The default values for both of these event types are set to 0. To generate events, configure these settings to a value that suits required memory usage.

Relative Events:

Event definitions are based on relative values i.e. percentage. Relative values are set according to warm-up settings, default value of 500 requests. If necessary, change the zend_monitor.warmup_requests directive in the zend.ini.

Database Error

Database Error Events report function errors such as:

- Connection errors
- Database selection errors
- General database function errors

These events do not require any additional configurations to the database. Production environments can use the information to delineate between a PHP problem and a database problem.

Database Errors can prove to be invaluable to an organization as they provide insight into the Database reliability along with a different perspective on problems (view outside problems through the eyes of PHP). Issues like these used to be difficult to reproduce however with the complete audit trail and full problem context, Database Errors can be easily reproduced to a level of accuracy that mirrors the actual time of occurrence.

Description:

Database errors generate events when database-related functions fail. This event is directly associated to the "Function Error" event and is activated and defined in correlation with this event type.

Database functions that should be reported are defined (or deleted) from the "Function Error" functions list.

Note:

To view supported databases, see the database related function prefixes listed in: <install_dir>/lib/db_functions.txt or in windows <install_dir>\lib\db_functions.txt.

Slow Query Execution

Slow Query Execution events identify slow queries that are related to database performance that can directly influence Web server performance.

Slow queries, if not pinpointed, can bring the server down by:

- Causing excess web server processes (Apache).
- Hang up queries in the database causing slower responses in the database.

These events do not require any additional configurations to the database. Production environments can use this information to pinpoint performance bottlenecks in the database.

Description:

Generates an event whenever database related function execution rises above the given threshold. This event is directly associated to the "Slow Function Execution" event and is activated and defined in correlation with this event type.

Database functions that should be reported are defined in the "Slow Function Execution" function list (in additional rules).

Note:

To view supported databases, see the database related function prefixes listed in: <install_dir>/lib/db_functions.txt or in windows <install_dir>\lib\db_functions.txt.

Inconsistent Output Size

Inconsistent Output Size events verify that pages render the same output to the client each time. If pages do not render the same each time, some clients are seeing different output than others and an error has occurred.
Production environments use this event as an indicator for possible usability issues.

Description:

Inconsistent Output events generate an event whenever the output size is below or above the normally produced average output. The default values for this event type are set to 0. To generate events, configure these settings to a value that suits acceptable variance in percents from output to output of scripts.

Relative Events:

Event definitions are based on relative values i.e. percentage. Relative values are set according to warm-up settings, default value of 500 requests. If necessary, the default value can be modified manually by changing the zend_monitor.warmup_requests directive in the zend.ini.

Load Average

Load Average events monitor the overall health of processes running on the server.
This event is used in production to highlight critical situations that might require analysis during high traffic situations.

Description:

In Unix, Mac, i5/OS and Linux this event is triggered when the number of active processes waiting for CPU time, is higher than the number defined in the rule. The default definitions for are set to 0 for moderate and 0 for severe events.

In Windows this event is triggered when the CPU exceeds a certain load percentage threshold. The default definitions for Windows are 90% for moderate and 95% for severe.

To start generating events set a logical value based on the server's capabilities.

Custom Events

Custom events are a unique type of event for Zend Platform users to initiate events from scripts.
This type of event is different from other event types because it controls event generation as opposed to other events that trigger events by a certain occurrence.
Custom events are used to generate an event whenever the API function *monitor_custom_event()* is called from the PHP script.

Description:

This event type enables the generation of an event on occurrences that are not necessarily built-in Zend Platform events (error and performance issues). Custom events are used whenever you decide that it is significant to generate an event in a certain situation. Each event type is given a name for easy identification ($type).

Function Usage:

monitor_custom_event(string $class, string $text[, integer $severe, mixed $user_data])

Parameters:

- *$class* – helps to define several types of custom events. This description will be showed in the Event List and in the Event Details.
- *$text* - error text used to describe the reason for the event. This text will appear in the Event Details.
- *$severe* - the severity level of the triggered event, default value is Severe.
- *$user_data* - adds a PHP variable that will be viewed in the Event Details (in Event Context-> Variables->User Defined). This forms the stored event data (similar to the information obtained in a PHP error event).

Aggregation takes place for these events when two events occur in the same place and have the same $class $text $sever(ity)

Note:

Action Rules defined for these events should be set to “send to URL” rather than “sending by e-mail” as there is only one definition for these events and event reports sent to a URL can be easily forwarded elsewhere. This is to prevent the overloading of e-mail. If we use the e-mail action, for every custom event, e-mail will be sent, and there can be many classes of custom events. However if the URL action is used, a script can be used to identify the event’s class and different behaviors can be implemented according to class.

Define Event Actions

Once Event Triggers are configured, the next logical step is to determine event actions and action rules.
All Events are immediately reported inside Zend Platform's PHP Intelligence module. Events can be viewed from: PHP Intelligence | Event List. However, Events and the information included in the Event Details screen can also be sent via E-mail or to a URL by configuring Event Triggers.
Actions are applied to generated Events.
All Events are immediately reported inside Zend Platform's PHP Intelligence module.
Events can be viewed from: PHP Intelligence | Event List.
Event Actions enable sending Event details via E-mail, URL or by SNMP (Simple Network Management Protocol).

To configure Event Actions go to: PHP Intelligence | Event Actions.

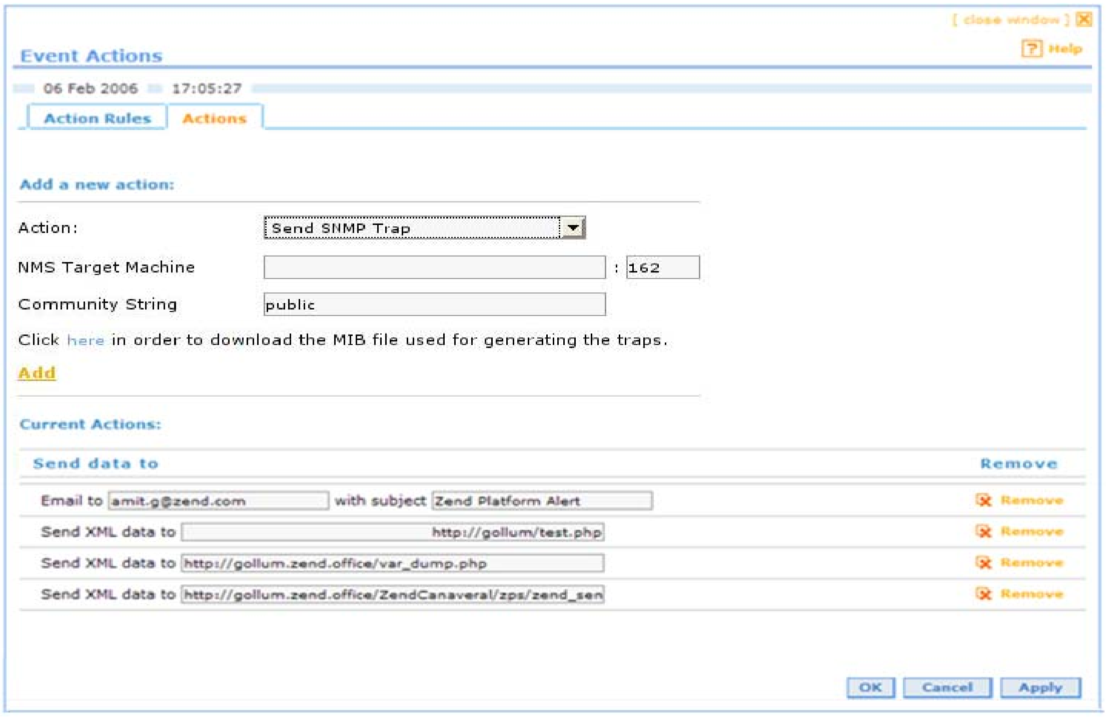
There are two steps to defining event actions. The first is to define "Actions and the second is to define "Action Rules".

- Actions determine how the Event Details will be sent by specifying an e-mail address, a URL or an SNMP alert.
- Actions Rules determine which events by specific criteria will be sent.

Note:
Read more about SNMP Traps in Appendix G - About SMNP.

Define Actions

Clicking the Event Actions URL opens the Actions dialog. This dialog allows you to define or remove Actions for the entire cluster.



Add a new action:

Action:	Send SNMP Trap
NMS Target Machine	: 162
Community String	public

Click here in order to download the MIB file used for generating the traps.

Current Actions:

Send data to	Remove
Email to amit.g@zend.com with subject Zend Platform Alert	Remove
Send XML data to http://gollum/test.php	Remove
Send XML data to http://gollum.zend.office/var_dump.php	Remove
Send XML data to http://gollum.zend.office/ZendCanaveral/zps/zend_sen	Remove

Figure 10 - Event Actions

From the Actions screen you can:

- Add/remove global Action Types from a central administrative station
- View Action Types currently defined in the system

To add an Action:

1. Select one of the options from the Action Type drop-down list.
2. Depending on the selection e-mail, URL or SNMP trap the action type details will change.
3. Enter the information according to the selected Action Type:
    - Target URL for the action type “Submit a report to the specified URL”
    - Recipient Address and Subject for the Action Type, “Send a report via e-mail.”
    - SNMP:
        - NMS Target Machine - The SNMP Trap's destination address.
        - Community String - The Community and port (default port is 162, and the default community string is 'public').
        - Download MIB (Management Information Base) file - Browse to find the MIB file and place it in the NMS. If a problem occurs accessing the MIB file, the relevant error will be given instead.
4. Click "Add" to add the new Action Type to the “Current Action Types” list.

Note:
These Action Types can now be associated with Action Rules (see below). You can also change or remove the Action Type settings at any time.

Zend Platform supports three types of reports:

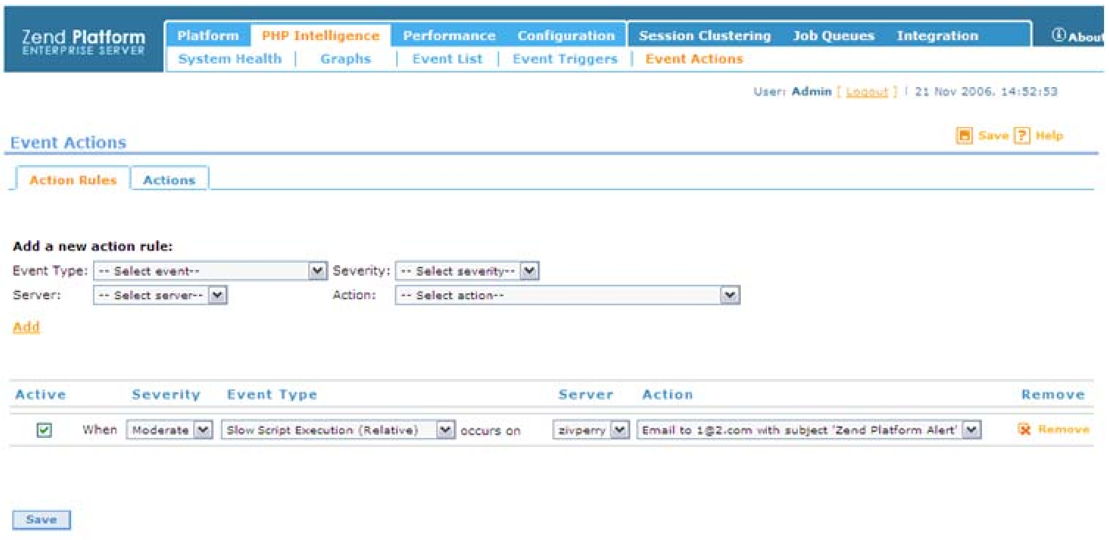
- E-mail Report - sends a text report to an e-mail recipient. This type of report is typically preferred by users who need to be notified of an event, but do not require the content of the report to be available for further use.
- Send SNMP Trap - Sends an SNMP trap to a NMS address containing the events parameters. SNMP traps are used in order to send an SNMP alert to a management server. The process is as follows; once the SNMP trap action is set and an action rule with it is defined, as soon as there is an occurrence in the system the rule is triggered and an SNMP trap will be sent to the NMS address provided by the user when the action was set.

- XML Report - a structured XML report which is not only informative, but which can be made available for further use. For example, the .xml event data could be used as input for a monitoring script. The structure of the .xml report follows the structure shown below:

```
#each attribute exists if it exists in the Event Details screen
<?xml version="1.0" ?>
<event type event_id timestamp time severity>
#if there is an error:
    <error type>error text</error>
    <stats triggered_value avg load_average/>
#if there is a source file:
    <source file line/>
    <script name host uri>
        <vardata type name value/>
    </script>
#if there is a function:
    <function name>
        <args>
            <arg num value/>
        </args>
    </function>
#if there are included files:
    <included_files>
        <file name\>
    </included_files>
#if there is a backtrace for this event:
    <backtrace>
        <call depth function file line/>
    </backtrace>
</event>
```

Define Action Rules

The Define Action Rules screen is accessed from: PHP Intelligence | Action Rules.
This screen ties together the elements of the rule-based notification system (monitoring and reporting) by creating a logical rule that can be understood as follows:
When an Event of a user-defined Severity occurs in the user-designated Server, a specific Event Action (notification) will be invoked.



User: Admin [Logout] | 21 Nov 2006, 14:52:53

Event Actions

Add a new action rule:

Event Type: -- Select event-- Severity: -- Select severity--
Server: -- Select server-- Action: -- Select action--

Active		Severity	Event Type		Server	Action	Remove
✓	When	Moderate	Slow Script Execution (Relative)	occurs on	zivperry	Email to 1@2.com with subject 'Zend Platform Alert'	Remove

Figure 11 - Define Action Rules

From this screen you can:

- Add/remove an Action Rule currently defined in the system.
- View Action Rules currently defined for an Action Type in the system.
- Edit an existing Action Rule and apply the changes.
- Disable an Action Rule.

To define Action Rules for a server:

1. Select the Define Action Rules tab
2. Enter Action Rule parameters in the Add a New Action Rule area.
   a. Select an Event from the Events combo. (For a complete list of Events supported in the current version of Zend Platform, refer to the Configure Alert Rules section above.)
   b. Select the severity from the drop-down list.
   c. Select a Server from the drop-down list of servers currently defined in the system.
   d. Select an Action from the drop-down list of Actions currently defined in the system.
3. Click "Add" or "Save".
   a. Clicking "Add" adds the new Action Rule to the list of Action Rules defined in the system.
   b. Clicking "Save" applies the changes to a rule that you have edited.

Note:

Read more about how your organization can leverage information generated by events in the Tutorial - "Integrating Existing and Legacy Applications".

Performance

IN THIS CHAPTER…

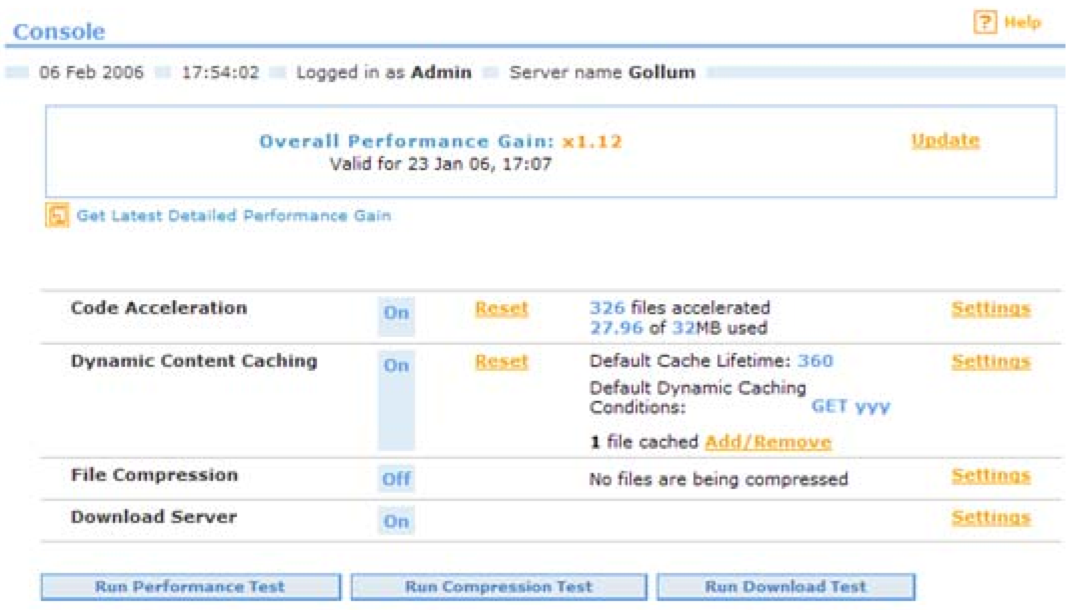
CONFIGURING PERFORMANCE
PERFORMANCE TAB
SETTINGS
CODE ACCELERATION SETTINGS
DYNAMIC CONTENT CACHING SETTINGS
FILE COMPRESSION SETTINGS
DOWNLOAD SERVER SETTINGS
FILE VIEW TAB
DYNAMIC CONTENT CACHING
FULL PAGE CONTENT CACHING
FILE COMPRESSION
BLACKLISTS
CONFIGURING THE ZEND DOWNLOAD SERVER (ZDS)
TESTING THE ZDS

Configuring Performance

Customizing performance is a way to benefit from the Zend Platform advance performance features. Setting initial defaults enable the use of basic performance features. Additional configurations can be applied, to customize performance to correspond with organization-specific requirements. These configurations are addressed in the Chapter "Performance Lifecycle".

Performance Tab

Zend Platform's Performance settings are configured and viewed from: Performance |Console.
The Console section of the Performance tab is a main performance management screen through which basic details and commonly used Performance actions can be viewed as follows:



06 Feb 2006 17:54:02 Logged in as **Admin** Server name **Gollum**

Overall Performance Gain: x1.12
Valid for 23 Jan 06, 17:07

Code Acceleration	On	Reset	326 files accelerated 27.96 of 32MB used	Settings
Dynamic Content Caching	On	Reset	Default Cache Lifetime: 360 Default Dynamic Caching Conditions: GET yyy **1** file cached Add/Remove	Settings
File Compression	Off		No files are being compressed	Settings
Download Server	On			Settings

Figure 12 - Performance Tab - Console

Initially the Console shows the installation defaults' regarding which feature is enabled (On or Off). Once changes are applied the console will be automatically updated with the new configuration settings (In some most changes are applied by restarting the Web-Server).

The following table lists the details and options available from the Console tab:

Component	Console Details	Actions
Overall Performance Gain	Shows the last performance test results.	Update - leads to Performance | Testing | Analyze Site. From this Tab site analysis tests can be run and results can be viewed. Get Latest Detailed Performance Gain - leads to Performance | Testing | Analyze Site, with the last performance test results expanded on the screen.
Code Acceleration	Shows the Code Acceleration component's status (On/Off) and basic code acceleration statistics.	Reset – Clears the Code Accelerator memory. Settings - Leads to the Code Acceleration section of the Settings Tab.
Dynamic Content Caching	Shows the Content Caching component's status (On/Off) and basic Content Caching statistics.	Reset – Clears the content cache. Settings - Leads to the Dynamic Content Caching section of the Settings Tab. Add/Remove - leads to Performance | File View, where Cache settings can be added/Removed.
File Compression	Shows the Compression component's status (On/Off) and file compression settings.	Settings – Leads to the File Compression section of the Settings Tab.
Download Server	Shows the Download Server's status (On/Off).	Settings – leads to the Download Server section of the Settings Tab.

At the bottom of the Console there are shortcuts to individual Test functions as follows:

- Run Performance Test - runs a test that evaluates improved performance via Code Acceleration and Dynamic Content Caching.
- Run Compression Test - runs a test that evaluates improved performance via File Compression.
- Run Download Test - runs a test on a selected file that uses the Zend Download Server (ZDS) to check the positive affect the ZDS has on performance.

Note:
Selecting one of these options opens a link to the appropriate option in the Testing Tab (Performance | Testing) and will not run the test before setting the preferences.

Once the overall functionality of the console has been established, the console can be used to apply initial performance settings. These settings are related to the following features:

- The File View to make sure that all Virtual Hosts are visible and select files to be cached (full page) in the Dynamic Content Caching section
- Use the ZDS (Zend download Server), to maximize large download handling over HTTP.

Settings

From the Performance Settings tab for a specific node, you can globally define the settings for all the modules of the Performance suite of tools: Code Acceleration, Dynamic Content Caching, File Compression and Zend Download Server.

From this configuration pane, you can:

- Configure performance monitoring and improve functions for a specific node
- Manage Performance settings for the network from a central administrative station

Code Acceleration Settings

Code Acceleration Enabled

"On" - The Code Acceleration is active and working.

"Off" - The Code Acceleration is not in use.

Accelerator Memory

The amount of memory allocated for use by the Code Acceleration for storing data structures and accelerated files.

Recommended: The memory allocated should correlate with the amount of scripts that you have, their size and complexity. Typically, 32MB is enough.

Memory Reclaim Threshold

During normal operation, some of the Code Acceleration memory may become unavailable for use. When the Code Acceleration runs out of memory, it will check how much of its memory is in use, and how much is unavailable. If the amount of unavailable memory is beyond this reclaim threshold, the Accelerator will perform an automatic restart, to reclaim all memory.

Recommended: 5%

Maximum Accelerated Files

The maximal number of files that will be accelerated

Recommended: Set this value to about 20% more than the actual number of scripts on your server. Typical memory usage ratio is a few hundred KB per thousand accelerated scripts.

Extensions for PHP Files

If your PHP files end with any other extension (rather than the default *.php extension), add all the extensions here separated by commas.

Dynamic Content Caching Settings

Dynamic Caching Enabled

"On" - The Dynamic Caching is active and working.

"Off" - The Dynamic Caching is not being used.

Maximum Cache Size

The maximum disk size allocated for caching. Occasionally and for short periods, this value may be exceeded but only until the next time that the cache cleaner deletes the files that expired.

For unlimited cache size, enter "0".

Minimum Free Diskspace

The minimum amount of free disk space that cannot be exceeded during caching. Reaching this value will end any further caching. The caching will resume as soon as the space is greater than this value.

Maximum Cached File Size

The maximum output cache file size allowed. An output cache file that exceeds this value will not be cached.

Default Cache Lifetime

The lifetime (in seconds) of a cached data. The data will be re-generated if the cached version is older than the expiration time.

Default Dynamic Caching Conditions

By default, the Dynamic Content Caching will cache each request, based on its full URI. You can modify the settings to be more general or more specific, as desired.

File Compression

"None"

All files are sent to the browser as is.

"Only cached files"

The cached files are transferred to the browser in a gzip format, if the browser supports the format. Other files are sent to the browser as is.

"All files"

All files are sent to the browser in a gzip format, if the browser supports the format.

Note:

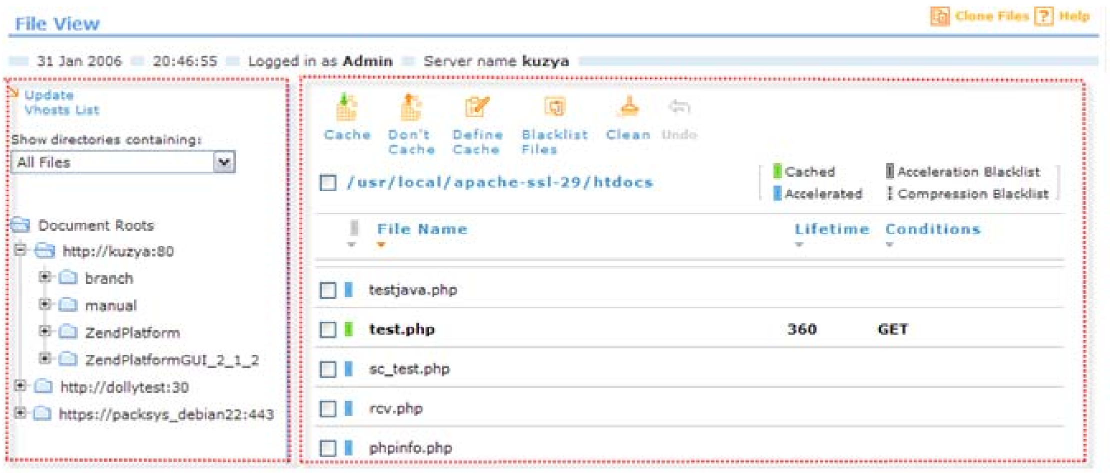
Compressing all files may cause some overhead and affect the overall performance. Use "All files" if your main concern is improving the download time for the user.

File View

Most performance configurations are done in the File View screen. Before describing the configuration tasks, it is important to understand the screen's layout and functionality.

The File View screen consists of two sections:

1. The Tree View on the left displays the list of directories and provides options for filtering the view (By status: Cached, Accelerated, Acceleration Blacklist, Compression Blacklist) the Virtual Hosts list is also updated from here.
2. The File View on the right lists files and their status, and also includes the different caching, acceleration and compression options that can be applied to selected files or to entire directories.



File Name	Lifetime	Conditions
testjava.php		
test.php	360	GET
sc_test.php		
rcv.php		
phpinfo.php		

Figure 13 - File View - Tree View and File View Sections

Note:
Status changes made in the File View screen are immediately reflected on the screen. However, the actual changes take affect only after manually restarting the HTTP Server. A reminder to restart Apache, will appear on screen after changes are made and disappears only after restarting the server.

Tree View

The Tree View on the left displays the directories available under a selected document root. All the directories are listed by default.

The list includes filtering options to display directories by file type. To filter the list, select the file type from the drop down list.

The filtering options are: All Files, Cached Files, Acceleration Blacklist Files, and Compression Blacklist Files.

To refresh the list of files displayed on the right: click a directory's name.

Tree View - Virtual Hosts List

The Tree View lists all the directories and files in the default Document Root as well as any Document Root listed in the Vhost List. Displaying all the directories and files enables to view files included in the Document Root directly from Zend Platform and select files for Dynamic Content Caching.

Upon initial setup it is important to verify that all the applicable Virtual Hosts are included in the Virtual Hosts List for two distinct purposes:

1. To Benchmark test cached files. The Zend Benchmark (Performance | Testing) tests URLs per Virtual Host.
2. To update the File View option to reflect all Virtual Host's Document Roots.

The Tree View option maps the entire Server's Document Roots providing a single view for displaying all the available directories and their contained files. Adding and deleting a Virtual host should reflect the actual Document Root activity on the server. (I.e. if you add/remove a document root from the server, you should add/remove its respective Virtual Host from the list).

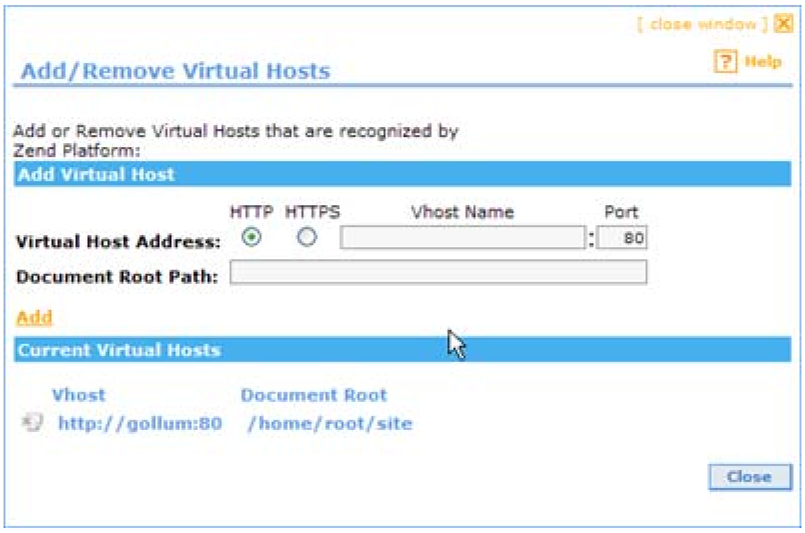
Note:
The initial installation process creates a default Virtual Host list however; this may not include all the required virtual hosts and some may need to be added/removed.

Updating the Virtual Hosts List:

In order for Platform Performance to display files residing in a particular Document Root, you must add the Virtual Host to the list
The Virtual Host list in the File View reflects the current Virtual Host list as defined in Manage Cluster. You can update the list directly from the File View.

To Add or Remove a Document Root:

Go to Performance | Settings, and select Update Virtual Hosts to open the Virtual Hosts list.



Add or Remove Virtual Hosts that are recognized by Zend Platform:

Virtual Host Address: HTTP ● HTTPS ○ : 80

Document Root Path:

Vhost	Document Root
http://gollum:80	/home/root/site

Figure 14 - Update Virtual Hosts List

This screen includes two sections:

- Update Virtual Hosts List – add a Virtual Host
- Current Settings – Remove a Virtual Host and view current virtual hosts on the server.

Make sure that all the necessary Virtual Hosts are displayed in this list if not use "Update Virtual Host " to modify the list as necessary.

To update the Virtual Host list:

1. Go to Performance | File View and select Update Virtual Hosts List from the options at the top of the screen. This will open the Update Virtual Hosts List screen.
2. Specify the virtual host's details and provide an alias for the Virtual Host under Vhost Name.
3. Press Add to save the new Virtual Host and add it to the Virtual Hosts list.

File View - Dynamic Content Caching

The File View displays files in a table, which can be sorted by column. The sorting options are: Status, File Name, Lifetime and Conditions.

Once all the Virtual Hosts have been established, and the default Caching Conditions have been set; specific Content Caching settings can be applied to selected files or directories.

Content Caching activities include the following in chronological order:

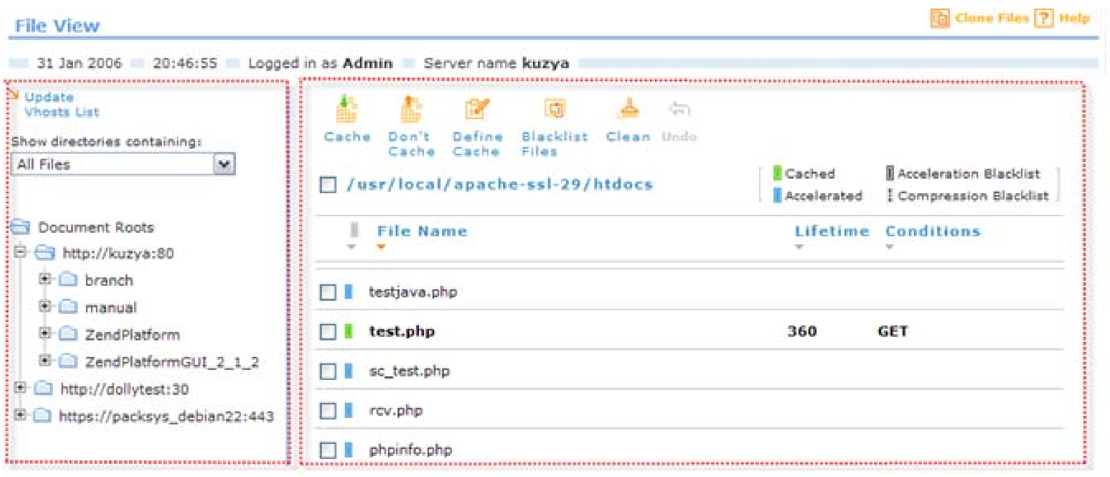
1. Define default caching settings
2. Modify file settings
3. Fine tune caching conditions
4. Define files to blacklist

Define Caching Settings:

The File View screen lists all the directories and files in the default Document Root as well as any Document Roots listed in the “Vhost List”.

Any cached file that has not been explicitly defined, automatically inherits the default cache settings

To open the File View screen, go to: Performance | File View.



File View

Clone Files | Help

31 Jan 2006 20:46:55 Logged in as **Admin** Server name **kuzya**

Update Vhosts List

Show directories containing:

All Files

- Document Roots
  - http://kuzya:80
    - branch
    - manual
    - ZendPlatform
    - ZendPlatformGUI_2_1_2
  - http://dollytest:30
  - https://packsys_debian22:443

Cache | Don't Cache | Define Cache | Blacklist Files | Clean | Undo

/usr/local/apache-ssl-29/htdocs

Cached | Accelerated | Acceleration Blacklist | Compression Blacklist

File Name	Lifetime	Conditions
testjava.php		
test.php	360	GET
sc_test.php		
rcv.php		
phpinfo.php		

Figure 15 - File View

The File view screen provides the following Full Page Content Caching options:

- Cache - Select files to be cached and select, to enable content caching for the selected files.
- Do not Cache - Disables Content Caching for the selected files.
- Define Cache - Displays the “Define Caching Conditions” screen to add detailed configurations for selected files.
- Update Blacklists - Opens and modifies the current Blacklist.
- Clean - Cleans cached file copies.
- Undo - Cancels the last change to the settings.

Modify File Settings

To Modify File Settings:

1. Select a directory in the Tree View. The list of files residing in that directory is displayed in the File View.
2. Check the box next to the file(s) you wish to modify or the directory to select all files (and sub-directories) under it.
3. Click on the relevant icon in the toolbar.

Fine Tune Caching Conditions

Caching conditions can be changed per file or group of files.

To modify Caching Conditions:

1. Check the box next to the directory or cached file(s).
2. Click Define Cache to open the Define Caching Conditions Dialog. (Alternately, click on a Cached file i.e. a file with the Cached indicator 🟩 next to it).
3. Apply Caching settings and press Save to save and close the dialog.

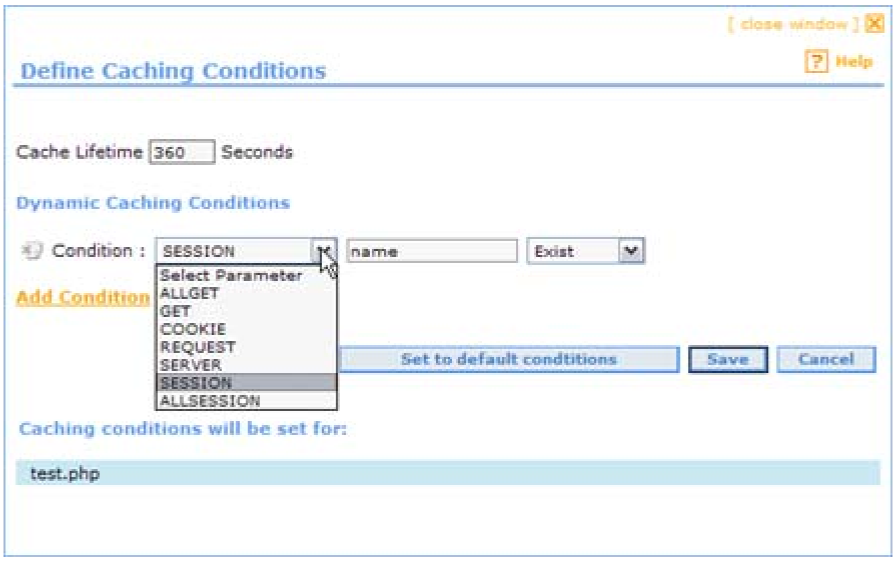


Figure 16 - Define Caching Conditions

Modified settings are displayed in the File View Tab next to the selected file/s. Restart the Web server in order to activate the new caching conditions. The restart server message remains on the screen until the server is restarted.

The Define Caching Conditions dialog includes three buttons:

- Restore Defaults - Returns to the default caching settings.
- Save - The new settings are saved and are reflected on the screen but the changes will take effect only after restarting the server.
- Cancel - Cancels the new changes and returns to the previous settings.

Caching conditions may also apply to Variables stored in an Array.

Note:
Go to “Default Dynamic Caching Condition Parameters” for a complete list of applicable conditions.

Note:

The zend_cache.ini file contains the list of all the files and directories assigned for Dynamic Content Caching including all the Conditions, as follows:

Use the File View to define the files and directories to be cached.

Do not edit this file manually!

```
zend_cache.lifetime=360
zend_cache.depends=ALLGET
zend_cache.path="/usr/local/apache/htdocs/hello.php"
zend_cache.lifetime=360
zend_cache.depends=COOKIE:my_cookie
```

A large cache.ini file can possibly result in slow performance. Therefore, it is recommended to un-cache (in the File View) any file deleted from the server.

Define files to Blacklist

The Blacklist separates acceleration and compression settings for files. With the blacklist users can prevent files from being accelerated or compressed.

Dynamic Content Caching

The concept behind Dynamic Content Caching is to store results of a first execution of a dynamically generated Web page. In this way, further requests made to the same page, will go to the Cache. Consequently, avoiding the overhead incurred by executing an application that renders output that does not change.

Zend Platform offers two ways to Content Cache files:

- Full Page Content Caching - for cases where it is possible to cache an entire output.
- Partial Page Content Caching - for cases where it is impractical or impossible to cache the entire output.

Note:

A separate tutorial has been included at the end of this guide to present Partial Page Content Caching functions and concepts (Please refer to the Tutorial: Partial and Preemptive Page Caching).

There are two caching conditions that can be applied to files:

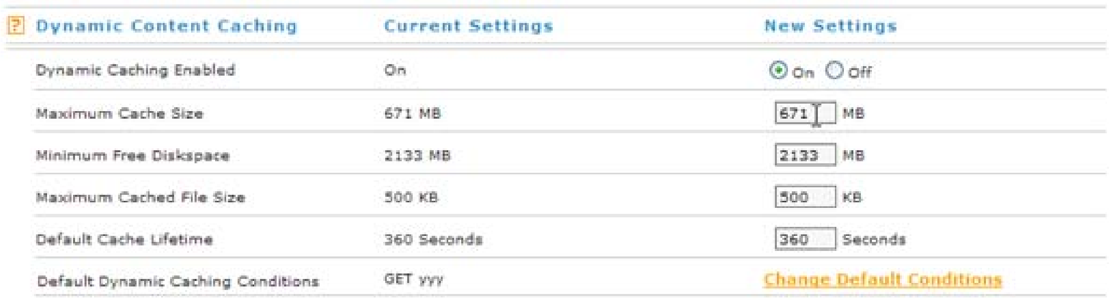
- Default Full Page Content Caching settings can be applied to all files marked as cached in: Performance | Settings and going to the Dynamic Content Caching section of the settings screen.
- Specific Full Page Content Caching configurations can be applied to specific files by going to: Performance | File View.

Full Page Content Caching

Default Full Page Content Caching settings are applied to all files marked as cached in: Performance | Settings and go to the Dynamic Content Caching section of the Settings screen.

The content caching options are as follows:

- Dynamic Content Settings
- Default Caching Conditions
- Default Dynamic Caching Condition Parameters



Dynamic Content Caching	Current Settings	New Settings
Dynamic Caching Enabled	On	[✓ On] Off
Maximum Cache Size	671 MB	671 MB
Minimum Free Diskspace	2133 MB	2133 MB
Maximum Cached File Size	500 KB	500 KB
Default Cache Lifetime	360 Seconds	360 Seconds
Default Dynamic Caching Conditions	GET yyy	Change Default Conditions

Figure 17 - Dynamic Content Caching Settings

Dynamic Content Settings

The lifetime and conditions settings in the Settings tab are default values. These settings can be modified per file or per directory in the File View workspace.

Dynamic Content Caching Settings are as follows:

- Dynamic Caching Enabled - On The Dynamic Content Caching is active and working. Off – The Dynamic Content Caching is not in use.
- Maximum Cache Size -The maximum size allocated for cache. Occasionally and for short periods of time, this value may be exceeded but only until the next time that the Cache Cleaner deletes the files that expired. Set to "0" for an unlimited cache size.
- Minimum Free Disk space - The minimal reserved free disk space required. Reaching this value will end any further caching. The caching will resume as soon as the space is greater than this value.
- Maximum Cached File Size - The maximum allowed output cache file size. An output cache file that exceeds this value will not be cached. Set to "0" for an unlimited cache size.
- Default Cache Lifetime - The lifetime, in seconds, of cached data. The data will be re-generated if the cached version is older than the expiration time.

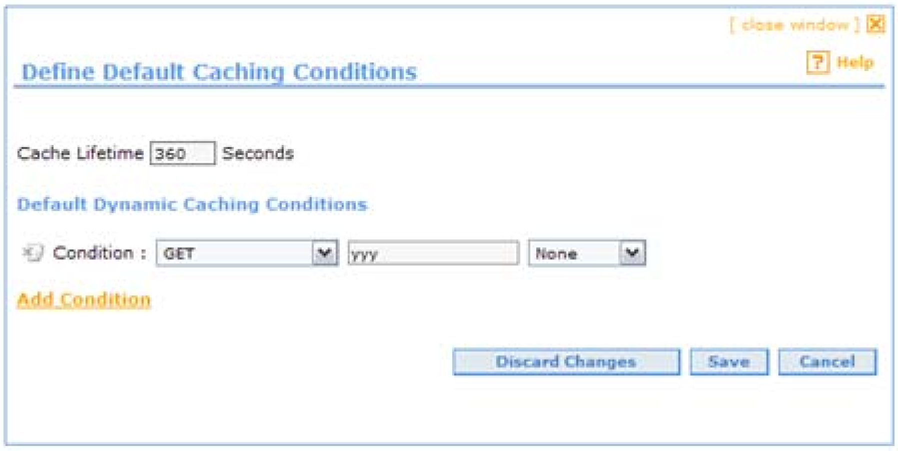
Note:
The Cache Cleaner is directly related to the directive zend_accelerator.cache_cleaner_freq that can be defined in the Configure PHP Settings screen. This directive defines when expired cache files are removed from the cache.

Default Caching Conditions

By default, Dynamic Content Caching, caches each request based on its full URL (ALLGET). You can condition the settings to be more general or more specific, as desired.

To change default caching conditions:

Go to Performance | Settings and go to the Dynamic Content Caching section of the settings screen. Select, "Change Default Conditions" to open the "Define Default Caching Conditions" dialog.



Cache Lifetime 360 Seconds

Condition : GET yyy None

Figure 18 - Define Caching Conditions Dialog

The default caching condition is ALLGET, which means that the entire GET string is used to identify a cached item. The GET string includes everything that appears after the question mark in a URL. (The ALLGET variables can be found in the $_GET PHP array as well).

The following actions and conditions can be applied to the Default Caching settings:

- To limit the ALLGET condition, select "Except" from the restrictions drop down list, to exclude a specific GET variable from the ALLGET.
- To change the ALLGET condition, select a new condition from the drop down list.
- To add another condition, click "Add Condition" and select another condition type from the list. Type the variables in the new condition row and restrict if necessary. The same condition can be used several times, each time with a different restriction.
- To remove any condition, click the delete icon next to the condition you wish to cancel.
- To change the Cache Lifetime’s duration, type the new number (in seconds).

When all configurations are completed, click "Save" to save and close the dialog. Modified settings will be immediately displayed in the Settings tab. Click “Apply Changes” and restart the Web server to activate the new caching conditions. The message will remain on the screen until the server is actually restarted.

Note:
Caching conditions may also apply to Variables stored in an Array.

Using Regular Expressions to Define Cache Conditions

Caching conditions can also be set using regular expressions to create conditions. Use the Match regexp and Dismatch regexp options to define caching conditions. These can either be case-sensitive or incase-sensitive. The regexp format is: “Unix Regular Expressions Format”.

Default Dynamic Caching Condition Parameters

The following lists and describes each of the applicable parameters.

- GET - Indicates that you have selected certain GET variables. For example, consider the URL: http://www.mysite.com/myscript.php?color=blue&size=L. When set to ALL GET, a new request for myscript.php?color=blue&size=M, will not be taken from the cache and will be regenerated. If, however, the setting is changed to GET, with the value 'color', then the two URL requests would both be taken from the same cache content, regardless of the order of the variables in the request string. (The GET variables can be found in the $_GET PHP array as well).
- COOKIE - The Cookie variable is the variable given in the HTTP cookie. (It can be found in PHP $_COOKIE array as well). By selecting a cookie variable, it will also be considered a determining factor for cache hits, in the same way that GET variables are considered.
- REQUEST - The variable is set by the GET or COOKIE methods. (Can be found in PHP $_REQUEST array as well).
- SERVER - Server variable is set as a server environment variable When selecting a server variable, (those listed in PHP $_SERVER array) it will also be used as a determining factor for cache hits, in the same way that GET variables are considered. To define a Server variable, select a variable from the list or choose Add a new variable to type in another variable.
- SESSION -The SESSION variable is useful when PHP sessions are in use. (Can be found in PHP $_SESSION array as well). Note: 1. If a script is cached using a SESSION variable and the session does not start in this script, the script will not be cached.2. If a script is cached using a SESSION variable, yet the cookies are disabled on the user side and the SESSION ID is embedded directly into the URL, the caching will not take place.
- ALLSESSION - The script depends on all of the variables present in the session. (Can be found in $_SESSION PHP array as well).

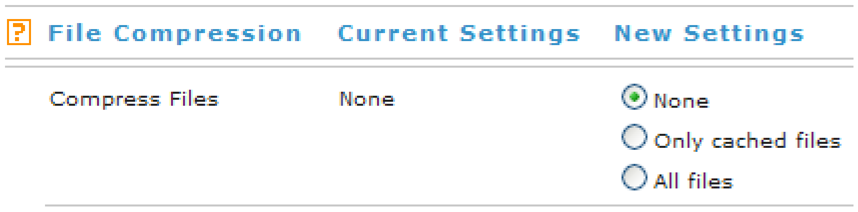
Note:
It is mandatory to choose at least one Dependency.

File Compression

In order to maintain that Cached files improve overall performance, compression settings should be defined. These settings determine which files should be compressed. The mode of compression is gzip format – if the browser supports this format (If not, the data will be transferred un-zipped).

To define compression settings:

Go to: Performance | Settings and go to the File Compression section of the settings screen. Select the file compression option that reflects your requirements.



File Compression	Current Settings	New Settings
Compress Files	None	[✓ None] Only cached files All files

Figure 19 - File Compression Settings

File compression options are as follows:

- None - File outputs are sent to the browser as is.
- Only Cached Files - Only the cached files are transferred to the browser in a gzip format—if the browser supports the format. If not, the data will be transferred un-zipped.
- All Files - Both accelerated and cached files are transferred to the browser in a gzip format—if the browser supports the format. If not, the data will be transferred un-zipped.

Recommended:

The recommended compression option is “Only Cached Files”, since the compression capabilities make use of the Dynamic Content Caching and there is no extra overhead for generating the compressed file (except for the very first time the URL is accessed.) Compressing accelerated files may cause some overhead and affect the overall performance. Use “All Files” if your main concern is improving the download time for the user.

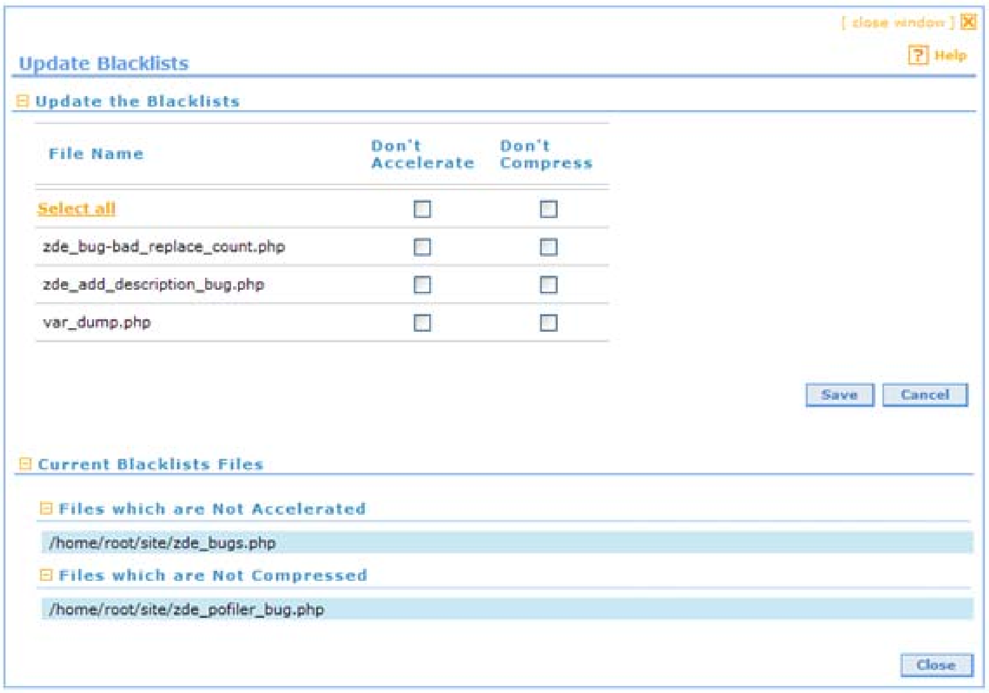
Note:
There are some instances where it is preferable to deactivate compression for select files.

To deactivate compression:

- Deactivate compression entirely – should be done if the server is set to handle compression to prevent compressing files twice and rendering them unusable or when using PHP’s compression feature zlib.
- Setting compression to "cached files only" – should be done when there is a large quantity of cached files and the rest of the files do not require compression.
- Blacklist – selectively disable compression for files do not require compression such as pictures that are already compressed or small files that do not require compression.
- Files under 1k are not compressed at all.

Blacklists

The Blacklist separates acceleration and compression settings for files. With the blacklist users can prevent files from being accelerated or compressed. The blacklist is accessed from: Performance | File View and pressing the Update Blacklist Files button.


[close window]
Help

Update Blacklists

Update the Blacklists

File Name	Don't Accelerate	Don't Compress
Select all	☐	☐
zde_bug-bad_replace_count.php	☐	☐
zde_add_description_bug.php	☐	☐
var_dump.php	☐	☐

Save Cancel

Current Blacklists Files

Files which are Not Accelerated

/home/root/site/zde_bugs.php

Files which are Not Compressed

/home/root/site/zde_pofiler_bug.php

Close

Figure 20 - Update Blacklists Dialog

This dialog has two distinct sections:

1. Update the Blacklist – for defining blacklist criteria for selected files
2. Current Blacklisted Files – for viewing current blacklist settings

To update the Blacklist:

1. Select a file or files from the File View by clicking the check box next to the file names.
2. Click "Update Blacklist".
   This will open the Update Blacklists dialog with a list containing the selected files in the dialog.
3. The following options can be applied to each file:
   - Add a file into the Acceleration Blacklist - Check the 'Don't Accelerate' box.
   - Add a file into the Compression Blacklist - Check the 'Don't Compress' box.
   - Remove a file from a blacklist - Un-check the appropriate box.
   - Add all the files to a blacklist - Check the appropriate box in the 'Select all' line at the top of the files list.

Current Blacklisted Files:

This section displays a list of files that are either not accelerated or not compressed or both. Files in the Compression Blacklist are not compressed (whether they are cached or accelerated).
To see the files in a blacklist, click on the Expand button.

Note:
Only single files (not directories) are added to the Blacklist.

Configuring the Zend Download Server (ZDS)

(This feature is currently not applicable for Windows Operating Systems)
The ZDS (Zend Download Server) is a PHP (Zend Engine) plug-in. The purpose of this plug-in is to efficiently deal with serving large, downloads. This is done to preserve website performance levels when handling large downloads that are served over the HTTP Protocol and consume bandwidth. Downloads include, Video Files, Binary Products (such as .exe and .msi files), and other large files which are served over the HTTP protocol, and can potentially limit the performance of your website.

The ZDS provides two options:

1. Configure ZDS Settings
2. Test ZDS

ZDS functions in two modes:

- Manual Mode - Calling the API function zend_send_file() from PHP scripts.
- Transparent Mode – mapping file extensions to zend_mime_types.ini

Either mode can be run separately or in conjunction. Read on to find out how to configure the ZDS to run in either mode.

Manual Mode

In Manual mode, downloads are initiated by a PHP script that uses one all-purpose PHP function call. ZDS includes the PHP function zend_send_file(filename). Calling zend_send_file() immediately starts the file download and terminates your PHP script's execution. This effectively frees up the Apache process to handle the next incoming request.
The zend_send_file() function can also serve files that are not under the Web server's document root. Furthermore, it can be used to run logical functions such as access restriction checks, before downloads are started.

Usability Example

If a download function is called my_send_file($filename), you should integrate the zend_send_file() call in the following way in your source code:

```
if (function_exists("zend_send_file")) {
    zend_send_file($filename);
} else {
    my_send_file($filename);
}
```

Alternate Method

zend_send_file can also be set to accept a second argument, the mime type of the file. This will override the default mime type setting.
The parameters are: zend_send_file(string filename[, string mime_type]) and it would be called in the following way in your source code:

```
if (function_exists("zend_send_file")) {
    zend_send_file("/path/to/file.wma", "video/my-wma-type");
} else {
    my_send_file($filename);
}
```

Note:
If the mime_type is not specified or empty, the first mime type mechanism is used.

Manual Mode Usability Notes:

Do not create any output in Manual mode, before calling zend_send_file() - neither headers nor body - as this will interfere with the HTTP download.

Once you call zend_send_file(), the script terminates, so make sure all of your business logic runs before you call this function.

Sometimes files that are not under the same document root need to be served. Therefore, It is recommended to use the full path name to the file you want to serve. This will guarantee your script will work, even if you move it from your Web server's document root.

Transparent Mode

In Transparent mode, the file types that should be downloaded via ZDS are preconfigured, by mapping these files in the configuration file of your Web server. Files greater than the min_file_size directive will be automatically served by the ZDS.

To run ZDS in Transparent mode, make sure you meet the following requirements:

- The file extensions appear in the zend_mime_types.ini file and the file is mapped to the correct mime type.
  For example: to serve .mpeg files via the ZDS, add the following line in zend_mime_types.ini:
  video/mpeg mpeg
- In your Apache Server's configuration file, map the file type to PHP.
- For example, to map all .mpeg files to the ZDS in Apache by adding the following line to the Apache Server's configuration:
  AddType application/x-httpd-php .mpeg

Usability Note for Mac OS (In case of persistent problems with this OS please contact Zend Support.)

Mac Players cannot work when the file in the URL has .php extension.

There are three suggested solutions to this issue based on different possible requirements:

1) Use the following line where $new_name is the new file name.:

header("Content-Disposition: attachment; filename={$new_name}");

2) Rename the extension to WMA (and assign the WMA extension to PHP). This will enable these files. However, by assigning the WMA extension to PHP, ZDS would automatically parse it as a download. Therefore the WMA extension should be removed from the mime types file. Please note that, files will now be delivered with default content type, which might have effect on other players.

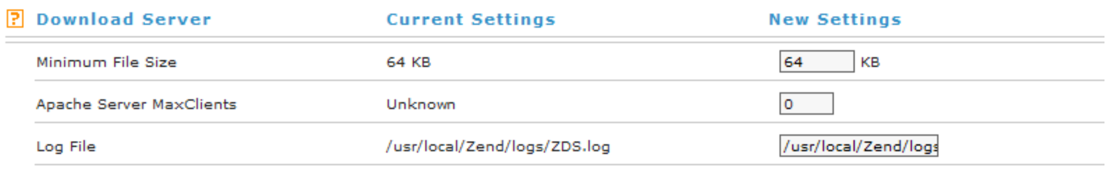
3) Add a condition to not auto-download these files, unless required by zend_send_file"

4) Add a parameter to the zend_send_file with the required mime type.

Both methods (manual mode and transparent mode) ensure that the Web application will continue to work even if, for some reason, you decide to temporarily disable the ZDS, (as long as the ZDS module was loaded).

To Configure the ZDS:

Go to Performance | Settings and go to the Zend Download Server Settings section of the Settings screen.



Download Server	Current Settings	New Settings
Minimum File Size	64 KB	64 KB
Apache Server MaxClients	Unknown	0
Log File	/usr/local/Zend/logs/ZDS.log	/usr/local/Zend/logs

Figure 21 - Zend Download Server Settings

The settings screen contains three general ZDS configuration settings:

- Minimum File Size - The minimum size of files that will be served by the ZDS. Small files need not be served by the ZDS, since performance gain is insignificant. Default: 64Kbytes.
- Server MaxClients - The testing tool (in Platform Administration) uses this value to determine your server's MaxClients. Keep this value updated to the actual number of max clients of your server.
- Log File - The name and location of the log file where the ZDS reports completed downloads. Default: <install_dir>/logs. Make sure the directory exists and that the user who starts the Web server (usually root) has "write" permission.

Server MaxClients Recommendation:
The MaxClients setting depends on your server hardware. To achieve accurate test results the server should be set between 50-150 MaxClients. The MaxClients value must be the same in the Download Server Settings and the Web server's configuration file.

These settings are applied to downloads handled in one of the two handling modes: Manual and Transparent

Testing the ZDS

Once the ZDS has been configured a test can be run to check and analyze the overall efficiency.

- The default ZDS test uses the Manual mode of operation to invoke a PHP script, which sends a file of approximately 300KB.
- The same test tool can be used to check the Transparent mode. Make sure that you correctly map the file type you are checking in your Web server's configuration file - according to the configuration instructions.

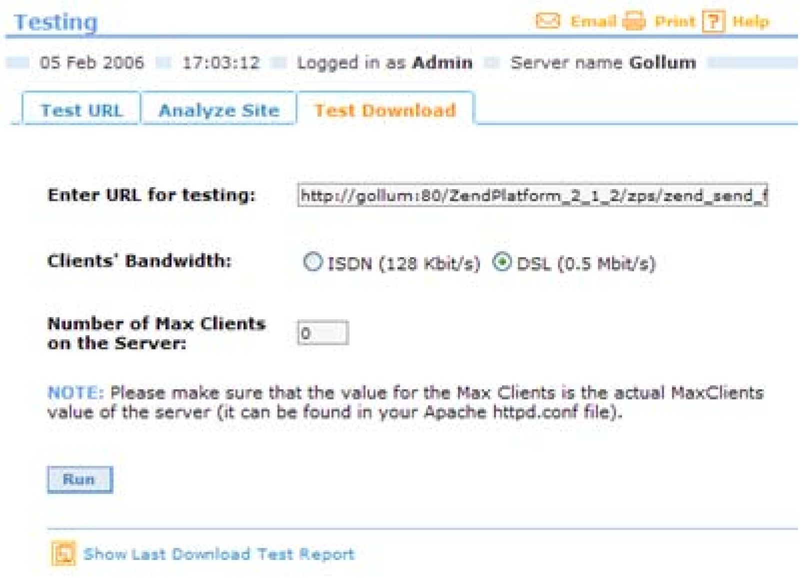
The test simulates multiple requests for a specified URL, with and without ZDS. There are three sets of tests, each test is performed twice (once with the ZDS disabled and once with the ZDS enabled). These tests differ in the number of concurrent clients that simultaneously perform requests to the server.

Note:
It is extremely hard to artificially test ZDS. The main reason is that testing it on a LAN can easily saturate your local network, and if your MaxClients is very high, Apache Benchmark (ab) may have difficulty handling the concurrency. For this reason, it is recommended to test ZDS with a relatively low MaxClients setting (e.g., 50-150) so that you don't reach any of these limits. The ZDS includes a version of ab, which was modified to support bandwidth throttling, which is used by the testing tool.

Caution:
During a test, your Web server will be fully loaded. A test can take several minutes so you should run it on a development machine or on an offline production machine.

To Test the ZDS:

Go to Performance | Testing and go to the Test Download tab.



Enter URL for testing: http://gollum:80/ZendPlatform_2_1_2/zps/zend_send_f

Clients' Bandwidth: ISDN (128 Kbit/s) [✓ DSL (0.5 Mbit/s)]

Number of Max Clients on the Server: 0

NOTE: Please make sure that the value for the Max Clients is the actual MaxClients value of the server (it can be found in your Apache httpd.conf file).

Figure 22 - Performance - Testing – Test Download Tab

The Test Download Tab consists of two sections: The test options are on the upper section and the test results appear below (after running a test or displaying the last test results).

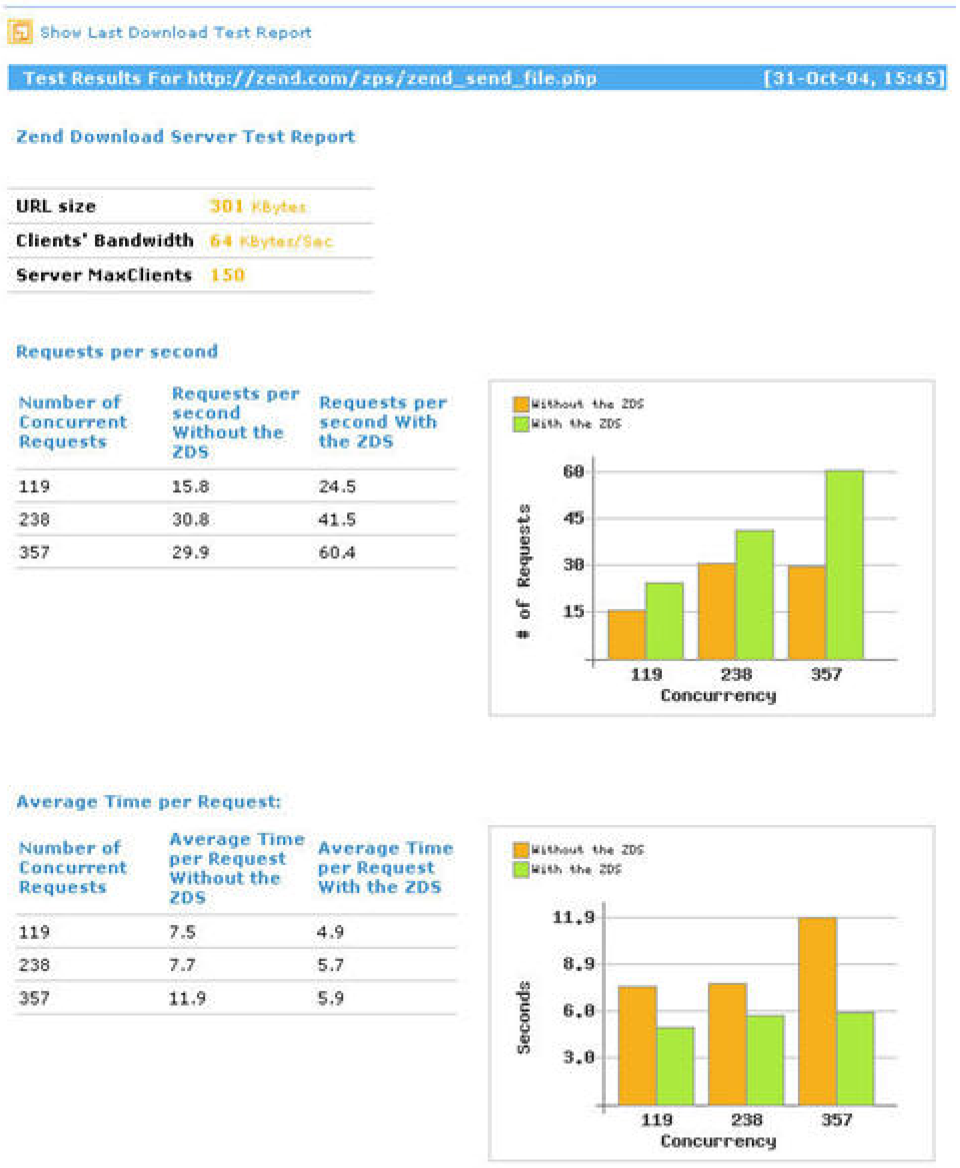
Running a Test

1. Type in the URL you want to test.
   The default test is a PHP script which uses zend_send_file() to send a 300K zip file (Testing very large files will take a very long time).
2. Choose the bandwidth limit you want to simulate for the clients. For a faster test, select a higher bandwidth (You cannot choose full bandwidth because your network card will be saturated, making the test irrelevant. The test tries to simulate a typical Internet server that has clients connected either by ISDN or DSL).
3. Enter the number of maximum clients that your server can handle.
   Use the precise value by checking the value of the MaxClients directive in your server's configuration file (The ZDS tries to identify your MaxClients value in the installation process, via the httpd.conf file, which is the default value. However, this value can be changed after the installation; and should be double-checked. Using an inaccurate MaxClients value, may not present accurate results).

4. Click Run.

Understanding Test Results

Once the tests have completed, you will see two tables and graphs with results that show Requests per Second and Average Time per Request for each test run.



Show Last Download Test Report

Test Results For http://zend.com/zps/zend_send_file.php [31-Oct-04, 15:45]

Zend Download Server Test Report

URL size	301 KBytes
Clients' Bandwidth	64 KBytes/Sec
Server MaxClients	150

Requests per second

Number of Concurrent Requests	Requests per second Without the ZDS	Requests per second With the ZDS
119	15.8	24.5
238	30.8	41.5
357	29.9	60.4

Average Time per Request:

Number of Concurrent Requests	Average Time per Request Without the ZDS	Average Time per Request With the ZDS
119	7.5	4.9
238	7.7	5.7
357	11.9	5.9

Figure 23 - Zend Download Server - Test Results

Disabling the Zend Accelerator changes the test configuration to cached scripts only.

Note:

If you do decide to run the ZDS Test on a production server, you can watch the log file to see how many concurrent jobs ZDS is handling. This indicates the number of Apache processes that would have been used if the ZDS were not installed.

Configuration Tab

IN THIS CHAPTER…

TUNNELING (COMMUNICATION SETTINGS)
CONFIGURING PREFERENCES FOR TUNNELING
CONFIGURING ZEND STUDIO TUNNELING SETTINGS
ON DEMAND CONNECTION
SERVER SETTINGS
PHP CONFIGURATION
PHP CONFIGURATION

Tunneling (Communication Settings)

Communication with Zend Studio facilitates the integration that combines Zend Platform's event reporting capabilities with Zend Studio's editing, debugging and profiling features.
This integration provides an efficient way for managing the different stages of the development lifecycle. Zend Platform's PHP Intelligence inspects performance and Zend Studio debugs, profiles and provides a means for resolving issues and deploying changes.

There are two modes of communication with Zend Studio that accommodate different requirements:

- Tunneling (*Auto Detect Mode) – creates a secure communication tunnel with Zend Studio (IDE) that keeps a persistent connection with the designated communication port. This mode of communication is the recommended mode of communication. It is also responsible for solving communication problems that arise when Zend Studio is behind a security devices such as a Firewall or NAT.
- On-Demand Communication – creates a connection on demand. Selecting to edit, debug or profile code opens a connection that is closed once the action is completed. This requires defining the port and IP for direct communication with Zend Studio.

*This feature is currently not applicable for Windows Operating Systems.

Choosing a mode of communication depends on how your environment is set-up. If there are security devices and there is no limitation to keeping, a persistent connection use the Tunneling option - the preferred mode of communication. However, if for some reason it is not possible to keep the port connection open at all times, use the On-Demand Communication option, this option should be used when Zend Platform and Zend Studio are not separated by any security mediation devices.
To Setup the Integration with the Zend Studio, there are several tasks that need to be performed depending on the type of connection you want to establish:

- Configuring Zend Central Preferences for Tunneling Communication
- Configuring Zend Studio's Tunneling settings
- Configuring Zend Central Preferences for On-Demand Communication
- Configuring the Studio Server settings for Tunneling and On-Demand Communication
- Debugger Tunneling Port Limits

Configuring Preferences for Tunneling

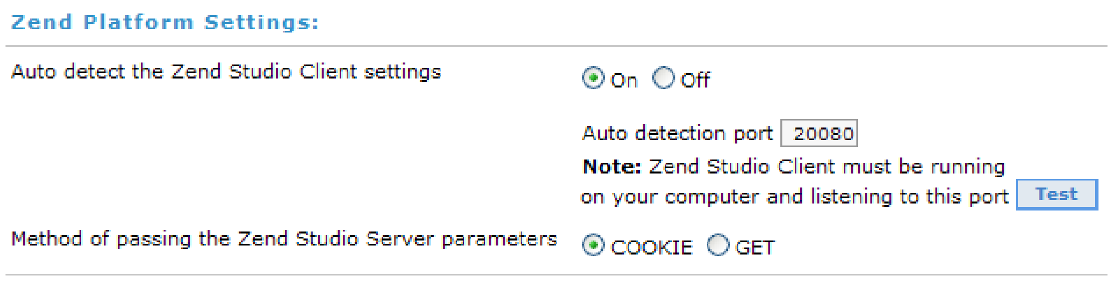
Communication Tunnel

This persistent connection operates even when separated by a firewall. The advantage of this method is that it is possible to use the Zend Studio Integration on several nodes at once. For example, debugging an entire cluster of machines behind a load-balancer over a single debugger connection to Zend Platform's Studio Server component.

The technology is based on two functional elements:

- The Zend Studio that includes an internal Web server that listens on the local host on a specific Auto Detection port.
- Zend Platform auto-evaluates Zend Studio's Auto Detection port, by evaluating Zend Studio's settings. These are the Tunnel Settings that are defined in Zend Platform.

To establish a communication tunnel between Zend Platform and Zend Studio:
Go to: Platform | Preferences.



Zend Platform Settings:

Auto detect the Zend Studio Client settings	[✓ On] Off Auto detection port 20080 **Note:** Zend Studio Client must be running on your computer and listening to this port
Method of passing the Zend Studio Server parameters	[✓ COOKIE] GET

Figure 24 - Communication Tunnel Method

In the Zend Platform Settings section, enable Auto detect the Zend Studio Settings by clicking On. This informs Zend Platform of the method of connection to Zend Studio. The relevant options/fields for configuring Tunneling are as described below:

- Auto Detection Port - Indicates that Zend Studio will listen to the local host on the signified Auto Detection port - Default=20080. Leave empty to automatically identify the IP from the browser.
- Test - Verifies if Zend Studio is listening to this port. This test should only be run when Zend Studio is running.
- Method of passing the Zend Studio Server parameters - Defines the means for passing communication parameters from the Zend Platform to Zend Studio. Choose COOKIE or GET method.

Configure communication as follows:

1. Use the "Test" option to verify that Zend Studio's Broadcast Port is set to the same port number as Platform 's Auto Detection Port.
2. Select a method of passing Zend Studio parameters.
3. Click "Save".

Note:
The default method is Cookie, and it is recommended that you use the default. Platform supports a Get method as well that can be used if experiencing problems with Cookies.

Zend Platform establishes a communication tunnel with Zend Studio based on the default detection port.

Note:
Tunneling to Debug and Profile code with the Zend Studio is only possible if Zend Studio is open and running and Zend Studio's Tunneling settings are properly configured in Zend Platform and Zend Studio.

Configuring Zend Studio Tunneling Settings

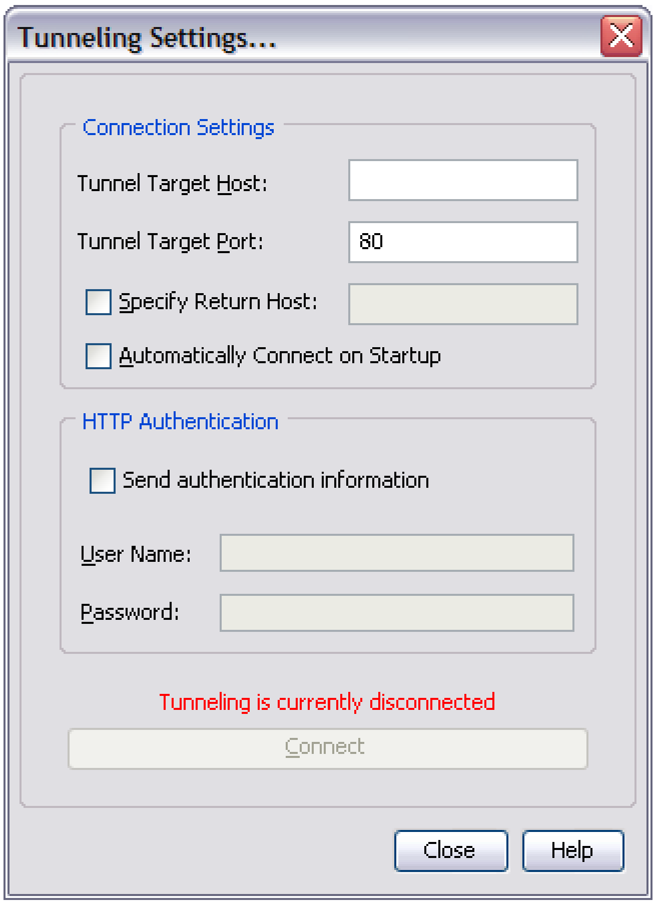
This feature enables Zend Platform users to connect to Zend Studio to edit, debug and profile Event source code using the Zend Studio IDE.

Configuring the Communication Tunnel in Zend Studio

To configure Tunneling Settings for Studio:

1. Open the Tunneling dialog: Tools | Tunneling Settings:
2. Define values for the following settings:
    - Tunnel Target Host - Address of the Web server on which the debugger resides.
    - Tunnel Target Port - Port of the Web server on which the debugger resides.
    - Specify Return Host - When enabled, this should contain the address of the main server in the cluster.
    - Automatically Connect on Startup - Enables the communication tunnel when Zend Studio starts up.
    - HTTP Authentication -Zend Studio Tunneling supports HTTP authentication. This enables users to send http authentication information (user name, password) together with the header sent to the server. Therefore, you can specify that tunneling to a server will require authentication, and improve security by adding information in the following fields:
        - Send Authentication Information - Use this option when working with a Web server that requires HTTP authentication. Zend Studio sends the authentication information in the header.
          Note: This assumes the user account is set up on the Web server.
        - User Name - User name as defined on the Web server.
        - Password - User password as defined on the Web server. Note: Whenever you use the debugger, the server will use the User Name and Password specified here.
3. Click "Connect" for Zend Studio to connect to the Tunnel Target Host over the specified port.

Note:
Information in Zend Studio's Debug Preferences, must match the information in Zend Studio's Tunneling Settings for tunneling to work.


Tunneling Settings...

Connection Settings

Tunnel Target Host:

Tunnel Target Port: 80

☐ Specify Return Host:

☐ Automatically Connect on Startup

HTTP Authentication

☐ Send authentication information

User Name:

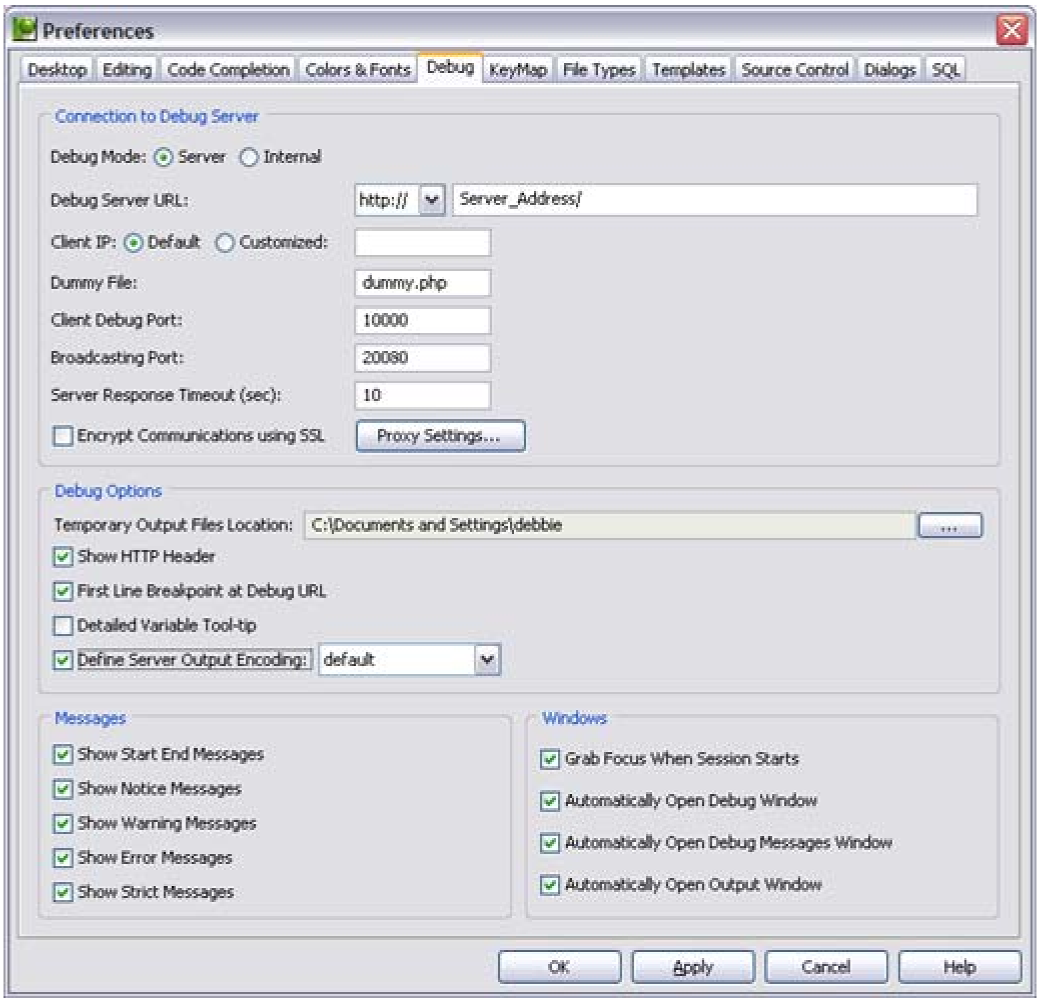
Password:

Tunneling is currently disconnected

Figure 25 - Zend Studio Tunneling Configuration

Debug Preferences

To view debug preferences to ensure that the information is suits the Tunneling Preferences go in Zend Studio to: Tools | Preferences | Debug:



Connection to Debug Server

Debug Mode: [✓ Server] Internal

Debug Server URL: http:// Server_Address/

Client IP: [✓ Default] Customized:

Dummy File: dummy.php

Client Debug Port: 10000

Broadcasting Port: 20080

Server Response Timeout (sec): 10

☐ Encrypt Communications using SSL

Debug Options

Temporary Output Files Location: C:\Documents and Settings\debbie

☑ Show HTTP Header

☑ First Line Breakpoint at Debug URL

☐ Detailed Variable Tool-tip

☑ Define Server Output Encoding: default

Messages

☑ Show Start End Messages

☑ Show Notice Messages

☑ Show Warning Messages

☑ Show Error Messages

☑ Show Strict Messages

Windows

☑ Grab Focus When Session Starts

☑ Automatically Open Debug Window

☑ Automatically Open Debug Messages Window

☑ Automatically Open Output Window

Figure 26 - Zend Studio Debug Preferences

In the section called "Connection to debug server", make sure the following settings are configured:

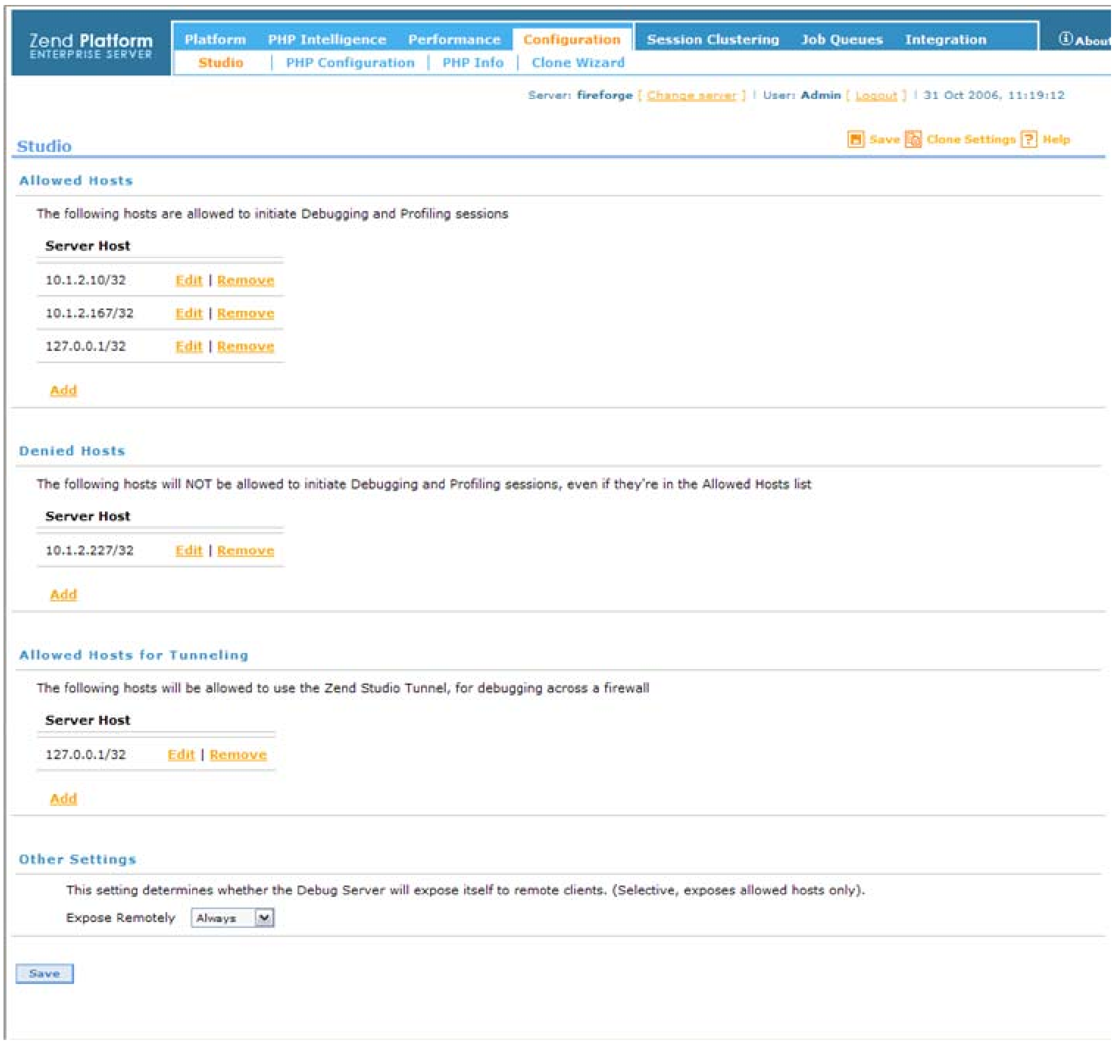
- Debug Mode - Sets the Debugger to Internal or Remote. Select remote for integrating with Zend Platform
- Debugger Server URL - The IP or URL of the host that runs the Zend Debug Server. To check the connection to the debug server, select the Check Debug Server Connection, from the Debug menu. If the test fails, check the list of common problems in Appendix A – Troubleshooting Zend Platform.
- Client IP - Set the IP address of Zend Studio's host machine.
- Client Debug Port - Set the port number for communication with the Zend Debug Server
- Broadcasting Port - Zend Studio's communication tunnel is implemented via a persistent broadcasting port that broadcasts information about tunneling to Zend Platform. Specify the port number in this field.

- Dummy File - This file used during the debug process for storing interim information.
- Server Response Timeout - The amount of time allowed for a server response. If no response is received within this time, a notification will be generated to inform you that the Server is not responding.
- Encrypt Communications using SSL - To enable SSL encryption make sure this option is selected here and in Zend Platform's Zend Central | Preferences.

Studio Settings

Studio is Zend Platform's embedded integration with Zend Studio's debugger. The Studio component is part of the Zend Platform node installation, ensuring that an instance of the debugger resides on every node.
Studio settings provide a way for allowing or denying accessibility to a selected server or to a selection of servers (using a Net Mask which implements Wildcards on IP addresses).

To access Studio Settings go to: Configuration | Studio.



Server: fireforge [Change server] | User: Admin [Logout] | 31 Oct 2006, 11:19:12

Studio

Allowed Hosts

The following hosts are allowed to initiate Debugging and Profiling sessions

Server Host	
10.1.2.10/32	Edit | Remove
10.1.2.167/32	Edit | Remove
127.0.0.1/32	Edit | Remove

Add

Denied Hosts

The following hosts will NOT be allowed to initiate Debugging and Profiling sessions, even if they're in the Allowed Hosts list

Server Host	
10.1.2.227/32	Edit | Remove

Add

Allowed Hosts for Tunneling

The following hosts will be allowed to use the Zend Studio Tunnel, for debugging across a firewall

Server Host	
127.0.0.1/32	Edit | Remove

Add

Other Settings

This setting determines whether the Debug Server will expose itself to remote clients. (Selective, exposes allowed hosts only).

Expose Remotely: Always

Save

Figure 27 - Studio Server Settings

The Studio Settings screen displays the Studio settings for the selected node (The server is selected in the server tree).

There are four settings categories:

- Allowed Hosts - These are the hosts that are allowed to initiate debugging and profiling sessions.
- Denied Hosts - These are the hosts that are not allowed to initiate debugging and profiling sessions, even if they are on the Allowed Hosts list.
- Allowed Hosts for Tunneling - These are the hosts that are allowed to use this node for tunneling. The Zend Studio Tunnel is used for debugging PHP code across a firewall to use the integration with the Zend Studio. **Note:** Tunneling is not supported in Windows.
- Other Settings - These are additional settings supported by Zend Platform. Currently, “Expose Remotely” is the only setting in this category. This setting determines whether the Debug Server will expose itself to remote clients. This is required if you want the Zend Studio Browser Toolbar to automatically detect pages that can be debugged.

The Settings screen is for Adding, Editing or Removing a host from the Allowed Hosts, Denied Hosts, or Allowed Hosts for Tunneling categories. You can also assign a value (Always, Selective, or Never) to the Expose Remotely setting for the selected node.

"Expose Remotely" settings:

- Always - Will expose all hosts
- Selective - Only exposes the hosts in the allowed host list
- Never - Will not expose any host

To access the Studio Server tab:

1. Go to Configuration |Studio.
2. From the Server Tree, select the server you wish to configure (or whose settings you wish to view).
3. Click "Select "to open the Studio Server tab for the selected node.

To change or add a host to allow or deny tunneling:

1. Go to Configuration | Studio.
2. Click "Add" - for example, in the Allowed Hosts category to open the Add New Allowed Host dialog box opens.
3. Enter the Settings for the new Allowed Host.
4. Click "Apply" - A new Host will appear in the Allowed Hosts category on the Server Settings screen.
5. To edit or remove a Host - for example, in the Allowed Hosts category, click "Edit" or" Remove" (To the right of the Server Host you wish to edit).

You will be prompted to confirm your new settings click "Save" to save the settings to the database. To set the Expose Remotely setting for the selected node, select a value, Always, Selective, or Never, from the drop-down list provided.

Net Masks:

The Net Mask option is 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 deny the following IPs: 24 (10.1.3. *) all IP addresses beginning with 10.1.3 will be denied access to the Zend Studio Server (i.e. Integration with Zend Studio will not be permitted for these IP addresses).

Debugger Tunneling Port Limits

To ensure persistent connections while using Tunneling over firewalls for debugging event information in Zend Platform or debugging scripts edited in Zend Studio, you can modify the following zend.ini directives that define a port range.

The directives are:

- "zend_debugger.tunnel_min_port"
  Description: Minimal possible value of Debugger tunneling port
  Default value: 1024
- "zend_debugger.tunnel_max_port"
  Description: Maximal possible value of Debugger tunneling port
  Default value: 65535

Note:
The Debugger uses the default values either when the directives are not present in the Zend ini, or if one of them is invalid. If the directives are not present, the Debugger will revert to random port allocation and not from a predefined range of ports.

These directives define a port range for Tunneling. While Tunneling, the Debugger will try to locate a free port starting from the minimum value defined in the "zend_debugger.tunnel_min_port" directive, but not above the maximal value defined in the "zend_debugger.tunnel_max_port" directive. Another consideration when defining a port range is, to ensure the amount of ports opened correspond to the amount of possible debugger connections that may occur i.e. the range should reflect the amount of Zend Studio's you have in your organization.
In parallel, the System Administrator must ensure the proper firewall policies [rules] are set to allow communication via the selected ports, in order for the tunneling to work.
The tunnel server, and not the debugger, uses these tunnel settings. The debugger will still use random ports for debugging.

Possible Error Message:
Could not find a free TCP port for tunneling. Please re-adjust the 'zend_debugger.tunnel_min_port' and 'zend_debugger.tunnel_max_port' directives in the php.ini file.
This means the Debugger could not find a free port to establish a communication tunnel, make sure you have defined an adequate port range in the directives. If the problem persists, consider checking the firewall policies.

PHP Configuration

The PHP Settings screen is the configuration tool for customizing PHP and Zend products, by modifying directives and extensions in the php.ini and zend.ini files.
Configuration options are separated by type (Directives and Extensions) in expandable lists. The [+/-] signs indicate if there are more options related to that list item or not.
Clicking on the Plus Icon [+] will expand the lists to expose the different options and where applicable, input fields are added to change an option's value. Alternatively, clicking the Minus Icon [-] will contract the list leaving only the option type visible.
Most of the directives can be viewed and modified. Directives that should not be changed under any circumstances are disabled (grayed out).
The directives and extensions sections are separated into two as follows:

- PHP Section - The PHP section reflects the exact content of the php.ini on the selected server. Changing directives in this section will change the php.ini settings.
- ZEND Section - This section contains Zend Product directives. Changes to this section are applied to the zend.ini or the php.ini according to their origin. Some of these directives do not require restarting the server. An indication will be given regarding which directives require restart and which changes will be automatically applied. (See Appendix F for a complete list of the directives and their settings)

Note:
Most of the directives can be viewed and modified. Directives that should not be changed under any circumstances are disabled (grayed out). Directives that are commented in the php.ini are only viewed in the PHP Configuration screen when the comment is removed. As an added precaution, changes that will be done to the php.ini will require entering your php.ini password, as defined in the 'zend_gui_password' directive (the password is MD5'ed). To change the password go to: Zend Central | Preferences.

Finding a Directive / Extension

Search for a directive or an extension by using the "Quick Filter" field. The "Quick Filter" allows to search for a directive or an extension that contains a certain word or combination of letters in its name or description.

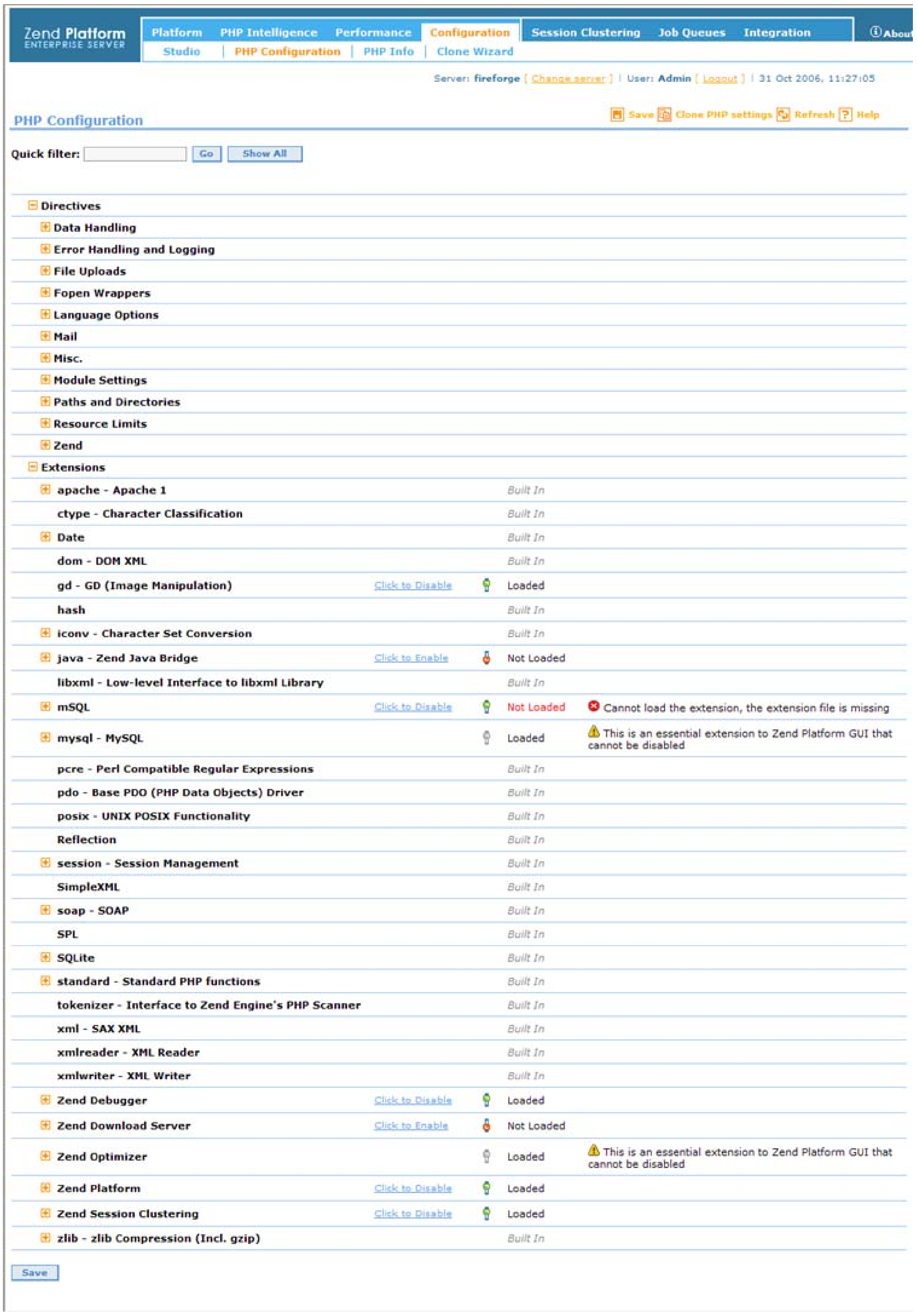
To find a Directive or extension:

Go to Configuration | PHP Configuration and enter the directive into the Quick Filter Field.

Configuring Settings for a Server (Node)

To configure Settings:

1. Go to Configuration | PHP Configuration or use the shortcut from Platform | Dashboard | Configure PHP Settings.
2. Use the + symbol to select and expand the PHP entry you wish to edit or use the "Quick Filter" to search for a directive or an extension that contains a certain word or combination of letters in its name or description.
   In both cases the expanded line will open with an active field for editing directive’s parameters.
3. In the Value column, enter a new setting in the editable field provided.
   Changed setting's backgrounds will automatically change to a darker shade of blue indicating that it has been modified and will stay that way until the "Save Settings" button is clicked.
4. Click "Save Settings".
   The new settings will be registered to the php.ini configuration file (you may be asked for your php.ini password for critical changes).



Server: fireforge [Change server] | User: Admin [Logout] | 31 Oct 2006, 11:27:05

PHP Configuration

⊟ Directives
- ⊞ Data Handling
- ⊞ Error Handling and Logging
- ⊞ File Uploads
- ⊞ Fopen Wrappers
- ⊞ Language Options
- ⊞ Mail
- ⊞ Misc.
- ⊞ Module Settings
- ⊞ Paths and Directories
- ⊞ Resource Limits
- ⊞ Zend

⊟ Extensions

⊞ apache - Apache 1		Built In	
ctype - Character Classification		Built In	
⊞ Date		Built In	
dom - DOM XML		Built In	
gd - GD (Image Manipulation)	Click to Disable	Loaded	
hash		Built In	
⊞ iconv - Character Set Conversion		Built In	
⊞ java - Zend Java Bridge	Click to Enable	Not Loaded	
libxml - Low-level Interface to libxml Library		Built In	
⊞ mSQL	Click to Disable	Not Loaded	Cannot load the extension, the extension file is missing
⊞ mysql - MySQL		Loaded	This is an essential extension to Zend Platform GUI that cannot be disabled
pcre - Perl Compatible Regular Expressions		Built In	
pdo - Base PDO (PHP Data Objects) Driver		Built In	
posix - UNIX POSIX Functionality		Built In	
Reflection		Built In	
⊞ session - Session Management		Built In	
SimpleXML		Built In	
⊞ soap - SOAP		Built In	
SPL		Built In	
⊞ SQLite		Built In	
⊞ standard - Standard PHP functions		Built In	
tokenizer - Interface to Zend Engine's PHP Scanner		Built In	
xml - SAX XML		Built In	
xmlreader - XML Reader		Built In	
xmlwriter - XML Writer		Built In	
⊞ Zend Debugger	Click to Disable	Loaded	
⊞ Zend Download Server	Click to Enable	Not Loaded	
⊞ Zend Optimizer		Loaded	This is an essential extension to Zend Platform GUI that cannot be disabled
⊞ Zend Platform	Click to Disable	Loaded	
⊞ Zend Session Clustering	Click to Disable	Loaded	
⊞ zlib - zlib Compression (Incl. gzip)		Built In	

Figure 28 - PHP Configuration

Users and Groups

IN THIS CHAPTER…

USER MANAGEMENT
ADDING AND EDITING USERS
ADDING AND EDITING GROUPS
FILTERING EVENT TRIGGERS
USER SETTINGS
PASSWORD ADMINISTRATION

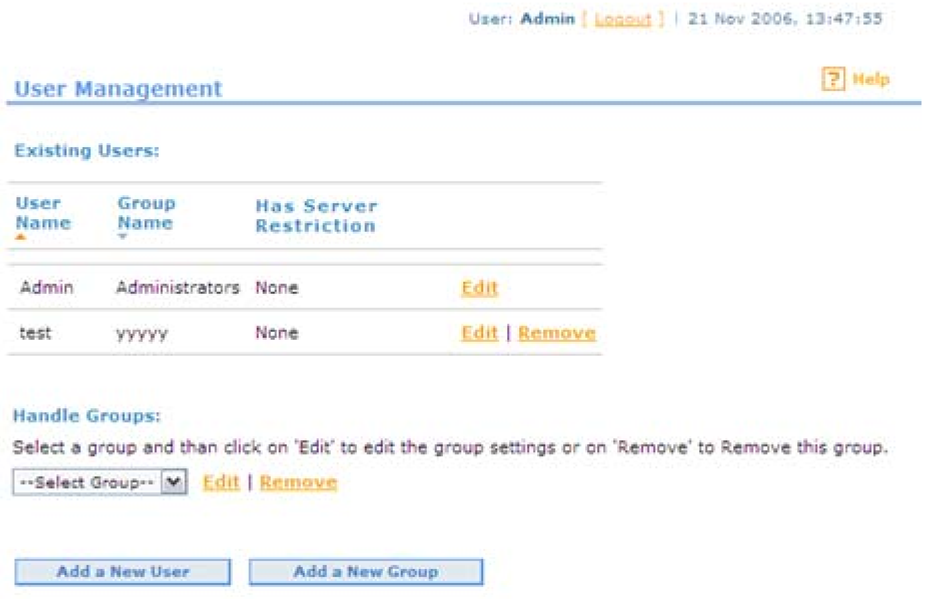
User Management

Granting different levels of permissions to different users provides a means for controlling actions performed in the environment and enforcing work procedures. This is the last step to customizing Zend Platform to suit your working environment.
Zend Central's User Management tab includes Platform's multiple users functionality. This feature set allows different users to login to Platform.

Each user has a set of permissions that are defined by the system administrator that determine:

- Data the user is allowed to view
- Zend products the user is allowed to access
- Actions the user is allowed to perform

The User Management workspace displays information about the users currently defined in the system. It also provides shortcuts to the User Management functions supported by Zend Platform.


User: Admin [Logout] | 21 Nov 2006, 13:47:55

User Management

Existing Users:

User Name	Group Name	Has Server Restriction	
Admin	Administrators	None	Edit
test	yyyyy	None	Edit | Remove

Handle Groups:

Select a group and than click on 'Edit' to edit the group settings or on 'Remove' to Remove this group.

Figure 29 - User Management

The information fields and functions that make up the User Information workspace are as follows:

- User Name - Displays the name of the User who is currently logged in to Zend Platform.
- Existing Users - A list of users currently defined in the system.
- Group Name - Users are defined within the system as belonging to a particular group—and not as independent entities. This is the name of the Permission Group to which the User belongs.
- Has Server Restriction - The specific user is restricted from performing certain actions on a specific server.
- Handle Groups - Selects the group whose attributes you wish to edit or remove entirely.
- Edit/Remove - Allows you to edit or remove entirely the settings for a specific User of a specific Group.
- Add a New User - A shortcut to the Add a New User Wizard.
- Add a New Group - A shortcut to the Add a New Group Wizard.

Adding and Editing Users

Zend Platform allows you to Add/Edit Zend Platform users under the following conditions:

1. The master user adding new users must have administrative level permissions in the system.
2. New users must be added to an existing group. Users are defined as belonging to a particular group - not as independent entities.

Adding a User

Zend Platform allows Administrator-level users to create new users.

To add a new user:

1. Click the Add a New User button in the lower left corner of the User Management workspace.
   The Add a New User wizard opens.
2. In the Add New User wizard, define the following General User Settings:
   - User Name - Enter a User Name that the User will use when logging in to Platform Administration.
   - Password - Enter a Password that the User will use when logging in to Platform Administration.
     Platform Administration Passwords, may contain, between 4-16 characters including the following: the alphanumeric characters 'a' through 'z', 'A' through 'Z' and '0' through '9' and the special characters (-) dash, (_) underscore and (.) period.
   - Confirm Password - Confirm the Password.
   - Permissions Group - Select the Permissions Group to which the New User will be assigned from the list of Permissions Groups that are currently defined in the system.
3. Click Next to go to the second step.
4. In the Add New User wizard (2) Select the servers that you wish to allow the New User to access.
5. Click Finish.
   The new user will be created in the database, with the defined permissions.

Note:
Added/Edited users can only view and edit events for the servers the User has permission to access. Conversely, servers denied to the User will not appear in the server tree when the User logs in to the system.

Editing a User

Zend Platform allows administrator-level users to edit the preferences for any User currently defined in the system. Non-administrator users can use this option to modify their password.

To edit user preferences:

1. Click the Edit button to the right of the user whose preferences you wish to edit.
   The Edit User Wizard opens with the user's name appearing in the User Name field.
2. In Edit User wizard – Step 1, define the following General User Settings:
   - Password - Enter a Password only if you wish to change the Password for that User; otherwise leave the Password field empty.
   - Confirm Password - Confirm the Password only if you are changing the Password for the User.
   - Permissions Group - Select the Permissions Group to which the User will be assigned from the list of Permissions Groups that are currently defined in the system.
3. Click Next to go to the second step.
4. In the Edit User wizard (2) screen, select the servers that you wish to allow the User to access.
5. Click Finish.
   The changes will be applied to the User settings and saved in the database.

Adding and Editing Groups

Add/Edit a Group

Zend Platform allows Administrator-level users to create new groups and define user access permissions.

To create a New Group:

1. Click the Add a New Group button in the lower left corner of the User Management workspace to open the Add a New Group dialog.
2. In the Add New Group dialog, select the preferences and permissions to assign to the New Group from the table.
3. Enter a name for the New Group.

Note:
Remember to click Save to save changes or create a Group in the database and register its settings.

The following is a list of group preferences and their definitions:

- Group Name - Enter a name for the New Group
- Delete an event - Enables the Delete Event option, for deleting events in the Event Details screen and the Event Window.
- Ignore an event - Enables the Ignore Event option, for ignoring events in the Event Details screen and the Event List Window.

- Close an event - Enables the Close Event option, for closing events in the Event Details screen and the Event List Window.
- Reopen an event - Enables the Reopen Event option, for reopening events in the Event Details screen.
- Preserve an Event - Enables the Preserve Event option, for preserving (saving) events in the Event Details screen and the Event List Window.
- See the event context data in Event Details - Allows users belonging to group to view the event internal data (variables, included files) from the Event Details screen.
- See the event source code in Event Details - Allows users to view the event source code in the Event Details embedded viewer.
- Use the Zend Studio Integration - Allows users to view, profile and debug event source code in Zend Studio.
- Changing Zend Platform Settings (in the Preferences) - Allows users to change preferences in Platform | Preferences.
- Configure and change the Event Actions - Allows users to configure and change Event Actions from the Event Actions screen.
- Configure and change the Event Action Rules - Allows users to configure and change Action Rules from the Define Action Types/Rules screen.
- Manage Server - Allows users to manage servers from the Manage Cluster screen.
- Manage VHosts - Allows users to manage Virtual Hosts from the Manage Cluster screen.
- Manage Groups - Allows users to add and change group settings from the Manage Clusters screen.
- Update all Information - Allows users to update server data.
- Use Support Tool - Allows users belonging to the group to access the Support Tool from the Support Tool link.
- Go into the Event Triggers Section - Allows users to configure and change Event Triggers from the Event Triggers screen.
- Configure PHP settings - Allows users to configure and change PHP settings from the Configure PHP Settings screen.
- Go into the Performance section of nodes - Allows users to access the Performance Tab for nodes.
- Go to the Configuration section of nodes - Allows users to access the Configuration Tab for nodes.
- Go in the Integration section of nodes - Allows users to access the Integration (Java Bridge, BIRT Reports) Tab for nodes.
- Go into the Queue Section - Allows users to access the Job Queues Tab.
- Manage Licenses - load and edit licenses.
- Use Web Services - Use the web Services API

User Settings

User settings are retained in the system in several ways:

- User Group settings are stored in the configuration database.
- Platform's User Management Tab remembers each user's last settings. The user's last settings automatically populate the component fields, when opening any of the sub-screens and dialog boxes that make up the User Management Tab.

Passwords

Password Structure

Platform Administration Passwords, may contain, between 4-16 characters including the following: the alphanumeric characters 'a' through 'z', 'A' through 'Z' and '0' through '9' and the special characters (-) dash, (_) underscore and (.) period

System Passwords

There are three kinds of passwords:

1. The admin user Platform Administration account
2. The central registration password (you need this password to register a new node)
3. Each node ini_modifier/Platform Administration password

Password Specifications

- The admin user password can be changed from: Dashboard |Preferences.
- The change_zend_gui_password.sh script changes the admin account and the Zend Central registration password.
- If you have a node/s on the central machine, then 2+3 are the same.
- If you install a node-only, then will change_zend_gui_password.sh not be installed and you will not be able to change the ini_modifier password. It will be permanently set to be the same as the central registration password at the time of the registration.

Licenses

IN THIS CHAPTER…

LICENSE MANAGEMENT
ABOUT ZEND PLATFORM LICENSES
MANAGING LICENSES
ACQUIRING A LICENSE

License Management

The License Management tab (Platform | License Management) allows you to manage licenses and view their status. the License Management tab is a central management tab for all the nodes governed by the Central Server.

About Zend Platform Licenses

There are four different types of licenses available for Zend Platform:

- Production-Enterprise: The Enterprise Server includes the performance management features along with Job management and Integration tools for Multi-Cluster enterprises.
- Production-Performance Management: The Performance Management Server includes the Platform, PHP Intelligence, Performance and Configuration tabs. This version is suited for production environments that can benefit from the information collected in events and the performance optimization tools.
- Development-Enterprise: The Enterprise Server is a full featured development version. This version includes features that can assist in the development lifecycle (PHP Intelligence and Performance) and provide an environment suitable for developing multi-cluster enterprise web applications (integration with Zend Studio, Actuate Reports etc.).This license is suitable for development environments and licenses a single Central Server (without additional nodes).
- Trial: A 30 day fully featured evaluation version of Zend Platform. After the 30 days, only the Server functionality for remote debugging will continue to work without a license.

Managing Licenses

The Manage License screen, displays a summary of all the licenses by type and if they are Active (valid) or Expired. The summary includes details about the number of licenses by type and total quantity of licenses. Licenses that have errors in them (for example: a corrupted license file) are included in the Expired license summary.

Below the License Summary is a detailed table that displays the details of each license, per server. The actions that can be performed through the table are: sort, acquire license and update information.

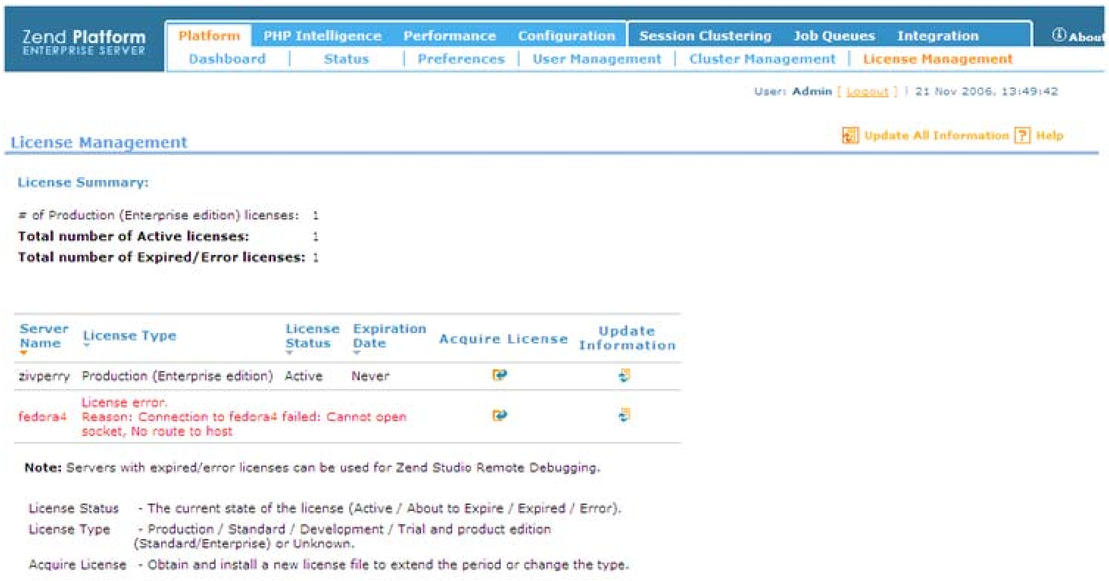
Sort: rearranges the licenses displayed in the table, by a selected column. To do so, click the column heading and the contents of the table will be automatically sorted by server name, type, status or expiration date.

Acquire License: This option is used to obtain and install license files. This option should be used to activate new licenses for each server that appears in the list. Once the license has expired use this option again to update the license.

Update Information: Updates information for a specific node to the Central Database.

To manage your licenses:

Go to Platform | License Management. The License Management screen will appear.



License Management

License Summary:

# of Production (Enterprise edition) licenses: 1
Total number of Active licenses: 1
Total number of Expired/Error licenses: 1

Server Name	License Type	License Status	Expiration Date	Acquire License	Update Information
zivperry	Production (Enterprise edition)	Active	Never		
fedora4	License error. Reason: Connection to fedora4 failed: Cannot open socket, No route to host				

Note: Servers with expired/error licenses can be used for Zend Studio Remote Debugging.

License Status	- The current state of the license (Active / About to Expire / Expired / Error).
License Type	- Production / Standard / Development / Trial and product edition (Standard/Enterprise) or Unknown.
Acquire License	- Obtain and install a new license file to extend the period or change the type.

Figure 30 - License Management Tab

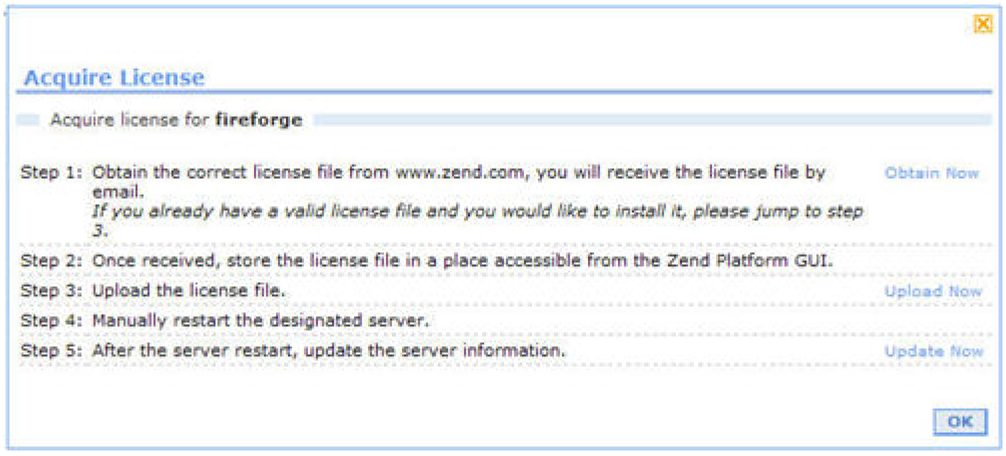
This screen displays the names of all servers (servers that are currently available and servers that were once connected), related statistics, license type and Zend Platform license status.

From this screen you can:

- Update All Information: Located in the top right corner of the screen, this button reads the license data from all the nodes and updates the Central Database. This action has to be done after making changes to the node licenses. This button should be used when you need to update all the nodes. If you need to update only a specific node, use the Update Information option.
- Update Information: Located in the license details table, this button reads the license data from a selected node and updates the Central Database. This action has to be done after making changes to the node's license. This button should be used when you need to update a selected node. If you need to update all the nodes, use the Update all Information option.
- Acquire License: obtain and attach a license file to a specific server.

Acquiring a License

1. To acquire a valid license for a specific server, click on the "Acquire" icon corresponding to the server. The "Acquire License Wizard" will appear.



Acquire License

Acquire license for **fireforge**

Step 1:	Obtain the correct license file from www.zend.com, you will receive the license file by email. *If you already have a valid license file and you would like to install it, please jump to step 3.*	Obtain Now
Step 2:	Once received, store the license file in a place accessible from the Zend Platform GUI.	
Step 3:	Upload the license file.	Upload Now
Step 4:	Manually restart the designated server.	
Step 5:	After the server restart, update the server information.	Update Now

Figure 31 - Acquire License

2. The Acquire License screen includes 5 steps, follow the instructions in steps 1-3 in order to obtain, store and upload your license file.
3. Once the license file is put into place follow step 4 and restart the designated server (if the license is for a node the designated node should be restarted.
4. After restarting the server use step 5 to update the server with the new license's information.
5. The Server's status in the table should change to show the uploaded license type, if this does not happen, open a ticket with support at: http://www.zend.com/support.

This completes the Administration and Configuration chapter of the User Guide. In this section we have described the different configuration and administration tasks that can be done. In the next chapter we will describe how to implement Zend Platform in the working environment by using PHP Intelligence to create a Problem Resolution lifecycle.
Please refer to “Appendix B – Configuration Check List” to read/print the Zend Platform Configuration Check List that summarizes the configuration tasks.

PART III: PERFORMANCE MANAGEMENT SERVER

The Zend Platform Performance Management server is a comprehensive set of tools for boosting PHP script performance and managing PHP applications.

The Zend Platform Performance Management Server components are:

- Platform (Management Console) - Includes the Dashboard, Preferences, license Management and User Management.
- PHP Intelligence - Event configuration, handling and management.
- Performance - Code Acceleration, Dynamic Content Caching, Benchmark and File Compression.
- Configuration - PHP and server configurations.

The Problem Resolution Lifecycle

IN THIS CHAPTER…

THE PROBLEM RESOLUTION LIFECYCLE
CREATING EVENTS
FINDING EVENTS THAT INTEREST YOU
UNDERSTANDING EVENT DETAILS
CONTROLLING INFORMATION DISPLAYED IN AN EVENT
CUSTOMIZING EVENTS
DATABASE MAINTENANCE
GRAPHS

In this chapter we will discuss how PHP Intelligence can provide improved communication between developers, managers and QA teams through implementing the Problem Resolution Lifecycle.

The Problem Resolution Lifecycle

Developing and maintaining Web applications is an intricate and highly demanding process. Zend Platform facilitates the intricacies of the development process by employing an efficient problem resolution infrastructure – "the Problem Resolution Lifecycle." 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.

With the Problem Resolution Lifecycle, organizations can improve communication between the development, testing and IT teams to streamline the development and deployment processes.

Using PHP Intelligence in development and production environments unifies the working environment and ensures improved information collection and distribution between development teams, testing teams and IT teams (See illustration below).

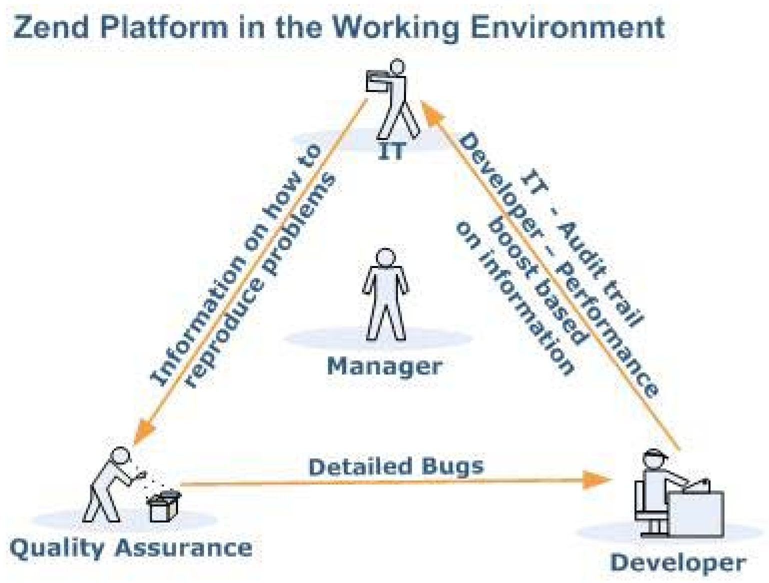

Zend Platform in the Working Environment
IT
Information on how to reproduce problems
IT - Audit trail
Developer - Performance boost based on information
Manager
Detailed Bugs
Quality Assurance
Developer

Figure 32 - Problem Resolution Lifecycle

Using Zend Platform 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 focused on improving and expanding the current application and less time spent on locating information necessary for recreating and resolving code and performance issues

In the Problem Resolution Lifecycle, PHP Intelligence 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.

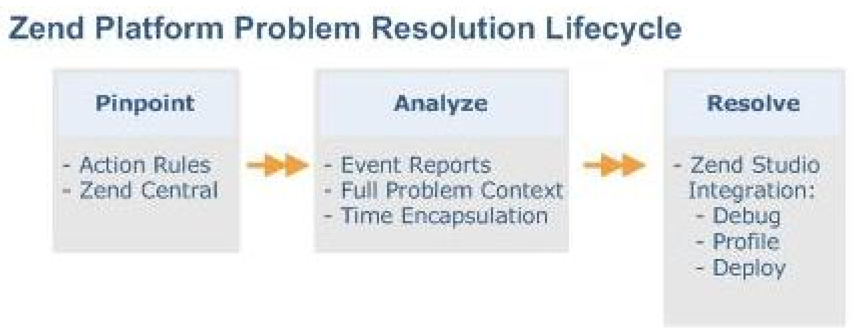


Figure 33 - Problem Resolution Workflow

Zend Platform's PHP Intelligence functionality is enhanced by:

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

Implementing the Problem Resolution Lifecycle

The Problem Resolution Lifecycle is a process of defining PHP Intelligence Event Triggers according to acceptable run-time, performance parameters. PHP Intelligence enforces Event Triggers and issues Event information in an Event Details screen according to the Event Trigger definitions. When an Event occurs, PHP Intelligence compiles a complete profile of the Event's occurrence and its precise details. An Event Details screen includes comprehensive details in order 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 Triggers to accommodate normal occurrences or resolve actual run-time problems and errors.

With Zend Studio Diagnostics, problems and errors can be easily diagnosed using the Event Details screen functions, Test URL and Profile URL, and further information can be analyzed using Debug URL and Show Source Code. In addition, problems in code can be immediately resolved using the Zend Studio Editor which allows changes to be immediately made and deployed, not only to a single server but also to all nodes belonging to the same Group.

Events can be preserved to leave an indicator of these occurrences if necessary. Furthermore, user permissions can define who is permitted to perform actions inside an Event Details screen; enforcing a structure that encourages communication between the different teams.

Creating Events

Event generation is an out-of-the-box feature. Directly after installation, Zend Platform's PHP Intelligence will begin to monitor events according to Default Settings. To further enhance the effectiveness of PHP Intelligence, events 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 done are to do with tuning Event Trigger values and defining a list of Functions and PHP errors to be monitored.

To Configure Event Triggers, go to PHP Intelligence | Event Triggers and change the default settings according to your requirements.

A help button ? appears next to each Event Type. Pressing this button will display a description of the selected Event and the Event's parameters (alternately go to Choosing and Defining Event Triggers).

Disabling Events (Triggers)

In some cases there may be Events that are either not applicable to your system or unnecessary. Events are disabled from the PHP Intelligence module. When an event is disabled the event will not be monitored and no event information will be stored.

Disable Event Triggers

To disable Event Triggers go to PHP Intelligence | Event Triggers and select "Configure Event Triggers."

In the Define Event Triggers Table, the Check box in the Active Column indicates if an Event Type is monitored or not.

To prevent a selected Event from being monitored, disable the Rule by unmarking the Check Box. This will deactivate and stop collecting event related information.

List Entry of Watched Functions

Zend Platform allows you to monitor a list of functions by referencing a text file that includes the functions you wish to monitor. Users who must monitor large numbers of functions will find this method of defining watched functions a convenient alternative to editing the php.ini file line by line.

Use the following PHP functions to reference a text file containing the list of functions to monitor.

The following function is typically used to create a list of functions to watch. It forms part of the php.ini file.

zend_monitor.watch_functions=mysql_connect,mysql_query

The following function refers zend_monitor.watch_functions to a text file at a specific location. This file contains the list of functions to monitor.

```
UNIX, Linux, i5/OS and Mac:
'zend_monitor.watch_functions=@/usr/local/Zend/Platform/lib/watch_funcs.txt'
'zend_monitor.watch_results=@/usr/local/Zend/Platform/lib/watch_res.txt'
Windows:
'zend_monitor.watch_functions="@C:\Program
Files\Zend\ZendPlatform\lib\watch_funcs.txt"'
'zend_monitor.watch_results="@C:\Program Files\Zend\ZendPlatform\lib\watch_res.txt"'
(In Windows the quotes must be present)
```

The text file should contain one function name per line.

Example:

```
mysql_connect
mysql_pconnect
mysql_query
mysql_db_query
mysql_unbuffered_query
```

User functions can also be included in the Watch Functions file. Each user function must be added with its Class (class::function).

If necessary, inheritances should also be included in the file as only functions explicitly specified in the Watched Functions file are watched.

Finding Events that Interest You

Zend Platform provides several ways for viewing Events that occur; each way has different advantages and can be used to suit different requirements as follows:

- Platform | Dashboard | Events at a Glance displays the top five events that occurred, on all the servers. Double clicking on an Event in the Console opens its Event Details screen.
- PHP Intelligence | System Health displays an up-to-date "snapshot" of events monitored by Zend Platform and listed by server and Event Type. Selecting a Location (Node) or a specific Event Type automatically filters the view to display relevant Events in the Event List.
- PHP Intelligence | Event List is a filterable display for viewing events in a table according to various parameters. Choosing "Change Table Fields" modifies these parameters. This opens a selection list of all the possible field options. This window also includes an option to locate events by their Event ID.
- Platform l | Dashboard |Configuration and Management Tools | Event Actions for sending events of a certain type to e-mail or URL according to predefined rules. This provides a proactive means for sharing information either with parties that need to be informed when certain events occur (e-mail) or for integrating event information to other applications (URL). For example: a manager may only want to know about Severe PHP errors that indicate some or all of the Web application is not working. Setting an Event Action to send event information by e-mail means that this manager is immediately informed of the event, as long as the e-mail account is accessible

Note:

Read more about how your organization can leverage information generated by events in the Tutorial – Integrating Existing and Legacy Applications.

PHP Intelligence

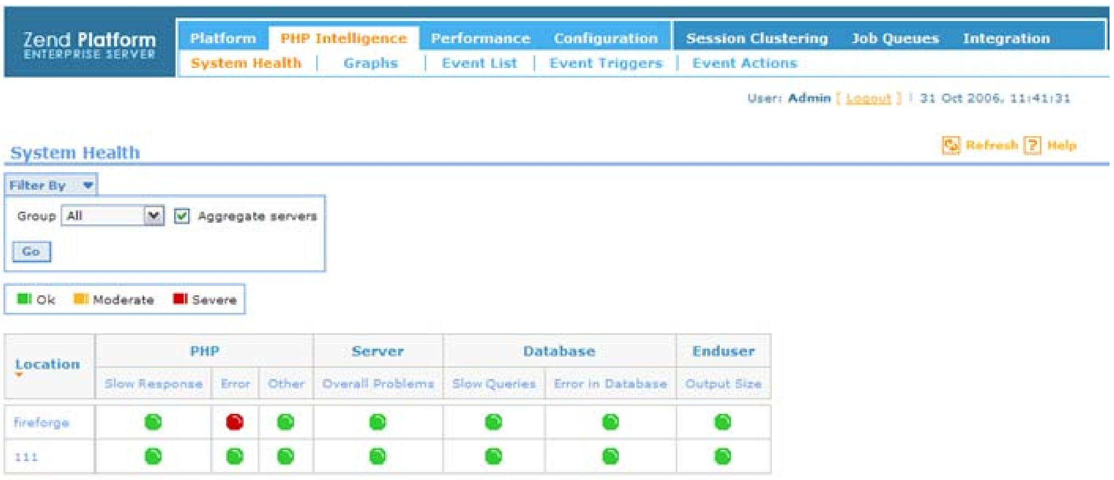
PHP Intelligence provides a means for monitoring activity on clusters and servers.

PHP Intelligence Includes:

- System Health - provides an up-to-date “snapshot” of events monitored by Zend Platform categorized by Host and Event Type.
- Graphs - display Event relates information in a graphical representation.
- Event List - a table of events that includes events that occurred during a user-defined time frame. The information displayed in the Event List can be filtered by Events From, Event Type, Virtual Host, Alert Severity, Status, and Time Filter. You can also find a specific Event by ID.
- Event Triggers - define and/or change triggers for monitoring events on a specific node. From this screen you can: configure Event Triggers, view Event Triggers currently defined for the node, and filter the view of events displayed in the Define Event Triggers table.
- Event Actions - apply actions to triggered Events, send to URL or mail or SNMP trap.

System Health

The most comprehensive way to view and manage events is through the System Health screen. The System Health sub-screen provides an up-to-date “snapshot” of events monitored by Zend Platform. Events are displayed in a filterable table, which is updated by manually refreshing the browser (Clicking Refresh) or by setting automatic refresh (Zend Central | Preferences). The System Health table includes events categorized by location (server/group) and event type.



User: Admin [Logout] | 31 Oct 2006, 11:41:31

System Health

Group: All
[✓] Aggregate servers

🟩 Ok 🟨 Moderate 🟥 Severe

Location	PHP			Server	Database		Enduser
	Slow Response	Error	Other	Overall Problems	Slow Queries	Error in Database	Output Size
fireforge	●	●	●	●	●	●	●
111	●	●	●	●	●	●	●

Figure 34 - System Health Screen

To view the System Health table:

1. Go to PHP Intelligence | System Health.
2. Select the "Filter By" option to filter events displayed in the table.
   Filter options are described in detail below.
3. Click "Go" to display the System Health table.

The fields that make up the Event Summary table are as follows:

- Location - Where the event occurred.
- PHP - Current PHP events for the selected Host.
- Server - Overall events related to general performance of server.
- Database - Database events related to handling of queries.
- End User - The output site; field includes status information about output site.

Filtering Table Data

Zend Platform allows you to filter the event information displayed in the System Health table. There are three group options:

- All - Displays System Health information for all servers (and groups).
- Ungrouped - Displays System Health information for servers that do not belong to a specific group.
- Server by name - Displays System Health information for the selected server group only.
- Aggregate servers - Aggregates the servers belonging to the same group into one row.

To filter the data displayed in the System Health table:

1. Select the Filter By options—Group and/or Event Name—you wish to apply to the table.
2. Click "Go" to display the filtered events in the System Health table.

Aggregation Groups

Event aggregation / reporting is the process of identifying events that were generated for the same reason and can be reported as one event that occurred X times rather that reporting multiple occurrences of the same event.

There are two options for aggregating events:

1. By Server
2. By Group

Aggregation by **group** happens when several servers are placed in a group that is defined as "Aggregated" (Platform | Cluster Management | Manage Groups).
At the bottom of the System Health table there is a message stating that:
(*) Events that took place on the server before the server was a member of an aggregated group.

Aggregation by **server** happens when identical events are generated on different servers. The same event is reported (for each server) separately. This is the default.

Event information can only be aggregated at the time of the occurrence and for information integrity purposes will not be retroactively aggregated. Therefore, events that occur on individual servers (ungrouped) before they were grouped will be reported as un-aggregated events.

Event List

The Event List is a filterable table that displays Events that occurred within a user definable period of time.

Information displayed in the Event List can be displayed as follows:

- Events can be filtered by the following categories: Events From, Event Type, Virtual Host, (Event) Severity, Status, and Time Filter.
- Events can be located in this screen by the event's ID.
- Columns in the Event List table can also be customized (Use Change Table Fields).

Specialized filters can be created and saved using the Manage Filters option.

Filtering by Virtual Host

Several servers can be added to a user-defined group and used to filter the display to show Events that occurred on a specific selection of servers.
To define a virtual Host name that will be added to the list in the filter, go to the Virtual Host filter field and choose "Selected". This option opens the Change Virtual Host screen.
The Manage Filters section allows users to save user defined filter definitions.

To save user defined filter definitions:

1. Define the filter settings in the Filter By section.
2. Select Manage Filters to expand the filter management options.
3. Enter a user defined name for the filter and press Save to add the new filter.
4. Use Load Filter each time you want to use a user defined filter.
5. Use Remove Filter to delete filters from the Load Filter List.

The filter section of this screen also includes a search field for searching for a specific Event by ID.
Zend Platform also allows you to change the table fields displayed on-screen. (Click "Change Table Fields "and select the fields you want to view from the list).

Working with the Event List

To view the Event List:

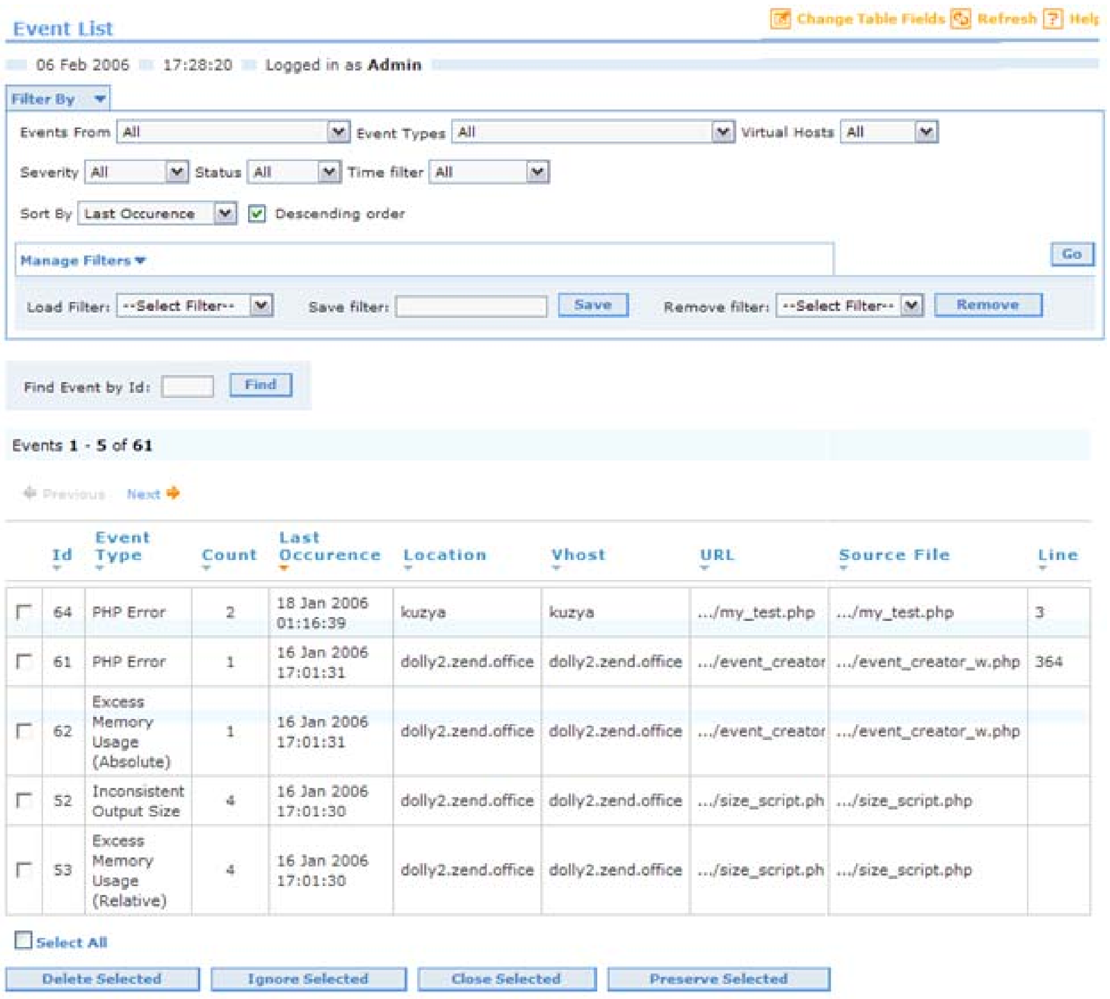
1. Go to PHP Intelligence | Event List.
2. Select the "Filter By" option you wish to apply to the table.

    - **Filter operations appear as follows:**

- Events from - Filter Events according to grouping definitions. These definitions list events according to where they were generated (in the PHP, Database) or what type of event (slow response, error). The values are: All, Bandwidth-Other, Web server-Other, PHP-Slow Response, PHP-Error, PHP-Other, Database-Slow Response, Database-Error
- Event Types - Filter Events displayed in the table according to event type. The values are: All, Slow Script Execution (absolute), Slow Script Execution (relative), PHP Error, Function Error, Slow Function Execution, Excess Memory Usage (absolute), Excess Memory Usage (relative), Database Error, Slow Query Execution, Inconsistent Output Size, Load Average, Custom Events.
- Virtual Hosts - View Event information for either All hosts or selected hosts. Click Selected to open the Change Virtual Host Selection dialog box. All, Selected or predefined user selections.

- Severity - Filters Event information according to severity. The values are: All, Moderate, Severe
- Status - Filters information displayed in the Event List according to the Event's handling status. The values are: All, Opened, Closed, Ignored
- Time Filter - Filters information displayed in the Event List according to a user-defined time frame. The values are: All, Past Hour, Past Day, Past Week, Past Month
- Find Event by ID - Finds (and displays) an event with a specific ID number. (By entering an event id and clicking "Find"f). The values are: Sequential numbers that were assigned to Event Details.

3. Click Go to display filtered events in the Event List table.



06 Feb 2006 17:28:20 Logged in as **Admin**

Events From: All | Event Types: All | Virtual Hosts: All

Severity: All | Status: All | Time filter: All

Sort By: Last Occurence | ✓ Descending order

Load Filter: --Select Filter-- | Save filter: | Remove filter: --Select Filter--

Find Event by Id:

Events **1 - 5** of **61**

	Id	Event Type	Count	Last Occurence	Location	Vhost	URL	Source File	Line
☐	64	PHP Error	2	18 Jan 2006 01:16:39	kuzya	kuzya	.../my_test.php	.../my_test.php	3
☐	61	PHP Error	1	16 Jan 2006 17:01:31	dolly2.zend.office	dolly2.zend.office	.../event_creator	.../event_creator_w.php	364
☐	62	Excess Memory Usage (Absolute)	1	16 Jan 2006 17:01:31	dolly2.zend.office	dolly2.zend.office	.../event_creator	.../event_creator_w.php	
☐	52	Inconsistent Output Size	4	16 Jan 2006 17:01:30	dolly2.zend.office	dolly2.zend.office	.../size_script.ph	.../size_script.php	
☐	53	Excess Memory Usage (Relative)	4	16 Jan 2006 17:01:30	dolly2.zend.office	dolly2.zend.office	.../size_script.ph	.../size_script.php	

☐ Select All

Figure 35 - Event List

The Following list describes the fields that make up the Event List.

- ID - Sequential number assigned to an event.
- Event Type - A descriptive name assigned to the event.
- Count - Number of occurrences of the event.
- First Occurrence - Date and time of event's original occurrence.
- Last Occurrence - Date and time of event's most recent occurrence.
- Location - The name of the server or Aggregated Group where the event occurred.
- Vhost - Name of the Vhost where the event occurred.
- URL - The URL where the event occurred.
- Source File - Path to PHP source file.
- Line - Line in code where event occurred.
- Aggregation Hint - The hint in the code that caused the event to be aggregated.
- Function Name - Name of PHP function where event occurred.
- Status - Status of the event.
- Severity - Severity of the event.

The following actions can be performed on the events in the Events Table:

To apply one of these actions to an event/events select the event by checking the check-box next to the events:

- Delete Selected – Deletes the Events form the database.
- Ignore Selected – A new Event Details screen will not be created for this specific - occurrence Additional occurrences of this event will be added to the original Event's details.
- Close Selected – Changes the status of the event to "Closed" and preserves the Event in the Database. If the same event occurs again a new Event Details screen will be created.
- Preserve Selected – Preserve the selected event from being deleted during Database cleanups.
- Reopen Selected - Changes the status of ignored or closed events to "Open".

Note:
The default value for database cleanup is 7 days.
Depending on the operating system database cleanup settings can be changed in: /usr/local/Zend/Platform/etc/php-embed.ini or<install_dir>\etc\php-embed.ini, by setting a different value for the parameter zend_monitor.event_lifetime

Change Virtual Host

Several servers can be added to a user defined group and used to filter the Event List to show Events and Alerts that occurred on a specific selection of servers.

To create a virtual Host:

1. Choose the servers from the tree.
2. Name the current selection.
3. Click "Save".

To view a Virtual Host's settings:

1. Select a Virtual Host from the Load Selection field.
2. Click "Load".

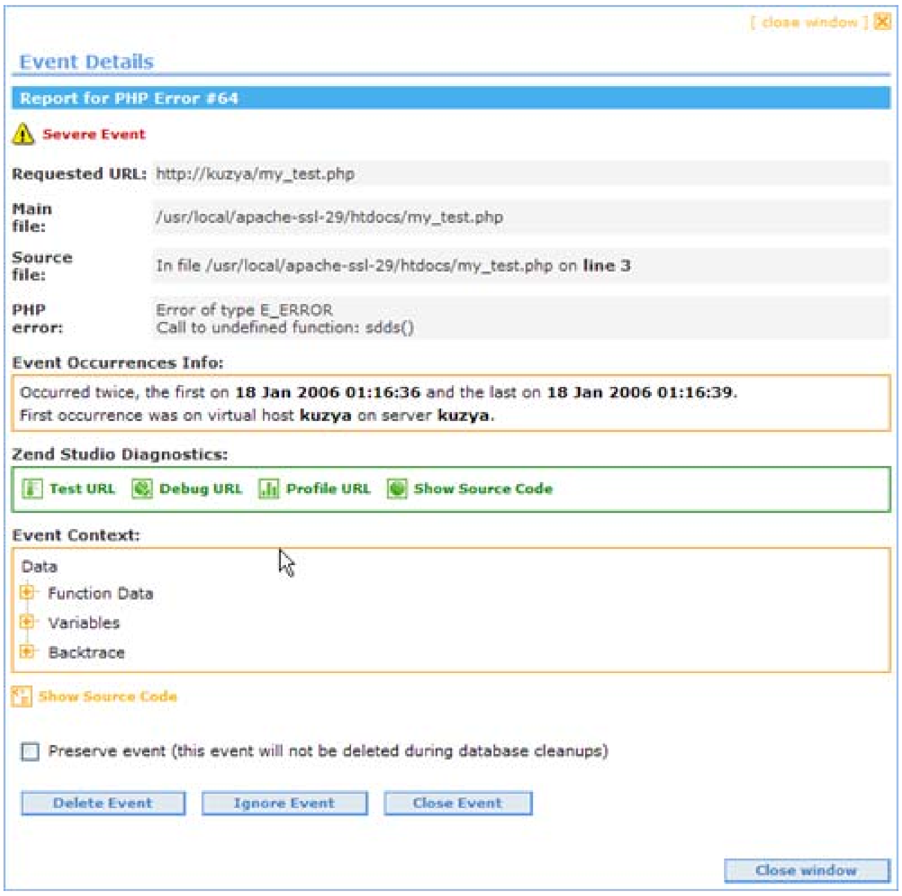
Virtual Hosts can be deleted by clicking "Remove Selection".

Event Details

Understanding Event Details

Event Details are generated in accordance to Event Triggers. Event Details are a diagnostic tool that provides a complete audit trail and options for investigating and resolving events.

Event Details are viewed in several ways. The regular way of viewing events is from Zend Platform. Events can be configured to be sent by e-mail recipients or to a URL (in XML format). However, Event Details always include the same information regardless of the viewed output (Regular Event Details, XML or e-mail).



Requested URL: http://kuzya/my_test.php

Main file: /usr/local/apache-ssl-29/htdocs/my_test.php

Source file: In file /usr/local/apache-ssl-29/htdocs/my_test.php on **line 3**

PHP error: Error of type E_ERROR
Call to undefined function: sdds()

Event Occurrences Info:

Occurred twice, the first on **18 Jan 2006 01:16:36** and the last on **18 Jan 2006 01:16:39**.
First occurrence was on virtual host **kuzya** on server **kuzya**.

Figure 36 - A PHP Event Details Screen

Event Details include five sections:

1. General Information
2. Event Occurrence Info
3. Zend Studio Diagnostics
4. Event Context
   - i. Function Data
   - ii. Variables
   - iii. Backtrace
   - iv. Included Files
   - v. Show Source Code
5. Event Administration

The information included in Event Details varies according to event type. For example: PHP Error Event Details include different information than a Slow Script Execution Event Details, simply because these events require different information to perform diagnostic analysis.

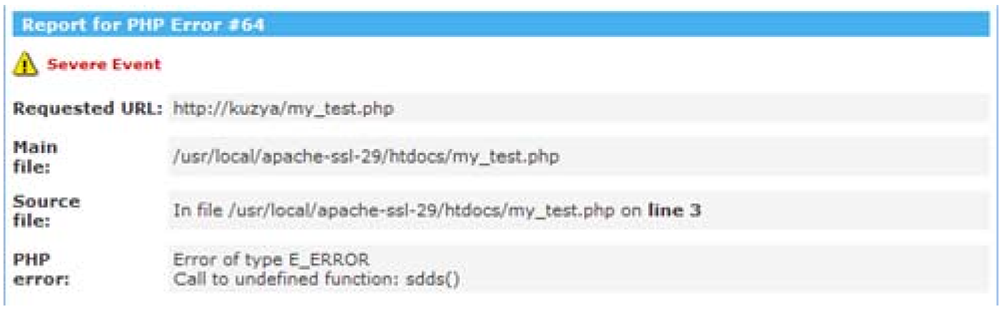
For example: A PHP Error will include in the General Information, the error's text and in the Event Context, the Function's Data. However, in a Slow Script Execution Error there is no need for the error text or the function's data and there will be information on how long the script ran for, included files and the load average at the time of the event.

Note:

If there is not relevant data to display in an Event Details screen, the section will not be included rather than appearing empty.

General Information

The general information section of the Event Details provides basic information about the event (depending on the event's type) as follows:



Report for PHP Error #64

Severe Event

Requested URL:	http://kuzya/my_test.php
Main file:	/usr/local/apache-ssl-29/htdocs/my_test.php
Source file:	In file /usr/local/apache-ssl-29/htdocs/my_test.php on **line 3**
PHP error:	Error of type E_ERROR Call to undefined function: sdds()

Figure 37 - Event Details - General Information

- Title - The top of the Event Details, displays the event that generated the Error and the Event ID (Error #). The Event ID can later be used, to locate the event in the Event List or the Console in the filter's Find section.
- Severity Level - An additional notification is added to severe events.
- Event Status - The Status of the event is indicated for all statuses except Opened.
- Requested URL - The requested URL
- Main Filename - The URL's main file
- Source Info - The path to the Source File and line in the code that triggered the event.

Note:
Especially with code related errors, this information can provide an immediate indication to the source of the error in the code.
Trigger Value The script's trigger value (runtime,output size,memory consumption etc.)

- CPU Load - The CPU load when the event occurred.
- Zend Error/ Error Description - Shows the event's error text (for code related errors) and the $type of Custom Events
- Aggregate Hint - Shows the Aggregate Hint for this event
- Associate Zend Error - Adds a link to an associated Zend error event.

Event Occurrence Info

To prevent an event from being continually reported for the same or similar event, PHP Intelligence enforces Aggregation Rules. These rules are based on a set of predetermined algorithms that determine which events are identical or are similar, to the extent they can be reported as a single Event. Aggregation Rules are also aware of events that occur on nodes belonging to a cluster – Groups and not just occurrences on a single server (node).
Aggregation information is displayed in event details to identify the number of times the event occurred and in case of Groups the information is expanded to include the servers on which the event occurred and the number of occurrences.

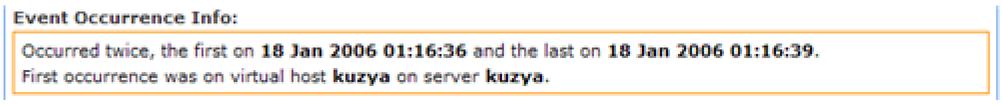

Event Occurrence Info:
Occurred twice, the first on **18 Jan 2006 01:16:36** and the last on **18 Jan 2006 01:16:39**.
First occurrence was on virtual host **kuzya** on server **kuzya**.

Figure 38 - Event Details - Event Occurrence Info

The Event Occurrence Info section shows, the total number of occurrences, the time and date of the first and last occurrence and the first server (and vhost) on which the event occurred.
In cases where an event occurred on several servers belonging to the same group, an additional expandable list is added. This list displays occurrences per server, i.e. the server's name, and total number of times the event occurred on the server.

Zend Studio Diagnostics

The Zend Studio Diagnostics section shows the advanced diagnostic options that can be performed on the event's data.
These diagnostic options reconstruct the precise conditions that generated the event by recreating the request with the same parameters that were in the original request (Information such as: GET/POST/COOKIE/etc.).

Note:
The recreation process will not create an additional event.

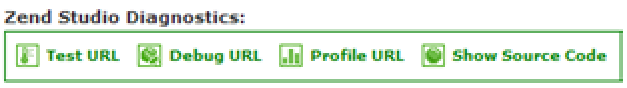


Figure 39 - Event Details - Event Diagnostic Options

The diagnostic options that can be applied to event information are as follows:

- Test URL - Loads the exact same URL from the event with the exact same parameters (GET/POST/COOKIE/HTTP HEADERS/etc.) and shows the script's output in the browser.

Note:
The Test URL option does not require the Integration with Zend Studio.

- Debug URL - Initiates a Debug session of this URL in the Zend Studio.
- Profile URL - Profiles the URL, using the Zend Studio Profiler with the same parameters (GET/POST/COOKIE/HTTP HEADERS/etc).
- Show Source Code - Opens the file where the event occurred in Zend Studio. This option provides a means for editing files and implementing changes to multiple servers using Zend Studio.

Important: Debug URL, Profile URL and Show Source Code can be activated when the following conditions are met:

1. Zend Debugger is installed on the server where the event occurred.
2. Zend Studio (ZDE) is open.
3. The Platform Administration preferences (Zend Central | Preferences) are configured to the correct port (The port on which Zend Studio is listening), the Zend Studio IP is correct (the exact IP of the computer where the Zend Studio resides), and the Debugger allows a debug session from Platform Administration (by going to: Zend Central | Configure PHP Settings | Zend | Zend Debugger and verifying the correct IP/S in the zend_debugger.allow_hosts directive).The Server settings are configured to the correct Host in Zend Core (See the Zend Core User Guide for more information).

Event Context

Event context includes relevant information available at the time of the occurrence. This information varies according to the type of event generated.

There are four main Event context categories:

1. Function Data
2. Variables
3. Backtrace
4. Included Files

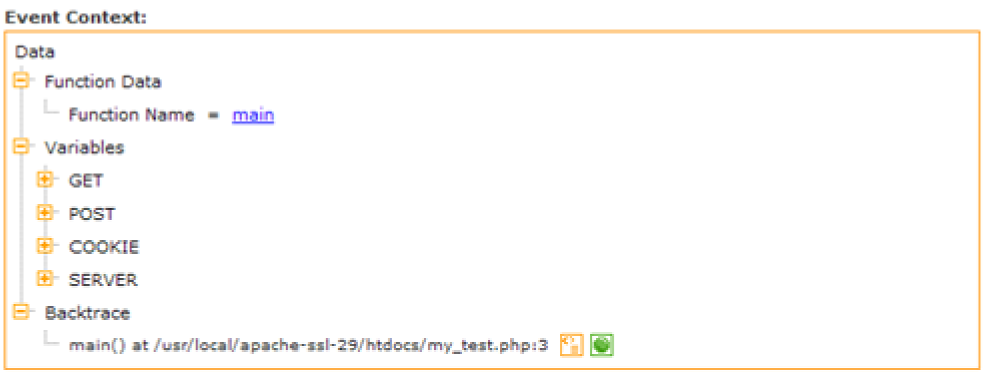

Event Context:
Data
Function Data
Function Name = main
Variables
GET
POST
COOKIE
SERVER
Backtrace
main() at /usr/local/apache-ssl-29/htdocs/my_test.php:3

Figure 40 - Event Details - Event Context

Function Data

Function Data is only added to function related Event Details. This addition shows the function's name and parameters at the time the event occurred.
The function always appears as a link. This link directs to the function's description in the online PHP manual at: http://www.zend.com/manual.

Note:
The link to the PHP manual also appears for user-defined functions. These functions naturally, will not be found in the PHP manual however, it is a good indication as to which functions are PHP functions and which are user-defined.

Variables

The information included in the Variables section, includes all variables data saved when the event occurred, such as: GET, POST, COOKIE, SERVER, etc.
The GET, POST, COOKIE and SERVER sections will always be displayed even if they are empty. This indicates that there was no available data at the time the event occurred.
Users may choose to change the type of information collected and displayed in an event (Change Event Details).

Backtrace

Backtrace only appears in function related events. The listed functions are the functions that lead to the actual function (occurrence) that triggered the error.
Functions are listed in chronological order from the most recent to the first function that was called.

There are two options for viewing Backtraced functions, in a pop-up screen or in Zend Studio.

- - Shows the function call in a pop-up screen.
- - Shows the function call in Zend Studio.

Included Files

The Included Files section only appears in slow script error events. The files listed are all the files that are included in the PHP script that caused the error to occur.

Show Source Code

Shows event data in the Event details in the form of an expandable text field. This option opens the file in the section of code where the event occurred.

To view the source code in Event Details:

Click Show Source Code to expand the text area.

Event Administration

The Event Administration section includes all the actions that can be applied to an event see Controlling Information Displayed in an Event, to learn how to disable these options for certain users.

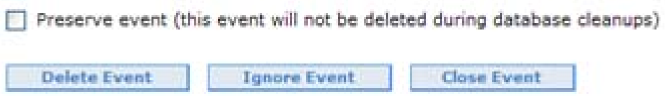

Preserve event (this event will not be deleted during database cleanups)
Delete Event
Ignore Event
Close Event

Figure 41 - Event Details - Event Administration Actions

The applicable actions (Buttons) are as follows:

- Preserve Event - Keeps the event in the database even during database cleanups.
- Delete Event - Deletes the event (i.e., removes the event entirely from the database.)
- Ignore Event - Ignores future instances of this event (i.e., changes the event status to ignore). Therefore, if the same event occurs again, Platform will not open a new event.
- Close Event - Closes the event (i.e., changes the event status to closed). Therefore, if this event occurs again, Platform will open a new event.
- Reopen Event -Changes the event's status from closed to open.

Controlling Information Displayed in an Event

Zend Platform's User Management settings (Zend Central | User Management), can be utilized to set restrictions per User Group. These restrictions can control permissions to view event information and prevent certain User Groups from changing Event Details status.

The following restrictions can be applied to Event Details information:

- Delete an event
- Ignore an event
- Close an event
- Reopen an event
- See the event internal data in the Event Details
- See the event source code in the Event Details
- Use the Zend Studio Diagnostics in the Event Details

These limitations can prove to be especially useful for the organization. For example: when working in collaboration with external organizations that should not be permitted to view information such as the source code.

Another example of the Event Details restrictions is seen when implementing development lifecycle processes that require that certain groups be limited to the actions that they can do with an event such as closing or reopening.

Note:

To find out how to create a User Group, go to: "Configuring Users and User Permissions.

Change Event Details

Variables

The information included in the Variables section, includes all variables data saved when the event occurred, such as: GET, POST, COOKIE, SERVER, etc.
The GET, POST, COOKIE and SERVER sections will always be displayed even if they are empty. This indicates that there was no available data at the time the event occurred.
Users may choose to change the type of information collected and displayed in an event at the time of the occurrence.

To define the type of variables data that will be displayed in Event Details:

1. Go to: Platform | Dashboard | Configure PHP Settings and locate the function zend_monitor.report_variables_data from the tree under: Directives | Extensions | Zend Platform.
2. Select the Variables types that you want displayed in Event Details by selecting the check box to the left of each variable and clicking "Save".

The new settings will be applied as soon as the Web Server is restarted.

Note:
Changes applied to the Event Details will take affect after the Server is restarted. Events that occurred before the changes will not be affected.

Customizing Events

Zend Platform provides several ways for customizing events in order to facilitate different requirements.

- Custom events – for generating a User Defined event that is not based on specific PHP Intelligence, Event Triggers.
- Event Callbacks – for adding user defined information to Event Details.
- Aggregation API – for setting different events to be aggregated with other events.

Custom Events

Custom events are a unique type of event that is provided for Zend Platform users in order to initiate events in their scripts. This type of event is different than other event types in that it allows controlling event generation as opposed to the other events that are triggered by a certain occurrence.
Custom events are used to generate an event whenever the API function monitor_custom_event() is called from the PHP script.

Description:

This event type enables the generation of an event on occurrences that are not necessarily built-in Zend Platform events (error and performance issues). Custom events are used whenever you decide that it is significant to generate an event in a certain situation. Each event type is given a name for easy identification ($type).

Function Usage:

void monitor_custom_event(string $class, string $text[, integer $severe, mixed $user_data])

Parameters:

- $class – helps to define several types of custom events. This description will be showed in the Event List and in the Event Details.
- $text - error text used to describe the reason for the event. This text will appear in the Event Details.
- $severe - the severity level of the triggered event, default value is Severe.
- $user_data - adds a PHP variable that will be viewed in the Event Details (in Event Context->Variables->User Defined). This forms the stored Event Context (similar to the information obtained in a PHP error event).

Aggregation takes place for these events when, two events occur in the same place and have the same $class $text $sever(ity)
When viewing these events in PHP Intelligence | Event List, they can be filtered by the Custom Events category.

Note:
Event Actions defined for these events should be set to “send to URL” rather than “sending by e-mail” as there is only one definition for these events and Event Details sent to a URL can be easily forwarded elsewhere. This is to prevent the overloading of e-mail. If we use the e-mail action, for every custom event, e-mail will be sent, and there can be many classes of custom events. However if the URL action is used, a script can be used to identify the event’s class and different behaviors can be implemented according to class.
(To find out how to leverage Event Details information sent to URLs go to: Tutorial - Integrating Existing and Legacy Applications)

Event Callbacks

The event callback mechanism is used for viewing additional information about local variables in order to investigate what happened when an event was generated. The additional information is displayed in the Event Details.
Event Callbacks are created by extending information already provided in Event Details to provide an audit trail for problem conclusion.

Register and Un-register User Event Handlers

The event callback mechanism uses the following API functions:

- register_event_handler
- unregister_event_handler

Register Event Handler

To register a user function as an event handler, the following API can be implemented:

```
register_event_handler($event_handler_func
[[,$handler_register_name],$event_type_mask])
```

Parameters:

- $event_handler_func - The first argument is a callback function that will be called when the event is triggered. Object methods may also be statically invoked using this function, by passing the array ($objectname, $methodname) to the function parameter.
- $handler_register_name - The second argument is optional and represents the name under which the function is registered. If no name is specified, the function will be registered under its own name.
- $event_type_mask - The third parameter is an optional mask of event types on which the handler should called. The default setting is MONITOR_EVENT_ALL.

When a monitor event is triggered, all the user event handlers are called and the return value from the handler is saved in an array keyed by the name under which the event handler was registered. The event handlers' results array is saved in the script_runs table.

Notes:

The first parameter is the name of the called function and it has to be a user-defined function. Built-in functions will not work with this API.
This function can get as a parameter the event type by which it was called.
If there is a PHP function register_error_handler in the JavaScript, events will not be reported. To report events call the function monitor_pass_events in the error handler.
Global Events should not be changed under any circumstances as they may produce unpredictable results.

Un-register Event Handler

The un-register event handler allows you to un-register an event handler. The API returns false if it cannot find a handler registered under the supplied name.

```
unregister_event_handler($register_event_handler)
```

Note:

Do not add the unregister_event_handler function to the end of scripts if you need to generate memory and script execution Events. These event types generate the event only after the script is executed and if unregister_event_handler is added it will stop the event from being generated.

The event types that should not include unregister_event_handler are as follows:

- Slow Script Execution Absolute
- Slow Script Execution Relative
- Inconsistent Output Size

Aggregate Hints

"monitor_set_aggregation_hint (page name)"

This API is a global variable that can be set anywhere and in any hierarchy. The purpose of this API is to incorporate locations of occurrences in the script.
This API is used when there are events that require the location in the script for diagnosing the reason behind the event occurring.

For example:

Global Events require the application that generated the event. Adding the Hint API can assist in the identification process.

Event Aggregation Rules

Event aggregation rules determine which events are aggregated into a single Event Detail.

There are four types of checks depending on the event type:

1. Database Error
2. Zend Error (PHP Error)
3. Function Error
4. Query Error

The Collector checks to see if these events occurred in the same source file based on the Line, Function and Hint. This is why it is important to used Hints if this level of separation is necessary.

Note:
To read more about the event aggregation mechanism go to: Appendix D - Event Aggregation Mechanism.

Database Maintenance

Once the cause of the event has been fixed, we can decide what to do with the event: Preserve, Ignore or Delete. If no actions are done to an event, it will be automatically deleted from the database.

Apart from closing events, other additional advantages can be obtained by customizing user permissions. Granting different users separate authorizations, by configuring different user permissions can facilitate different organizational requirements such as enforcing responsibilities and work structures. For example: by providing “Read Only” authorization to people who only need to see event details and granting authorization to Close events only to those who should close events (such as managers or team leaders) we can create and maintain a structured working environment.

The Event List provides four options for handling events in the system:

- Close Event - Closes the event (i.e., changes the event status to closed). Therefore, if this event occurs again, Platform will open a new event.
- Ignore Event - Ignores future instances of this event (i.e., changes the event status to ignore). Therefore, if the same event occurs again, Platform will not open a new event.
- Delete Event - Deletes the event (i.e., removes the event entirely from the database.)
- Manual Override - Users can also manually change the status of an event by clicking on the event in the Event List and changing its status. This method is helpful, for example, if you want to "un-ignore" an ignored event and restore it to the main screen.

Graphs

Zend Platform includes an option to display event related information in a graphical representation (Pie and Bar graphs).

Graph Behavior

The Zend Platform graphs display general statistics about event segmentation. All graphs display tool-tips with a description of the results displayed in the graph.

Graphs are viewed from two different locations:

1. Platform | Dashboard (only Pie Charts)
2. PHP Intelligence | Graphs

The following lists the available list of graphs, their display type and other additional information:

- Top 5 Events by Aggregation Hints - Pie Chart
- Top 5 Events by Event Type - Pie Chart, Drill down to view a list of events relevant to the selected part of the chart
- Top 5 Events by Location - Pie Chart, Drill down to view a list of events relevant to the selected part of the chart.
  An asterisk (*) next to the name of a server indicates that the statistics contain numbers about events that took place on the server before the server was a member of an aggregated group.
- Top 5 Events by Script - Pie Chart
- *Total Events by Hour - Bar Graph
- *Total Events by Month - Bar Graph
- *Total Events by Weekday - Bar Graph

* These graphs can only viewed from PHP Intelligence | Graphs

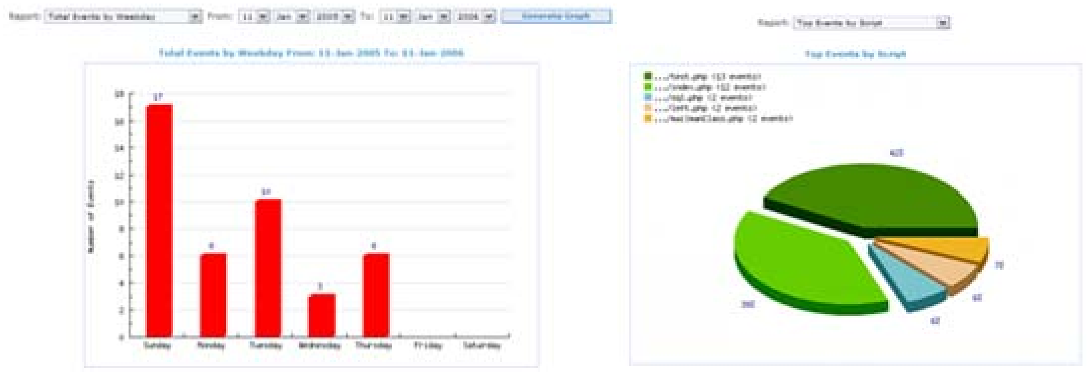


Figure 42 - Bar Graph and Pie Chart

Generating Graphs

Regular pie charts do not require any additional actions in order to generate a graph. However, bar charts display information based on a specific parameter therefore, a parameter has to be selected and defined in order to generate a Bar Chart representation.

To generate a Bar Chart:

1. Go to PHP Intelligence | Graphs.
   The Graphs screen opens.
2. Select one of the bar chart options: Total Events by Hour, Total Events by Month, and Total Events by Weekday.
   The display changes to add input fields to select the range for the Graphs.
3. Define the range and click “Generate Graph”.
   The graph will be generated and displayed on screen.

This completes the PHP Intelligence chapter of the User Guide.
In this section we have described the Problem Resolution Lifecycle and how it can be implemented in an organization’s environment.

Performance

IN THIS CHAPTER...

OVERVIEW
PERFORMANCE LIFECYCLE
IMPLEMENTING THE PERFORMANCE LIFECYCLE
EVENT TRIGGER SETTINGS AND ANALYSIS
PERFORMANCE OPTIMIZATION TOOLS
TUNING
ACCELERATOR PERFORMANCE LEVEL DESCRIPTIONS
TUNING ZEND PLATFORM FOR OPTIMAL PERFORMANCE ON I5 OS

The Zend Performance module provides a collection of comprehensive tools for enhancing PHP Web applications and Server performance in Enterprises.

Using Performance Provides:

- Increased server throughput, with less hardware
- Improved user-experience, with faster response time and download time
- Reduced stress on production database servers and http servers
- Reduced costs on new hardware purchases and IT maintenance operations
- Better utilization of existing hardware resources and capacity

Overview

The Zend Performance module consists of several components for providing server performance optimization:

- Performance Tests
- Code Optimization
- Dynamic Content Caching
- Code Acceleration
- File Compression

Code Optimization

Code optimization begins from the first moment Zend Platform is installed. The Zend optimization component performs several passes, each pass searches for specific points in the PHP code that are known to have a negative affect on performance and changes them for faster execution.

Dynamic Content Caching

Dynamic Content Caching dramatically reduces the number of times your server must run complex scripts, execute resource-intensive database queries, or call external web services.
How it works: Server-side caching eliminates the need to return to databases, duplicate processes or re-build a web page for each page access. Cached versions of any URL can be maintained for any amount of time that you determine. Fully configurable parameters determine what to cache and based on which conditions. No code-level modifications are required. Moreover, you can use the PHP API for partial and conditional caching of parts of script functionality.

Code Acceleration

Code Acceleration begins from the first moment Zend Platform is installed. The Zend acceleration component performs a pre-compilation of your PHP scripts, eliminating the lag time and interpreter time involved in script parsing. During compilation, the code is also optimized, resulting in even faster execution time.
How it works: Server-side pre-compilation generates persistent bytecode. Modified scripts are automatically detected. Compiled scripts are optimized using advanced code optimization methods.

File Compression

File Compression increases the end-user download speed and decreases the workload on your http server. Better than any other compression option due to integration with Dynamic Content Caching eliminating the time it takes to run the compression.
How it works: Specific browser capabilities are auto-detected. If browser supports gzip format, the results are compressed prior to returning to the user. Both the original and the compressed version are cached and reused, depending on browser capability and cache lifetime.

Performance Lifecycle

Maintaining Web applications at optimal performance levels is a necessary requirement for ensuring customer satisfaction and organizational efficiency. Zend Platform's Performance module provides tools for optimizing Web application performance by employing a detailed performance enhancement method – the Performance Lifecycle. The Performance Lifecycle is a process of calibrating Zend Platform to provide an optimal performance boost to business critical Web applications.

Deploying Zend Platform in organizations will improve the overall performance of Web applications by:

- Enhancing code and download performance.
- Employing full and partial page caching capabilities.
- Preserving memory consumption through file compression.

The use of Zend Platform Performance tools in development and production environments provides a means for testing and maintaining Web application performance.
The following illustration displays the three stages of the Performance Lifecycle.

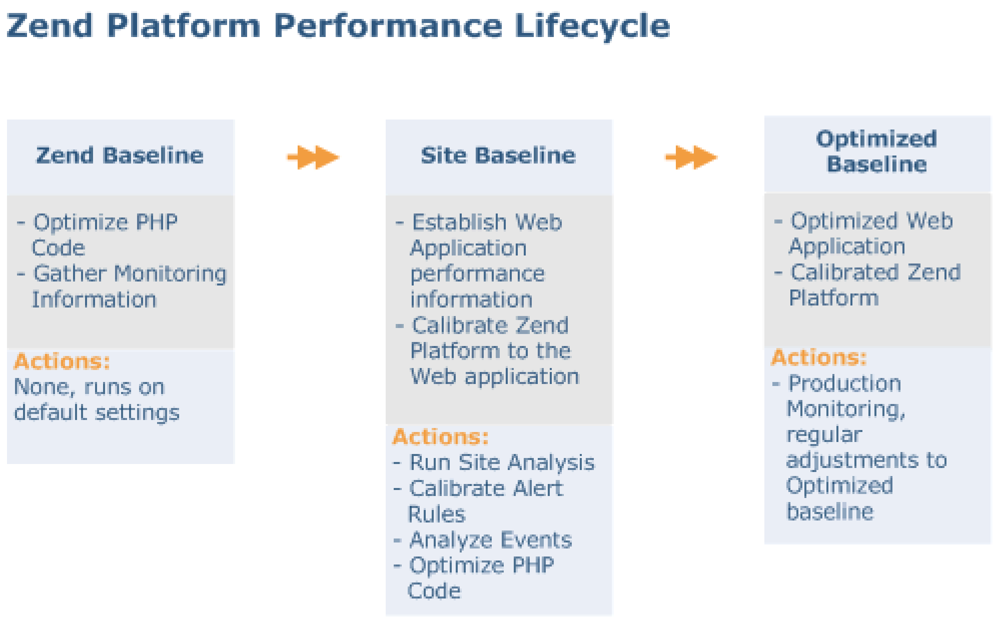

Zend Platform Performance Lifecycle
Zend Baseline
- Optimize PHP Code
- Gather Monitoring Information
Actions:
None, runs on default settings
Site Baseline
- Establish Web Application performance information
- Calibrate Zend Platform to the Web application
Actions:
- Run Site Analysis
- Calibrate Alert Rules
- Analyze Events
- Optimize PHP Code
Optimized Baseline
- Optimized Web Application
- Calibrated Zend Platform
Actions:
- Production Monitoring, regular adjustments to Optimized baseline

Figure 43 - Performance Lifecycle

The Zend Platform Performance Lifecycle is an iterative cycle for analyzing Web application performance. The purpose of this cycle is to identify areas that require Zend Platform calibration and areas that require PHP code optimization.

The Performance Lifecycle Baselines (stages) are as follows:

Zend Baseline

The first Baseline is the Zend Baseline. This baseline measures performance based on Zend Platform's default parameters that are immediately activated upon installation. The default parameters are, Event Trigger settings, Code Optimization and Code Acceleration. Once Zend Platform is installed, these components automatically begin to work on the PHP code. The result is an immediate improvement to the Web application and initial PHP Intelligence event generation (based on default Event Trigger settings).

The purpose of the Zend Baseline is to evaluate overall performance in relation to the Zend Platform defaults. This information is used as an initial starting point for subsequent calibration and optimizations.

Note:
In the Zend Baseline stage, it is common to experience abnormal event generation behavior (too many or too little events generated). This is a normal part of the initial calibration stage, necessary for identifying how to adjust performance settings to obtain optimal Web application performance.

Site Baseline

The second Baseline is the actual calibration process. Based on information collected and observed in the first Baseline, the performance settings can be calibrated to suit each organization's specific Web application. The Site Baseline enables one to obtain insight into the overall performance of the Web application. Once the Site Baseline is established by configuring Event Triggers these events can be further analyzed to evaluate the mode of action required to optimize the Web application's performance. At this point it is recommended to perform a Site Analysis to benchmark the Web application. The Benchmark information provides an initial indication of the Web application's current performance before applying the additional performance tools. This will provide a point of comparison to view improvements that occur after subsequent optimization is done with Zend Platform.
After the Site analysis, the PHP code can be optimized. Optimization is obtained by implementing performance tools to areas in the PHP that exceed the Site Baseline settings (still generate events).

There are four possible choices for adding performance features to PHP code:

1. Full page caching
2. Partial page caching
3. Compression
4. Blacklist files or directories

Completing optimization of the Site Baseline brings us to the Optimized Baseline.

Optimized Baseline

The Optimized Baseline represents the stage where the Web application is optimized and Zend Platform is calibrated with the Web application. From this stable stage all that is left to do is to let Zend Platform perform regular Production Monitoring.

Note:
When the Web application is re-deployed or changes are made, this process should be repeated from the Site Baseline stage in order to reestablish the Optimized Baseline.

Now that we have established what the performance lifecycle does and the tools it comprises. The next step is to see how to implement the performance lifecycle. At the end of this guide you will find a Performance Lifecycle Check List that details the steps to establishing an Optimized Baseline for Web applications. The next chapter "Implementing the Performance Lifecycle" details each of these steps.

Implementing the Performance Lifecycle

The following section provides a detailed instructional overview of performance optimization features and components for implementing the Performance Lifecycle.

Benchmark - Site Analysis

Site Analysis enables to obtain insight into the overall performance of Web applications. Benchmark information provides an initial indication of the Web application's current performance. This information can be used as a starting point for observing the performance boost gained applying the performance tools. Benchmarking measures Web server performance and durability.
In Zend Platform, Benchmarking is achieved through the Testing screen (Performance | Testing).

This screen includes three options.

- Test URL – tests a single script, running Performance and Compression tests at the same time. The test results indicate the script improvements achieved by Code Acceleration, Dynamic Content Caching and File Compression.
- Analyze Site – tests performance for the entire Web application, running the Performance test separate from the Compression test. The test results indicate the overall script improvements achieved by Code Acceleration, Dynamic Content Caching and File Compression and the popularity of each file.
- Test Download – tests the efficiency of the Zend Download Server. This test is addressed in the Enterprise Server chapter titled "Zend Download Server".

The above-mentioned tests analyze an entire site's performance or monitor a single script. The test results can be further used outside Zend Platform as they can be printed or sent by e-mail.

Notes:
Since testing may take a while to run, it is suggested that you choose only the most recently added files. You may select as many files as you wish; nonetheless this will increase the duration of the test.
Prior to running the Compression Test and in order to ensure accurate results, you may want to add query strings to the script path entries.
When running Performance Tests, query strings can only be added to cached scripts, to check for performance gain.

Testing

The Testing workspace shows you the improvement achieved in performance by Zend Platform Performance.
In addition, it identifies files that are prime candidates to be cached-a convenient feature during initial configuration.

There are three tests, which you can run:

- Test URL - Tests a single script, running both the Performance and the Compression tests at the same time. The test results indicate the script improvements achieved by Code Acceleration, Dynamic Content Caching and File Compression.
- Test Download - Tests the efficiency of the Enterprise Server feature, Zend Download Server.
- Analyze Site - Tests performance for the entire site by running the Performance test separately from the Compression test. The test results indicate the overall script improvements achieved by Code Acceleration, Dynamic Content Caching and File Compression and the popularity of each file.

All test results can be sent by e-mail to view or archive performance information.

Test URL

Test URL, tests a single script, running Performance and Compression tests at the same time. The test results indicate the script improvements achieved by Code Acceleration, Dynamic Content Caching and File Compression.

To test a URL (Benchmark Web applications):

1. Go to: Performance | Testing and select the Test URL tab.
2. Click "Test URL "and type the full path of the script. To select a previously tested URL, click "Show History". By default, URLs are tested using GET variables defined in the query string.
3. Click "Add variables to URL" to add the variable Name and Value to test URLs using specific SESSION or COOKIE variables. Add the User Name and Password to test URLs that are restricted by HTTP Authentication.
4. To delete any variable from the list, click "Delete" next to the variable.
5. To determine a test's duration, specify the time in seconds (per script) in the "Duration of test" box.
6. Press Run.

Note:
To add specific SESSION variables to the test, make sure that your PHP is configured correctly to work with sessions. For example: if you use 'files' as your session.save_handler, confirm that the session.save_path is a valid path. If you use 'user' as your session.save_handler, you must prepend the file containing the user-level session storage functions.

Test Results

The Test Results screen depicts the Dynamic Content Caching Results, the Code Acceleration Results, and the File Compression Results:

- Dynamic Content Caching Results compare between the Base script and the script after it has been cached, and calculates the total performance improvement accomplished due to caching.
- Code Acceleration Results display the performance gain. If the script is cached, acceleration does not improve performance acceleration is therefore not necessary
- File Compression Results compare between the original file size and the compressed file size and calculates the savings in bytes and the improvement in percentage.

Additional Variables

By default, URLs are tested using GET variables defined in the query string.

- To test URLs using a specific SESSION or COOKIE variable, add the variable Name and Value.
- To test URLs restricted by HTTP Authentication, add the User Name and Password.

Analyze Site (Benchmark)

Benchmark Web Applications:

To Benchmark Web Applications:

Go to Performance | Testing and select the Test URL Tab.

To test a script, follow these steps:

1. Click Test URL and type the full path of the script. To select a previously tested URL, click Show History. By default, URLs are tested using GET variables defined in the query string.
2. Click "Add variables to URL" to add the variable Name and Value to test URLs using specific SESSION or COOKIE variables. Add the User Name and Password to test URLs that are restricted by HTTP Authentication.
3. To delete any variable from the list, click next the variable (To add specific SESSION variables to the test, make sure that your PHP is configured correctly to work with sessions. For example: if you use 'files' as your session.save_handler, confirm that the session.save_path is a valid path. If you use 'user' as your session.save_handler, you must prepend the file containing the user-level session storage functions).
4. To determine the Duration of test, specify the time in seconds (per script) in the Duration of test box.
5. Press Run.

Test Results

The Test Results screen depicts the Dynamic Content Caching Results, the Code Acceleration Results, and the File Compression Results.

- The Dynamic Content Caching Results compares between the Base script and the script after it has been cached, and calculates the total performance improvement accomplished due to caching.
- The Code Acceleration Results displays the performance gain. If the script is cached, acceleration does not improve performance acceleration is therefore not necessary
- The File Compression Results compares between the original file size and the compressed file size and calculates the savings in bytes and the improvement in percentage.

When running the Analyze Site test, you can choose to run the Performance test separately from the Compression test:

- Performance Test results - indicate the average improvement achieved by Code Acceleration, Content Caching and the average overall improvement achieved;
- Compression Test results - present the improvement accomplished due to the File Compression.

To Analyze a Site:

Go to: Performance | Testing and select the Analyze Site tab.

The following options are presented in the tab:

- Run Performance Test
- Run Compression Test
- Show Last Performance Test Report
- Show Last Compression Test Report

Performance Test

1. Select the number of scripts to test and press Next.
2. The following screen lists the scripts selected. To change the number of the scripts to be tested, click "Previous". To continue, click "Next"
3. The Site Analysis Report appears indicating the Performance Gain and Popularity Rank of the scripts.

N/A Test Results

N/A test results can be caused by various reasons.

To check the cause of a N/A test result:

Place the cursor on top of the N/A and the cause of the problem appears in a Tooltip.
N/A can be caused when a script cannot be accessed or if the access time to the script is greater than the test duration (3 sec). If this is the case, test the script separately using the Test URL option.
After analyzing the test results you may wish to modify the caching status of the scripts and re-run the test.

To re-run the test:

Click "Edit" next to the script you wish to modify, change the caching conditions and save the new settings. If the message appears, the test results are incorrect since the changes you made did not take effect. In this case, restart the server and simulate a typical user session to get valid results.

Compression Test

The Compression test analyzes download time improvement of the popular scripts as the result of Compression.

1. Click "Run Compression Test".
2. Choose the number of scripts to test and click "Next".
3. To ensure accurate results for the scripts, add query strings in the Script Path entries. For example: /site/example.php?var1=value1&var2=value2).
4. Click "Run" to run the test.

The Site Analysis Report screen details the results for the compression test. The report shows the original script size, along with the compressed size and the compression improvement rate. The actual compression functionality is not affected.

Note:

If one or more scripts failed the test, an indicative message appears on the screen. The Compression Test failed since Platform Performance cannot resolve the URL from some script paths or the URL cannot be accessed. Note that scripts defined on a virtual host or a symbolic link can cause the test to fail.

Show Last Test Reports

The last Test Results summary is displayed in the Testing environment.
To display last Test Reports, click the Show Last Performance Test Report or Show Last Compression Test Report button. The displayed Test Report will be updated as soon as you run another test.

Note:
Test results can also be sent by e-mail and stored outside Zend Platform

Event Trigger Settings and Analysis

Initially Zend Platform Event settings are based on predefined default parameters. As such these settings require calibration to suit specific environments. The anticipated results are Event Details generated based on default triggers. Running Zend Platform for the first time will most probably cause abnormal event generation behavior (too many or too little events).

This behavior is attributed to two causes:

1. The Default triggers need to be calibrated.
2. The PHP needs to undergo additional performance Optimization.

Before making any adjustments to the PHP code that may turn out to be unnecessary, first calibrate PHP Intelligence.

Calibrating Event Triggers for Performance Optimization

Calibrating PHP Intelligence is the process of adjusting Event Triggers to accommodate a specific Web application.
Event Triggers are calibrated from PHP Intelligence | Event Triggers.
When calibrating Event Triggers for performance optimization, the following performance-related event types should be addressed:

- Slow Script Execution (Absolute and Relative) – generates an event when script execution exceeds defined limits
- Slow Query Execution – generates an event whenever database related functions exceed the threshold defined in the event.
- Slow Function Execution - Generates an event when a specified PHP function’s execution time exceeds the threshold defined in the event.
- Excess Memory Usage (Absolute and Relative) - Generates an event when memory use for PHP script execution is above or below average.

These event types are performance related indicators. Each one of these events should be configured to generate an Event according to your environments performance requirements. Events generated following calibration indicate that certain areas of the application are not performing according to your requirements and need further investigation.

Investigating Performance Related Events

Performance related events are investigated when events continue to occur after initial calibration. The contents of Event Details provide in-depth diagnostic information and tools for investigating the occurrence. A generated event does not necessarily indicate a problem with the PHP Code, it can also indicate that the Event Trigger settings need to be adjusted or the PHP code should be reevaluated.

The most effective performance diagnostic tool is the Zend Studio Profiler. The Zend Studio Profiler is a Zend Studio component that can be employed on Event Detail information through Zend Platform's integration with Zend Studio.

Profiling PHP Code with Zend Studio

Running the Zend Studio Profiler on PHP code provides time-related snapshot of the Code's overall performance. Profiling uses the Zend Studio IDE tools, to analyze PHP code. When profiling, the event's information is transferred from Zend Platform to Zend Studio. This information includes all the information necessary to precisely recreate of the actual occurrence that generated the event. The Zend Studio Profiler is so accurate that this process is paramount to running the profiler when the event originally occurred. Profiler information is generated by, placing timers within the code and running them over and over. The profiling tool is able to build a "profile" of how fast or slow specific areas of the application will run.
The Profiler is activated through Event Details from the Zend Studio Diagnostics section by selecting the option, Profile URL.

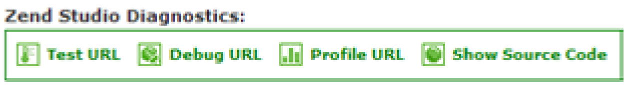


Figure 44 - Event Details - Zend Studio Diagnostics, Profile URL

The Profiling process takes place in the Zend Studio IDE and automatically opens the profiler results.

Note:
Zend Studio has to be installed and running to profile code. Also, the Server settings have to be configured in Zend Core to allow connectivity between Zend Platform and the Studio Server.

Understanding Profiler Results

Based on the information provided with the profiler, developers can identify the cause for the performance problem and implement changes to the code accordingly.

The Profiler user interface contains 3 tabs:

- Profiler Information - provides general information on the profiling duration and date, number of files constructing the requested URL and more. In addition, it displays a Time Division Pie Chart for the files in the URL.
- Function Statistics - provides you with the list of files constructing the URL and detailed information on functions in the files.
- Call Trace - provides a hierarchical display of functions according to process order, enabling you to jump to the function, view the function call, function declaration, details and more.

Note:
Additional information about the Studio Profiler can be found in the Zend Studio Online Help.

To perform in-depth examinations of slow code or functions the Zend Studio Debugger can be used to debug information in the occurrence's relevant context.
The Debugger is also activated through Event Details from the Zend Studio Diagnostics section by selecting the option, Debug URL.

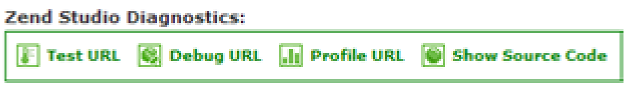


Figure 45 - Event Details - Zend Studio Diagnostics, Profile URL

This completes the description of the investigation and diagnostics tools that can be used to identify performance bottlenecks in the code. The next step is to see what performance tools can be applied to optimize the Web application's performance.

Performance Optimization Tools

Performance optimization tools are used once it is apparent the code is performing as it should, and the Event Triggers are calibrated.
From this stage on the performance optimization tools can be applied to further enhance Web application performance.
There are several levels of performance optimization that can be applied to files: Caching (full page), Acceleration and Compression.
Applying these three settings to your PHP code provides optimal performance boost.
Optimization Tools:
Caching, Acceleration and Compression

The default settings for these optimization tools are as follows:

- Caching - Default setting Off
  Description - Runs code and saves the output on the server
- Acceleration - Default setting On
  Description - Compiles the Code and saves the compiled code on the server
- Compression - Default setting Off
  Description - Saves a compressed version of the code on the server

To apply these optimization tools to all of the PHP files on the server caching and compression must be activated.
To activate/disable Caching, Compression or Acceleration, go to Performance | Settings and activate the enabling options.

When to Apply Optimization Tools

Now that we have established how to apply the optimization tools to "All Files" it is important to state that there are different circumstances that require disabling one or more of these features for select directories or files and in some cases altogether. The following section describes the possible optimization alternatives that should be considered when applying optimization tools.

Content Caching (Dynamic)

Dynamic Content Caching is the process of running code once and saving the output on the server for reuse in a set time frame (Cache Lifetime). Each time the code is requested, performance is improved by using the already run output instead of generating the same output each time.

When Should Files be Cached?

Files should be cached when their content is stable and does not require frequent changes.

When Not to Cache Files?

Caching is not recommended for files that have constantly changing output. For example: clocks, timers and database queries. (See Caching Alternatives to find out how to Partial Page Cache).

How to Cache Files

Caching by default is enabled (set to “ON”). In order to view/change the setting:

1. Click the Performance tab.
2. Pick a server to configure.
3. Click the Settings tab.

The dynamic caching enabled settings status will be displayed. Next, define the default caching settings. These settings are applied to all cached files.
To prevent unnecessary memory use, caching has to be actively applied to either a selected file or directory.

The performance module provides two options for caching files:

- Through the File view
- Through the Performance Test Report

Caching with the File View

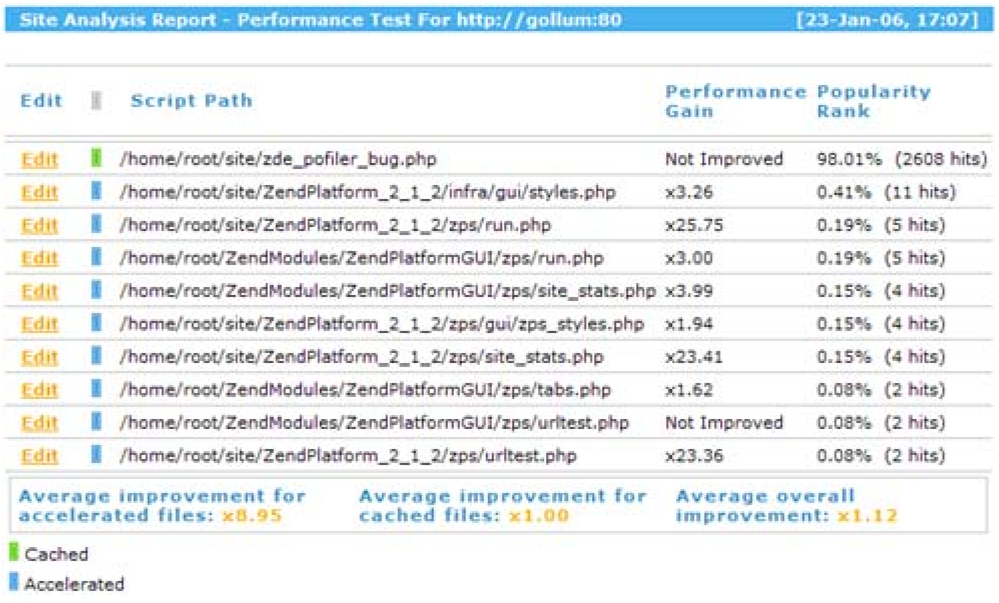
Caching can be applied to single files or do entire directories. Go to Performance | File View and choose one of the following options:

1. Apply Caching to a single file
2. Apply Caching to an entire directory.

Specific caching settings, given to files and directories, override the main settings defined in: Performance | Settings.

Caching with the Performance Test Reports

Performance Test Reports provide site analysis information in terms of performance gain and popularity. The information included in these reports provides a strong basis for evaluating if a file should be cached.



Site Analysis Report - Performance Test For http://gollum:80 [23-Jan-06, 17:07]

Edit		Script Path	Performance Gain	Popularity Rank
Edit	🟩	/home/root/site/zde_pofiler_bug.php	Not Improved	98.01% (2608 hits)
Edit	🟦	/home/root/site/ZendPlatform_2_1_2/infra/gui/styles.php	x3.26	0.41% (11 hits)
Edit	🟦	/home/root/site/ZendPlatform_2_1_2/zps/run.php	x25.75	0.19% (5 hits)
Edit	🟦	/home/root/ZendModules/ZendPlatformGUI/zps/run.php	x3.00	0.19% (5 hits)
Edit	🟦	/home/root/ZendModules/ZendPlatformGUI/zps/site_stats.php	x3.99	0.15% (4 hits)
Edit	🟦	/home/root/site/ZendPlatform_2_1_2/zps/gui/zps_styles.php	x1.94	0.15% (4 hits)
Edit	🟦	/home/root/site/ZendPlatform_2_1_2/zps/site_stats.php	x23.41	0.15% (4 hits)
Edit	🟦	/home/root/ZendModules/ZendPlatformGUI/zps/tabs.php	x1.62	0.08% (2 hits)
Edit	🟦	/home/root/ZendModules/ZendPlatformGUI/zps/urltest.php	Not Improved	0.08% (2 hits)
Edit	🟦	/home/root/site/ZendPlatform_2_1_2/zps/urltest.php	x23.36	0.08% (2 hits)

Average improvement for accelerated files: x8.95 | Average improvement for cached files: x1.00 | Average overall improvement: x1.12

🟩 Cached
🟦 Accelerated

Figure 46 - Performance Test Report

This screen shows if a file is cached and provides an option to cache a selected file from the list.

To Cache a file from the Site Analysis Report:

1. Go to Performance | Testing and select the Analyze Site Tab.
2. Run a performance test (or view the last performance test)
3. Go to the test results and press Edit. This will open the “Define Caching Conditions” dialog.

Caching Alternatives

Web pages that contain sections that continuously change can also be cached. This partial page caching solution can be accomplished through, applying caching APIs to portions of code that do not change. Partial Page Caching provides an intermediate solution for providing a partial performance boost that sustains the accuracy of changing content. To find out more about “Partial Page Caching” go to: “Tutorial - Partial and Preemptive Page Caching,”

Note:
Dynamic Content Caching can be deactivated from Performance | Settings and changing Dynamic Caching Enabled to Off. This will remove all Dynamic Content Caching settings from the files on the server. However, Partial Page Caching will not be affected. Partial Page Caching can only be disabled by, removing the Caching APIs from the code.

Code Acceleration

Code Acceleration is the process of gaining a performance boost by eliminating the code compilation time. Once 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 incurring a compilation lag each time the code is used.

Note:
Acceleration should not be confused with Caching. Acceleration saves the compiled script in the server’s memory whereas Caching saves the script’s output in the server’s memory.

When Should Files be Accelerated?

The general recommendation is to always use Code Acceleration to boost Web application performance. Therefore, the default setting for Acceleration is set to "On".

When Not to Accelerate (Blacklist)?

There are some instances where it is preferable to disable acceleration for select files. Acceleration is disabled by means of a Blacklist. Files should be added to the blacklist under the following conditions:

- Directories containing files that are larger than the Accelerators memory allocation or containing more files than the allocated quantity of files.
- Large files that have high memory consumption
- Files that have long execution time (makes the compilation save irrelevant).

Increasing Accelerator Memory Allocation

The alternative to blacklisting files is to increase the Accelerator memory allocation. The accelerator settings can be changed to increase allocated memory and the maximum quantity of files that can be accelerated. This alternative depends on the amount of memory available for allocation to the Accelerator.
When the Zend Accelerator is disabled, only cached scripts are tested.
To enable the Zend Accelerator, set the 'zend_accelerator.enabled' directive in the php.ini file to 'On'.

To change Accelerator memory allocation:

1. Go to Performance | File View
2. In the Code Acceleration section:
    a. Increase the Accelerator Memory
    b. Increase the Maximum Accelerated Files (default 2000)

Note:
If the memory fills up quickly, especially if there is only a small amount of Accelerated files. Increase the memory allocation or blacklist the file. Files exceeding allocated memory or quantity will not be accelerated.

Accelerator Duplicate Functions Fix

Some PHP code produces different opcodes for different situations, function defined or not. This causes a discrepancy for the accelerator in situations where the accelerator caches one version, and then a different situation occurs that requires a different function. If not addressed the script would just cease to work and raise a "duplicate functions" error.
To maintain proper performance in situations like these the zend_accelerator.dups_fix parameter should be activated. This parameter shuts down the Zend Accelerator’s duplicate function check, so that the errors will not occur.
This parameter belongs to The Zend Accelerator settings in the PHP Settings screen (Configuration | PHP Configuration | Extensions | Zend Platform).

Reset Accelerator

Programmatically resetting the Accelerator with accelerator_reset()
You can programmatically reset the Accelerator by calling the PHP function accelerator_reset() from within your PHP script.

Note:
Under certain circumstances, such as a busy server or complex PHP processes, it may take a few minutes to reset.

While Platform Performance Accelerator is being reset, a certain degree of server performance degradation takes place, since the accelerated scripts cannot be used during the reset period. The server will run as if Platform Performance had not been loaded.

Note:
Platform Performance Accelerator memory is also reset whenever the Web server is restarted.

Code Compression

Code compression is the process of using less bandwidth and increasing performance by compressing code before it is sent.

When to compress files

The default setting for file compression is Off. However, files should be compressed under the following conditions:

- When the Web application’s users are accessing the Web application with various low bandwidths that can benefit from the extra performance gained by compression.
- When there is a large quantity of Cached files.

When not to compress files

There are some instances where it is preferable to deactivate compression for select files. Compression can be deactivated in several ways:

- Deactivate compression entirely – should be done if the server is set to handle compression to prevent compressing files twice and rendering them unusable.
- Setting compression to cached files only – should be done when there is a large quantity of cached files and the rest of the files do not require compression.
- Blacklist – selectively disable compression for files do not require compression such as pictures that are already compressed or small files that do not require compression.
- When using PHP’s compression feature zlib.

Setting Code Compression

To change File Compression settings:

1. Go to Performance | File View
2. Select the appropriate Setting:
    a. None – disables Compression
    b. Only Cached Files – applies compression to cached files only
    c. All Files – collectively applies compression to all files (The list of files can be viewed in Performance | File View).

Zend Optimizer

The Zend Optimizer is a passive performance component that runs within the Zend Platform Framework and automatically optimizes scripts. Zend Optimizer is also designed to detect and load files encoded with the Zend Encoder.

Note:
The Zend Optimizer's default setting is "On" which means the Optimizer will start to run as soon as Zend Platform is installed. The Optimizer component does not require any additional configurations.

If you do not plan to use the Zend Optimizer to load encoded files, you can slightly improve the Optimizer's performance by adding the zend_optimizer.enable_loader = 0. This disables the transparent auto-loading mechanism that is built into the Zend Optimizer.

To change zend_optimizer.enable_loader settings:

Go to Configuration | Configure PHP Settings and choose from the list of directives Zend | Optimizer.

Tuning

The Tuning page is accessed from: Performance | Tuning.
This page allows users to define their Accelerator's performance level to increase PHP application performance. Tuning helps improve performance by disabling the following features: Performance (Caching, Acceleration and Testing), PHP Intelligence (Event collection) and Zend Guard (Obfuscation and Licensing).
Depending on the environment some of these features may not be necessary for example: in environments that do not deploy obfuscation and licensing the Zend Guard features can be disabled by using the Custom tuning level and selecting the options that disable licensing and obfuscation.

Tuning settings are applied to a selected node, the name of the server to which the changes will be applied is displayed beneath the top navigation bar (use the "Change Server" option to select a different server to tune).

There are four Accelerator Performance Levels:

- Normal - Preserves normal activity without disabling any functionality.
- Enhanced - Disables PHP Intelligence and Performance
- Extreme - Disables PHP Intelligence, Performance and Zend Guard
- Custom - Allows to independently select which out of the six options should be disabled.

To apply Tuning Settings:

Select an option from the "Accelerator Performance Tuning Level" drop-down and press Apply.

To view advanced options:

Click on the "Advanced Options" button to expand the list. A list of options with a detailed description will be displayed. Use this list to also view and read the descriptions of the options included under each tuning level.

To apply advanced options:

1. Select Custom from the "Accelerator Performance Tuning Level" drop-down.
2. Click on the "Advanced Options" button to expand the list.
3. Check the options from the list to apply then to the server.

Accelerator Performance Level Descriptions

Enhanced Accelerator performance tuning level

Applying this Accelerator performance tuning level impacts the following features:

- PHP Intelligence - New events will not be collected.
- Performance – File timestamps, consistency checksums and script timing will not be checked. Therefore, Accelerator tests will be disabled, and the Accelerator cache will not be refreshed.

Extreme Accelerator performance tuning level

Note:
This Accelerator performance tuning level would, in some cases, cause your environment to become unstable and even in-operable.

Applying this Accelerator performance tuning level impacts the following features:

- PHP Intelligence - New events will not be collected.
- Performance – File timestamps, consistency checksums and script timing will not be checked. Therefore, Accelerator tests will be disabled, and the Accelerator cache will not be refreshed.
- Zend Guard - Obfuscated pages will become inaccessible along with files using Zend Guard licenses.

Tuning Zend Platform for Optimal performance on i5/OS

The following procedure describes the actions that need to be performed to tune Zend

Platform for Optimal performance on i5/OS.

To tune performance, make sure that:

- Zend Optimizer 3.3.2 or above is installed. This version is available as part of the Zend Core 2.5 package or as an update for Core 2.0.X. (Download Zend Core for free from: http://www.zend.com/downloads)
- The directive ' zend_optimizer.disable_licensing ' is set to '1' (zend_optimizer.disable_licensing=1) in Configuration | PHP Configuration.
  Note: when set to Off, all code using Zend Guard Licenses will not be executable
- The directive 'zend_accelerator.use_cwd' is set to '0' (zend_accelerator.use_cwd=0) in Configuration | PHP Configuration.

Important Note:
When set to Off (0), performance will be increased however, some existing applications including Zend Core and the Zend Platform Administration Console will not be available.

- All modules are loaded and enabled. These by default are loaded and enabled out-of-the-box. Check to see they are loaded through the PHP Info tab (Configuration | PHP Info) or go to the Configuration tab to load the modules (Configuration | PHP Configuration). The modules are: Zend Debugger, Zend Download Server, Zend Optimizer and Zend Platform.
- The JobQ and Java Bridge daemons have been stopped.
  This is critical for weaker machines. To disable the daemons, Go to the i5/OS command line and type "go zendplat/zpmenu" Select the "Stop service menu" (option 2 ZPSTOP) and select the options:
  - o 3. Stop JavaMW server
  - o 6. Stop Job Queue Daemon

Set the Tuning Settings to Extreme (Performance | Tuning) to deactivate all non-critical functionality as follows:

- zend_accelerator.validate_timestamps=0 - Notes:
  - Not needed for PHP 5.2.x
  - When set to Off all changes to existing files require that you restart the Accelerator using 'accelerator_reset' to reset the contents of the Accelerator cache. Instructions on how to reset the Accelerator can be found in the section called " Code Acceleration".
- zend_accelerator.consistency_checks=0
- zend_accelerator.perform_timings=0
  Note: When set to Off, Zend Platform stops collecting acceleration info making the results in the option: Performance | Testing | Analyze Site incomplete.
- zend_optimizer.obfuscation_level_support=0
  Note: when set to Off, all code obfuscated with Zend Guard will not be executable.

Note:
Performance tuning is set to Extreme by default since version 3.0.3a.

Web Services

IN THIS CHAPTER

INTRODUCTION TO WEB SERVICES
GENERAL TASKS
GET/SET ACTIONS
ADD/REMOVE SERVERS ACTIONS
EVENT HANDLING

Introduction

Web services are a standardized way of allowing applications to interface and share data across the network. Web service messages are written in XML,
thus allowing for different applications in different programming languages to interface with each other.

Zend Platform provides an API that extends Zend Platform's functionality making it accessible via Web Services. Users will be able to get information and perform several tasks as follows:

- General Tasks - Login to the server and get information about the servers and services in your environment.
- Get/Set Actions - View and change directives.
- Add/Remove Server Actions - Add or Modify allowed host settings or remove hosts. These actions can be applied to nodes or server groups (clusters).
- Event Handling - Get Event information and Change Event Statuses.

Note:
Most of the functions check to see if the user is already logged-in to the system. Therefore, before calling a function call the login() function with the correct user information (user name and password).

System Requirements for Web Services

In order to run Web Services in your environment. Please make sure you have the following:

- PHP 5
- SOAP extension
- You must have the following zend.ini entry - zend_central.web_services.enabled=1 (this setting is set by the installer when choosing to use Web Services in the installation process).

General Tasks

ServiceResponse login

ServiceResponse login(username, password)

- Description: Used to login to the system with your given user name and password (the same user name used in order to log into Platform Administration).
- Return Values: ServiceResponse
- Parameters: string $username: Username to use for login. Same user name used in order to log into Platform Administration string $password: Password use for authenticating the user name

Important Note:
This function must be called before calling any other function that needs the user to be logged in.

ServiceResponse getServers()

ServiceResponse getServers()

- Description: Get a list of servers.
- Return Values: Returns a response object that will hold a list of all the clusters (and a list of their servers) and un-clustered servers (The term 'cluster' means a group of aggregated servers).
- Parameters: None

For example:

```
cluster1 =>     server1
server2
server5
cluster2 =>     server3
server6
server4 =>      null <- unclustered server
server7 =>      null <- unclustered server
```

ServiceResponse getVersion()

ServiceResponse getVersion()

- Description: Get the Web Service version to know what functionality it supports or not. (This function does not require login)
- Return Values: Returns a response object with the web service version.
- Parameters: None

Get / Set Actions

DirectiveServiceResponse getAllDirectives

DirectiveServiceResponse getAllDirectives(server)

- Description: Gets information about directives on a given server.
- Return Values: Returns an object containing all the directives read from the given server name and their relevant data (name, current value, loaded value, INI file the directive was read from).
- Parameters: string $server: Server alias of the server you would like to get directives from

DirectiveServiceResponse getDirective

DirectiveServiceResponse getDirective(server, name)

- Description:Gets the desired directive name and the server from which the directive value will be read.
- Return Values: Returns a response object holding the following information: directive name current value, loaded value, name of INI file the directive was read from.
- Parameters: string $server: Server alias of the server you would like to get directives from
  string $name: Directive name

ServiceResponse setDirectiveOnServer

ServiceResponse setDirectiveOnServer(server, name, value, ini_file, password)

- Description: Sets a new value for the given directive on the specified server. The user must specify the ini file this directive will be set in. The Password is only needed if the ini_file is php.ini.
- Return Values: None
- Parameters: string $name: Directive name to set
  string $value: New directive value
  string $ini_file: Name of INI file to save the directive in
  string $server: Server alias of the server you would like to set directives on
  string $password: Password for INI modifier (required only if INI file is php.ini)

ServiceResponse setDirectiveOnCluster

ServiceResponse setDirectiveOnCluster(cluster, name, value, ini_file, password)

- Description: Sets a new value for the given directive on all the servers belonging to the given cluster. The Password is only needed if the ini_file is php.ini. (assuming all servers in cluster use same password).
- Parameters: string $cluster: Server Group name
  string $name: Directive name
  string $value: Directive Value
  string $ini_file: Name if INI file to use in order to set the directive
  string $password: Password for INI modifier (required only if INI file is php.ini)

Add / Remove Servers Actions

ServiceResponse addAllowedHostOnServer

ServiceResponse addAllowedHostOnServer(server, host_ip)

- Description: Adds the given IP to the list of allowed hosts on the given server.
- Parameters:
  string $server: Name of the server on which to allow the host's IP.
  string $host_ip: Host IP to allow

ServiceResponse addAllowedHostOnCluster

ServiceResponse addAllowedHostOnCluster(cluster, host_ip)

- Description: Adds the given host IP to the allow_hosts directive in the zend.ini file (if it's not already there), on all servers in the given cluster
- Parameters:
  string $cluster: Name of the server group on which to allow the host's IP.
  string $host_ip: Host IP to allow

removeAllowedHostOnServer

ServiceResponse removeAllowedHostOnServer(server, host_ip)

- Description: Removes the given IP from the list of allowed hosts on the given server by adding it to the denied hosts list.
- Parameters:
  string $server: Name of the server on which to deny the host's IP.
  string $host_ip: Host IP to deny

removeAllowedHostOnCluster

ServiceResponse removeAllowedHostOnCluster(cluster, host_ip)

- Description: Removes the given IP from the list of allowed hosts on all the servers on the given cluster by adding it to the denied hosts list.
- Parameters: string $cluster: Name of the group we want to deny the host's IP access.
  string $host_ip: Host IP to deny

ServiceResponse isHostAllowed

ServiceResponse isHostAllowed(server, host_ip)

- Description: Checks if the given IP is in the allowed list on the given server.
- Parameters: string $server: Name of server to check if the host IP is allowed or not string $host_ip: Host IP to check

Event Handling

EventServiceResponse getAllEvents

EventServiceResponse getAllEvents(filter, offset, number_of_rows,sort_by,desc_order)

- Description: Gets a list of Events along eith a short version of the Event Details. It is also possible to get a range of event rows by specifying an offset and number of rows from that offset.
- Parameters: array $filter_array: Assoc. array where the key is the filter name and the value is the filter value integer $offset:
  integer $number_of_rows: Number of events to get
  string $sort_by: Value to sort by
  $desc_order: Ascending order (false) or descending order (true). Default is descending order.

ServiceResponse getTotalNumberOfEvents

ServiceResponse getTotalNumberOfEvents(fileter)

- Description: Shows the total number of events according to the given filter. This function should be used along with the getAllEvents() function in order to get the total number of events matching a given filter and show these events by pages
- Parameters: array $filter_array: Assoc. array where the key is the filter name and the value is the filter value

EventServiceResponse getEventData

EventServiceResponse getEventData(event_id)

- Description: Gets a specific event ID and returns all data available for the given event.
- Parameters: integer $event_id: The ID of the event you would like to get

getEventFilterNames

ServiceResponse getEventFilterNames()

- Return Values: Returns a list (array inside a ServiceResponse) of all available filter names that can be used with the getAllEvents() function.
- Parameters: None

getEventFilterAvailableOptions

ServiceResponse getEventFilterAvailableOptions($filter_name)

- Return Values: Returns all the possible options that the given filter name can accept (filter name is one of the elements returned by the getEventFilterNames() function.
- Parameters: string $filter_name:

getEventSortOptions

ServiceResponse getEventSortOptions()

- Return Values: Returns all the possible sort options for events

debugEvent

ServiceResponse debugEvent(event_id)

- Description: Starts a debug session while recreating the exact conditions that resulted in this event (same as the 'Debug' button in the 'Event Details' page).
  (This option works with Zend Studio)
- Return Values: Will start a debug session
- Parameters: integer $event_id: The ID of the event you would like to debug.

profileEvent

ServiceResponse profileEvent(event_id)

- Description: Profiles the script that originated the given event (same as the 'Profile' button in the 'Event Details' page).
- Return Values: Will start profile session
- Parameters: integer $event_id: The ID of the event you would like to profile

deleteEvent

ServiceResponse deleteEvent(event_id)

- Description: Use this function to remove events from the Database (same as the 'Delete' button in the 'Event Details' page).
- Parameters: integer $event_id: The ID of the event you would like to delete.

ignoreEvent

ServiceResponse ignoreEvent(event_id)

- Description: Changes the Event's status to Ignored (same as the 'Ignore' button in the 'Event Details' page).
- Parameters: integer $event_id: The ID of the event you would like to set as 'ignored'

closeEvent

ServiceResponse closeEvent(event_id)

- Description: Changes the Event's status to Closed (same as the 'Close' button in the 'Event Details' page).
- Parameters: integer $event_id: The ID of the event you would like to set as closed

preserveEvent

ServiceResponse preserveEvent(event_id)

- Description: Sets the Event's status to Preserved (same as the 'Preserve' checkbox in the 'Event Details' page).
- Parameters: integer $event_id: The ID of the event you would like to set to be preserved

unpreserveEvent

ServiceResponse unpreserveEvent(event_id)

- Description: Changes the status of a preserved so that it will be deleted in the next periodical Database cleanup.
- Parameters: integer $event_id: The ID of the event you would like to set to be un-preserved.

PART IV: ENTERPRISE SERVER

The Enterprise Server provides enterprise grade functionality for managing multi-server environments, ensuring interoperability and information consistency between nodes belonging to a cluster.

The Enterprise Server includes:

- Job Queues - Zend Platform's Job-Queue provides PHP production environments with a standard approach to streamline offline processing.
- Zend Download Server - A PHP (Zend Engine) plug-in which deals with serving large downloads which are served over the HTTP protocol.

Job Queues

INSIDE JOB QUEUES

INTRODUCTION
JOB QUEUES
JOB QUEUE API
JOB QUEUE TAB (JOB MANAGEMENT)
CREATING JOBS
QUEUES
JOBS
JOB DETAILS
JOB QUEUE SETTINGS

Introduction

Zend Platform’s Job Queues feature provides PHP production environments with a standard approach to streamlining offline processing.

Job Queues are serviced by the Job Queue Server and provide organizations with the ability to pre-schedule and determine Job activity to the level of even defining the processing server. Job Queues provide the necessary functionality to delay the execution of processes that are not essential during user interaction with the Web Server. Off loading non essential processes frees up your servers to provide a better user experience in terms of response time.

Requests can be off-loaded in one of two ways:

- By rerouting the requests to different Web server (Such as a server that does not perform customer-facing processes).
- By scheduling requests to be performed in low traffic hours to reduce the traffic on the Web server

Incorporating job management functionality into Zend Platform provides several obvious benefits such as the ability to manage queues in one-stop-shop fashion to managing your Web application environment from a "single point of access".

Along with these advantages, Zend Platform also enables you to:

- Manage Job fault and Performance using PHP Intelligence which fully monitors Jobs run by the Job Queue. Subsequently, whenever a Job generates a fault or performance degradation occurs, PHP Intelligence will generate a system Event containing the occurrence's details - information that can be later used to debug the occurrence.
- Generate real-time and historical reports for Jobs - Zend Platform offers Queue and Job reporting according to various criteria including: Current Jobs and Job History as well as current Queue load statistics that indicate any potential performance degradation.
- Leverage Job performances through code caching and content caching - Jobs can run under the Platform performance environment.
- Flexible installation options - Queues are optionally created during Platform Node installation. After installation the Queue becomes immediately available to Zend Central and can be configured and controlled from a central location.

Job Queues

Jobs are scheduled routines, that are processed at a certain point in time, whether it be time based or action based.
Zend platform's Job Queue functionality provides the means to run jobs in several ways to suit different needs using the following tools:

- The Job Queue API
- The Job Queue Tab in Zend Platform

Jobs can be created by directly adding functions to the Code using the Job Queue API (for more information see the User Guide, chapter, "APIs and Directives"). These Jobs are automatically detected by the Job Queue. Additional, simpler jobs can be created directly from the Jobs Tab by basically, defining the Jobs METADATA and directing to a script that needs to be run. Once Jobs are defined or identified by the Job Queue functionality you gain the ability to change, modify and reschedule the Job details and settings while storing historical information for future reference. Please see the Zend Platform User Guide for a complete description of the Job Queues API and its components.

The Job Queue API

The Job Queue API, contains a set of functions that enables to run a Job directly from your application's scripts. Running Jobs with the API, is beneficial in situations where a certain query has to be run in parallel with other actions. Such a situation could be in an online store where a user's credit card details have to be authenticated. In such a case the authentication process can be a Job that is activated when the user enters the credit card details. The authentication process, may take some time therefore, this job can be run separately on a separate server or even on a dedicated secure server (for credit card details). These Jobs can even be scheduled to run the authentication query at a later time when there is less traffic.

An example of this kind of script is as follows:

```
<?php
$job = new ZendAPI_Job('/usr/local/scripts/scrip1.php');
$job->setJobPriority(JOB_QUEUE_PRIORITY_HIGH);
$job->setUserVariables(array('name' => 'customer name', 'email' =>
'somebody@somewhere.com'));
?>
```

This script should be added to the application's script, at the point where you want to run a query. As we can see from this example, the new Job ($job) contains information indicating to the location of a script (scrip1.php). This script will contain the query that we want to run. Surrounding that script are different parameters that define the Job's variables, priority and other essential information. This information is also picked up by Zend Platform so that it can be viewed in the Jobs Tab eliminating the need to open numerous small files in order to find information relevant to managing Jobs.

Important Note:
All scripts managed by the Job Queue have to be placed in a set location. To view the default location or define a different location go to Job Queue | Settings and in the General Settings section view or change the Scripts Folder.

This naturally is only a simplified example of a basic Job that will run a query (script) and send an e-mail. In the Chapter on APIs and Directives we will show a complete description of the Job Queue API.

Additional Examples

Change the recurrence data of an existing job and add to it a dependency (assuming that the job id is known)

```
<?php
$job_queue = new ZendAPI_Queue('gollum.zend.office');
$job_queue->login('1234');
$job = $job_queue->getJob(8);
$job->setRecurrenceData(3600, time()+3600*24);
$job->setJobDependency(16);        // This job will perform only after job 16 was
succefuly executed
$job_queue->updateJob($job);            // The job queue will update job #8 with the
new properties of the given Job object
$job_queue->logout();
?>
```

Get all the jobs from a queue with urgent priority that waiting for process and belong to application id 8, change their priorities to LOW

```
<?php
$job_queue = new ZendAPI_Queue('gollum.zend.office');
$job_queue->login('1234',8);
$jobs = $job_queue->getJobsInQueue(
                array('priority' => JOB_QUEUE_PRIORITY_URGENT,
                      'status' => JOB_QUEUE_STATUS_WAITING
// we could also request for jobs in application id using the following line:
//'application_id' => 8
                )
    );
foreach ($jobs as $job) {    /* @var $job Job */
    $job->setPriority(JOB_QUEUE_PRIORITY_LOW);
    $job_queue->updateJob($job);
}
$job_queue->logout();
?>
```

Job Queue Tab (Job Management)

The Job Queue tab is accessed by clicking the Job Queues option. The Job queue configuration and management options include the following tabs:

- Queues - Displays Queue information and statistics.
- Jobs - A filterable display of Job information.
- Settings - Configure Queue settings.

Note:
In all of the tabs, the selected Queue name is displayed in the upper status line.

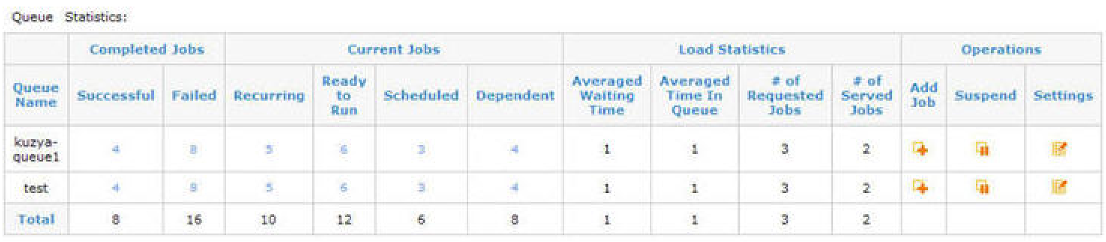
Queues

A queue is a logical container which contains a set of Jobs to be executed concurrently. Currently it is possible to setup a single Queue for a single machine.
The Queue page allows users to handle and manage queues across a cluster.
The Queue page is accessed from: Job Queues | Queues.

Through this screen users can:

- View Queue details and statistics.
- Queue Operations:

- Add Jobs.
- Suspend a Queue.
- View/Edit Settings.


Queue Statistics:

	Completed Jobs		Current Jobs				Load Statistics				Operations		
Queue Name	Successful	Failed	Recurring	Ready to Run	Scheduled	Dependent	Averaged Waiting Time	Averaged Time In Queue	# of Requested Jobs	# of Served Jobs	Add Job	Suspend	Settings
kuzya-queue1	4	8	5	6	3	4	1	1	3	2			
test	4	8	5	6	3	4	1	1	3	2			
Total	8	16	10	12	6	8	1	1	3	2			

Figure 47 - Queue Statistics Page

Queue Details and Statistics

Queue Statistics table displays information regarding a specific queue.
The table's columns are grouped into three different groups pertaining to different information as follows:

- Completed Jobs - Displays the number of completed jobs, divided into Successful and Failed:

- Successful - Number of Jobs belonging to the queue that ran successfully.
- Failed - Number of Jobs belonging to the queue that ran but failed.

- Current Jobs - Displays the number of Jobs that are currently being processed, by type. The types are:

- Recurring - Number of recurring jobs belonging to the queue.

Note

All recurring jobs are counted in this column regardless of their execution status i.e. including 'Ready to Run' and 'Waiting' jobs.

- Ready to Run - Number of jobs belonging to the queue that are ready to be executed.
- Waiting - Number of jobs belonging to the queue that are waiting to be executed.

Note:

Jobs with the status Scheduled (i.e. waiting to future schedule time) and Dependent (i.e. waiting for a previous job to be completed) jobs are counted in this column.

- Load Statistics - Displays statistics regarding job occurrences such as jobs that entered/exited the queue etc. The Statistics can be defined to display information on occurrences in the last X minutes. To define the duration go to Job Queues | Settings and set the time in the field "Save statistics history". This part of the table displays four types of statistics for queues:

- Average Waiting Time - The average time it took for all of the jobs to move from the 'Ready to Run' status to 'Running'.
- Average Time in Queue - The average time it took for all of the jobs to move from the 'Running' status to Successful/Failed.
- # of Requested Jobs - The number of jobs that have exited the queue (either successful or failed).
- # of Served Jobs - The number of jobs that have entered the queue (i.e. added to the queue).

The bottom row of the table displays the statistical accumulation of the entries in the table.

Queue Additional Details

Clicking on a row will automatically open the Jobs page which will display the Jobs related to the selected queue. The selected queue will also be remembered for further operations so that when entering the Settings page the settings for the previously selected will be displayed.

Queue Operations

The Queues table has an additional column that includes the different actions that can be performed on a queue as follows:

- Operations - The possible actions that can be performed on a queue. the operations are as follows:

- Add Job - Add a new job to the queue. Clicking this button opens the 'Job Details' page in add mode.
- Suspend - Suspend the entire Queue (Jobs belonging to a queue can be individually suspended by clicking on the Queue and modifying the queues displayed in the jobs page).
- Settings - View and Edit Queue settings.

Add Jobs

Clicking the "Add Jobs" button in the table opens the Job Details screen in edit mode. To add a Job enter the Job information and click " Save" (For more information on creating new Jobs see Job Details).

Note:
Jobs can only be added if there is an existing Job script located in the "Scripts folder" you defined in the Job Queue | Settings screen.

Suspend a Queue

Clicking the "Suspend" button in the table changes the status of the Queue to Suspended. In this status, all Jobs belonging to the Queue will not be executed and running jobs will be completed and only then suspended. Clicking this button again will toggle the status of a suspended queue to un-suspended and the Jobs belonging to this queue will be resumed at the next possible time (for recurring jobs).
New Jobs can be added to a suspended queue however, they will not be executed until the queue is un-suspended.

View / Edit Settings

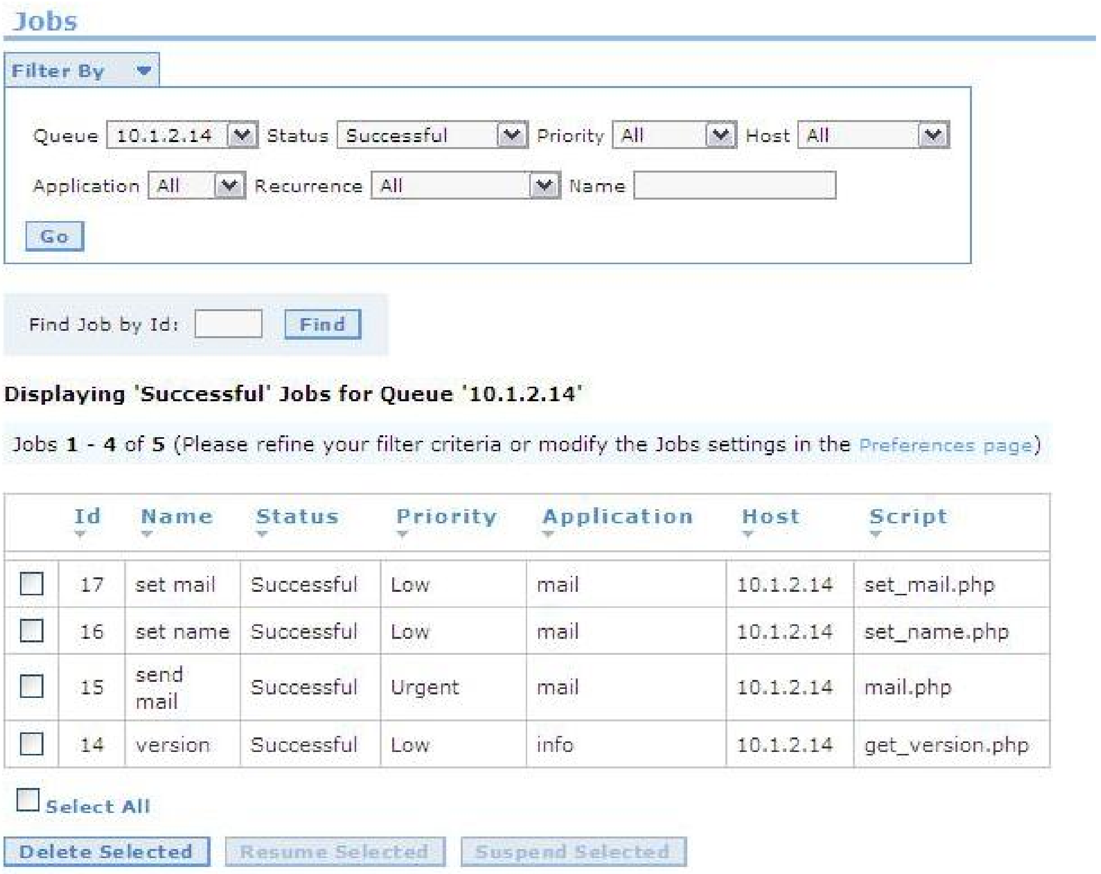
Clicking the "Settings" button opens the Job Queue Settings screen and loads it with the selected queues settings (For more information on viewing and editing queue settings see Job Queue Settings)

Jobs

A Job is a specific task that can be independently executed. Using the Zend Platform's Job Queue feature, Jobs can be handled in queues providing an easy way to manage and handle Job execution. The Jobs screen is accessed from: Job Queues | Jobs.

Through this screen users can:

- Filter Table Data
- Locate Jobs by their ID number
- View Job Details in a sortable table
- Manage Jobs


Jobs
Filter By
Queue 10.1.2.14 Status Successful Priority All Host All
Application All Recurrence All Name
Go
Find Job by Id:
Find

Displaying 'Successful' Jobs for Queue '10.1.2.14'

Jobs **1** - **4** of **5** (Please refine your filter criteria or modify the Jobs settings in the Preferences page)

	Id	Name	Status	Priority	Application	Host	Script
☐	17	set mail	Successful	Low	mail	10.1.2.14	set_mail.php
☐	16	set name	Successful	Low	mail	10.1.2.14	set_name.php
☐	15	send mail	Successful	Urgent	mail	10.1.2.14	mail.php
☐	14	version	Successful	Low	info	10.1.2.14	get_version.php

Figure 48 - Job Queue Screen

Filtering Table Data

Zend Platform allows you to filter the Jobs displayed in the Jobs table.
The filter options are as follows:

- Queue - The name of the Queue
- Status - The statuses are: Scheduled, Dependant, Ready to Run, Running, Suspended, Successful and Failed.
- Priority - High, medium or low.
- Host - The host on which the Job is to be executed.
- Application
- Recurrence - Jobs can be nonrecurring (run once) or recurring (defined to be executed repeatedly).
- Name - The alias for the Job script. Defined when creating a new Job or in Code based Jobs defined in the function: setJobName($name), if left empty the filter by script name.

The status line (above the filter by options) changes to show a summary of the items that are currently displayed in the table.

To filter the data displayed in the Jobs table:

1. Select the filter criteria to apply to the table by clicking "Filter By" and selecting the options from the drop-down lists..
2. Click "Go" to display the filtered events in the Jobs table.

The Jobs Table displays the following information:

- ID - a unique identifier given to each Job managed through Zend Platform.
- Name - an alias for the Job script. Defined when creating a new Job or in Code based Jobs defined in the function: setJobName($name), if left empty the script name will be displayed.
- Status - the current state of the Job (see list of statuses below)
- Priority - the importance of the job, the priorities are (in order of importance) Low, High, Normal, Urgent. Defined when creating a new job or in code based Jobs defined in the function: $_priority = JOB_QUEUE_PRIORITY_[TYPE];
- Application - an additional identifier for grouping and filtering Jobs. Defined when creating a new Job or in Code based Jobs defined in the function: $_application_id = null, if left empty no name will be displayed.
- Host - the name of the Host on which the Job was generated.
- Script - the name and location of the actual script initiated by the Job.

Job statuses

A Job can hold one of the following statuses:

- Scheduled - a Job which is waiting to be executed since it has been scheduled to a time later than the current time.
- Dependent - a Job which is waiting to be executed since it waits for a previous Job to finish running.
- Ready to run - a Job which is ready to be executed.
- Running - a Job which currently runs.
- Suspended - a Job which can run but is currently suspended (paused). A Suspended Job can be resumed and hence moved to another status.
- Successful - a Job which has completed and resulted in a success.
- Failed - a Job which has completed and resulted in a failure.

Note:
The maximum count of Jobs displayed in this page is configured in the Preferences page: Job Queues | Settings.

Locating Jobs by Job ID

Each Job is given a unique ID number when it is created. If you know the Job ID of a specific Job you want to view, enter the number into the "Find Job by ID" field and click Find to display the Job in the Jobs table.

View Job Details

The Jobs table displays various details regarding the displayed Jobs.
Apart from the basic information displayed users can click on a Job in the Jobs table and open the Job Details Page (for more information on Job Details see Job Details).

Manage Jobs

The Jobs tab includes the following Job management options:

- Delete - Remove a Job from the list. Jobs that are in progress will complete running and then be deleted.
- Resume - Resume a Job that was previously suspended.
- Suspend - Suspend a Job from running.

To apply one of these options to a Job, select a Job by marking the check box (you can select more than one) and click one of the Job management options (Delete, Resume or Suspend).
If you want to apply one of the options to all the Jobs in the table click "Select All".

Note:
When applying one of the management options to multiple Jobs make sure their current state does not conflict with the option for example, you cannot delete active Jobs.

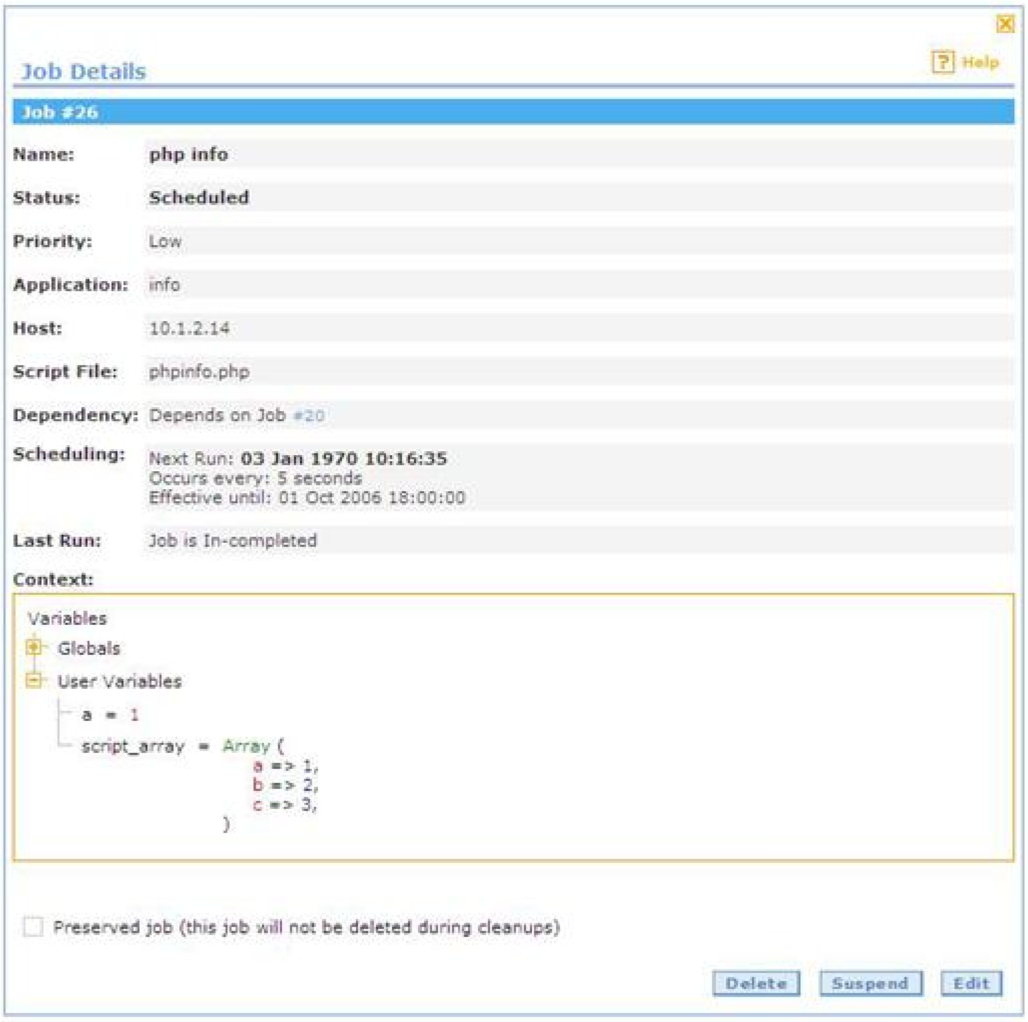
Job Details

The Job Details tab, displays a specific Job's properties. This page has two modes: Read-only and Edit.

To add Jobs go to: Job Queues | Queues and click "Add Job".

To view Job details go to: Job Queues | Jobs and click on one of the Jobs in the table.

To edit Jobs go to: Job Queues | Jobs, click on one of the Jobs in the table.



Job #26

Name:	**php info**
Status:	**Scheduled**
Priority:	Low
Application:	info
Host:	10.1.2.14
Script File:	phpinfo.php
Dependency:	Depends on Job #20
Scheduling:	Next Run: **03 Jan 1970 10:16:35** Occurs every: 5 seconds Effective until: 01 Oct 2006 18:00:00
Last Run:	Job is In-completed

Context:

```
Variables
 Globals
 User Variables
    a = 1
    script_array = Array (
                      a => 1,
                      b => 2,
                      c => 3,
                   )
```

☐ Preserved job (this job will not be deleted during cleanups)

Figure 49 - Job Details

Job Details Page Components

The following is a detailed description of the Job Details page components (Components marked with * can be edited in edit mode):

- Job Id - A unique identifier that is given to each Job. You can use this identifier to search for Jobs in Job Queues | Jobs by entering the ID into the "find Job by ID" field.
- Name - The name given to the Job by the user.
- Status - The statuses are: Scheduled, Dependant, Ready to Run, Running, Suspended, Successful and Failed.
- *Priority - High, low or medium.
- Application - The application running the Job
- Host - The host on which the Job runs.
- Script File -The location of the script for running the Job.
- *Dependency - Create a dependency on another Jobs completion. In edit mode, clicking here will open a pop-up with a "search Job by name" option.
- *Scheduling - View or define when the Job will run, recurrence details and effective date.
- Last Run - Details of when the Job last ran.
    - Show last job output - A link to HTML or Error details
- Context - Includes the following Variables:
    - *User Variables - Shows the name and value of the Job's user variables.
    - Globals - Shows the name and values of the Job's global parameters.
- Preserve (E) - if this flag is turned on, it means that the Job history will be saved in the history data even after the 'Time to Save History' time has passed. This is useful if the Job history needs to be saved.

Job Details Page Buttons:

The following actions can be applied to Jobs:

- OK - Closes the dialog.
- Delete - Opens a dialog asking whether the user is sure he wishes to delete the Job. Clicking "No" will close the dialog, and clicking "Yes" will delete the Job and the Jobs page would be displayed. Deleting a Job would delete its instances from the historical data as well.
- Suspend/Resume - this is a toggle button, it will be active for suspended Jobs and disabled when a Job is not suspended.
  (This button is only displayed when Jobs are still running)
- Edit - Displays the same page in the "Edit" mode.
  (This button is only displayed when Jobs are still running)
- Re-queue - Displays the same page in "Edit" mode. As a result users will be able to re-set the scheduling data, and then re-queue the Job (add it back to the Queue).
  (This button is only displayed when Jobs are still running)

The following buttons are only displayed in Edit mode:

1. Save - Saves the Job's data.
2. Cancel - Closes the Job Details page without saving changes.

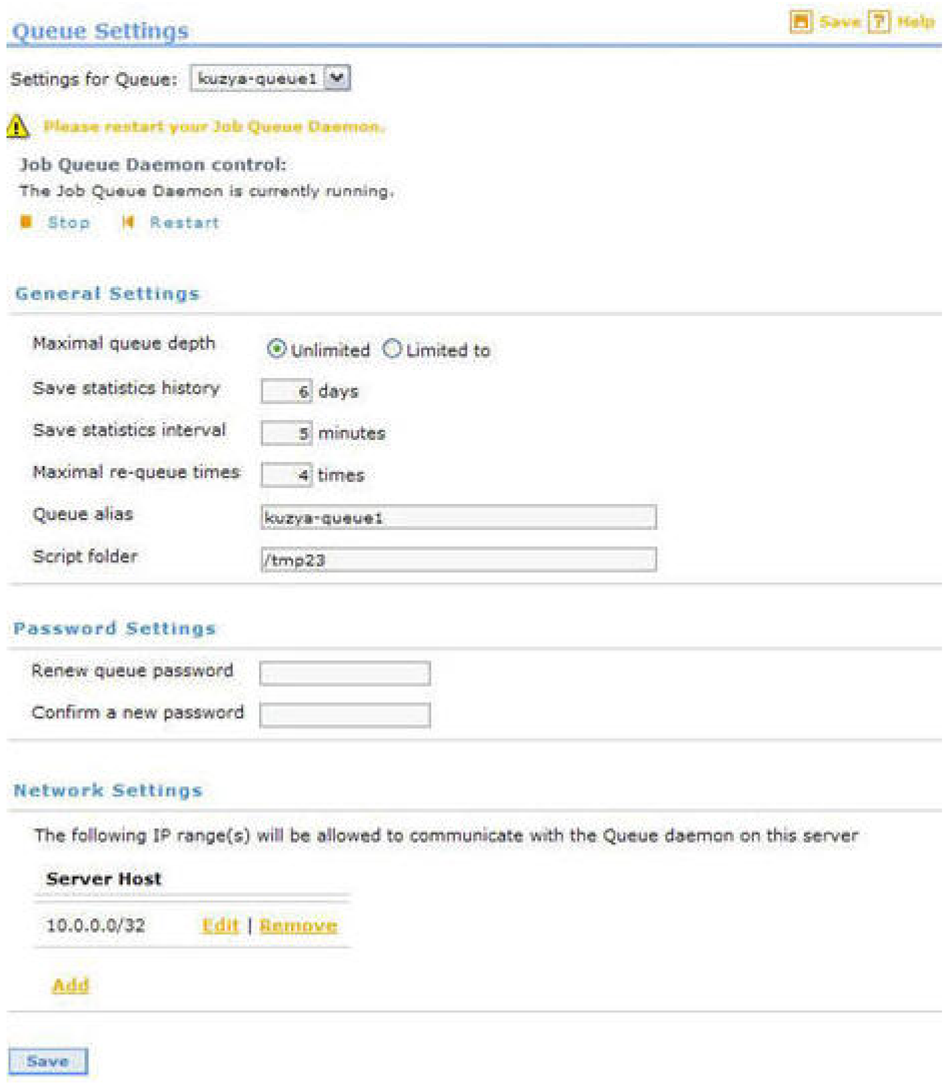
Job Queue Settings

The Job Queue Settings page displays settings associated with a specific Queue.

The Job Queue Settings page is accessed from: Job Queues | Settings.

Through this screen users can:

- View Queue settings.
- Edit Queues



Settings for Queue: kuzya-queue1

Please restart your Job Queue Daemon.

Job Queue Daemon control:

The Job Queue Daemon is currently running.

General Settings

Maximal queue depth	[✓ Unlimited] Limited to
Save statistics history	6 days
Save statistics interval	5 minutes
Maximal re-queue times	4 times
Queue alias	kuzya-queue1
Script folder	/tmp23

Password Settings

Renew queue password	
Confirm a new password	

Network Settings

The following IP range(s) will be allowed to communicate with the Queue daemon on this server

Server Host	
10.0.0.0/32	Edit | Remove

Figure 50 - Job Queue Settings

View Queue Settings

Queue settings are the specific definitions that define the Queue's behavior. There are two ways to determine which Queue will be displayed in the Settings screen:

1. Select a queue from the "Settings for Queue" drop-down list.
2. Jump to the Job Queue Settings tab from the Queues tab (Job Queues | Queues). Clicking the Settings button in the Queue Statistics table automatically opens the Job Queue Settings screen and loads the selected queues details.

Windows Vista Note:

The Start/Stop options are not available in Windows Vista. Use the services manager to Start and Stop services.

Queue Settings

The title of the Job Queue Settings tab always states the name of the Queue displayed in the tab. Typically, the recently-selected Queue (i.e., the one recently selected in the Queues page) will be selected in the drop-down list. If a Queue was not previously selected the user has to choose a queue from the "Settings for Queue" drop-down list. Displayed queues are remembered therefore users may switch tabs and when they return to the Job Queue Settings tab and the same queue will still be displayed.

The Job Queue Settings screen is divided into three sections:

1. General Settings:
   - Maximal Queue Depth - Specifies the amount of concurrent Jobs for the queue or leave it unlimited.
   - Save Statistics History - Specifies how much of the queue's history should be saved.
   - Maximal re-queue times - Specifies how many times a Job can be re-queued.
   - Sliding window - Collects Job Statistics
   - Queue Alias - Displays the Queue's name according to the name given in the installation process.
   - Script folder - Specifies the file name and location of the script the Queue should run.
2. Password Settings - Specifies the security settings to limit editing and deleting a Queue to authorized users only. Not entering a password will mean the specific Queue will not require authorization.
   - Enter New Queue Password - Specifies a password for limiting access to the Queue.
   - Confirm New Password - Confirm the password.
3. Network Settings - List of allowed hosts.
   - Add/Edit/Remove Server Host - Configure the IP's that are permitted to communicate with the Queue Daemon on this server.

Edit Queues

To edit a queue:

1. Go to Job Queues | Queues
2. In the Queue Statistics table, Click "Edit"
   The Job Queue Settings tab will open with the selected queue's information
3. Edit the Queue's settings and press Save.

Job Queue Server

The Job Queue Server is a container for Queues, it executes Queues according to a scheduling algorithm (e.g. Round Robin). Typically a machine only runs one Job Queue Server.
A Queue is a logical container which contains a set of Jobs. The Queue executes the Jobs and contains information about each Job. Each Queue is associated with a set of settings. For additional information, please refer to Job Queue Settings.

Creating Jobs

Jobs are scheduled routines, that are processed at a certain point in time, whether it be time based or action based.
Zend platform's Job Queue functionality provides the means to run jobs in several ways to suit different needs using the following tools:

- The Job Queue API
- The Job Queue Tab in Zend Platform

Jobs can be created by directly adding functions to the Code using the Job Queue API. These Jobs are automatically detected by the Job Queue. Additional, simpler jobs can be created directly from the Jobs Tab by basically, defining the Jobs METADATA and directing to a script that needs to be run.
Once Jobs are defined or identified by the Job Queue functionality you gain the ability to change, modify and reschedule the Job details and settings while storing historical information for future reference.

Zend Download Server (ZDS)

The Zend Download Server* is a PHP (Zend Engine) plug-in which efficiently deals with serving large downloads such as videos e.g. .mpeg files, binary products such as .exe and .msi files, and any other large files which are served over the HTTP protocol.
How it works: The Zend Download Server supports two modes of operation (both of which can be used together or separately according to your needs):

- Manual mode - The download is initiated by a PHP script using one simple PHP API function call. Not only does it allow you to serve files which aren't under your web server's document root but also it allows you to run logic such as access restriction checks before the download is started.
- Transparent mode - In your web server's configuration file you map the files you want to be sent through the efficient downloading mechanism to PHP, and the Zend Download Server will jump into action automatically and serve them.

More information about configuring and testing the ZDS can be found in the Zend platform User Guide.
*The Download Server is currently not applicable for Windows Operating Systems.

Configuring the Zend Download Server (ZDS)

(This feature is currently not applicable for Windows Operating Systems)
The ZDS (Zend Download Server) is a PHP (Zend Engine) plug-in. The purpose of this plug-in is to efficiently deal with serving large, downloads. This is done to preserve website performance levels when handling large downloads that are served over the HTTP Protocol and consume bandwidth. Downloads include, Video Files, Binary Products (such as .exe and .msi files), and other large files which are served over the HTTP protocol, and can potentially limit the performance of your website.

The ZDS provides two options:

1. Configure ZDS Settings
2. Test ZDS

ZDS functions in two modes:

- Manual Mode - Calling the API function zend_send_file() from PHP scripts.
- Transparent Mode – mapping file extensions to zend_mime_types.ini

Either mode can be run separately or in conjunction. Read on to find out how to configure the ZDS to run in either mode.

Manual Mode

In Manual mode, downloads are initiated by a PHP script that uses one all-purpose PHP function call. ZDS includes the PHP function zend_send_file(filename). Calling zend_send_file() immediately starts the file download and terminates your PHP script's execution. This effectively frees up the Apache process to handle the next incoming request.
The zend_send_file() function can also serve files that are not under the Web server's document root. Furthermore, it can be used to run logical functions such as access restriction checks, before downloads are started.

Usability Example

If a download function is called my_send_file($filename), you should integrate the zend_send_file() call in the following way in your source code:

```
if (function_exists("zend_send_file")) {
    zend_send_file($filename);
} else {
    my_send_file($filename);
}
```

Alternate Method

zend_send_file can also be set to accept a second argument, the mime type of the file. This will override the default mime type setting.

The parameters are: zend_send_file(string filename[, string mime_type]) and it would be called in the following way in your source code:

```
if (function_exists("zend_send_file")) {
    zend_send_file("/path/to/file.wma", "video/my-wma-type");
} else {
    my_send_file($filename);
}
```

Note:

If the mime_type is not specified or empty, the first mime type mechanism is used.

Manual Mode Usability Notes:

Do not create any output in Manual mode, before calling zend_send_file() - neither headers nor body - as this will interfere with the HTTP download.

Once you call zend_send_file(), the script terminates, so make sure all of your business logic runs before you call this function.

Sometimes files that are not under the same document root need to be served. Therefore, It is recommended to use the full path name to the file you want to serve. This will guarantee your script will work, even if you move it from your Web server's document root.

Transparent Mode

In Transparent mode, the file types that should be downloaded via ZDS are preconfigured, by mapping these files in the configuration file of your Web server. Files greater than the min_file_size directive will be automatically served by the ZDS.

To run ZDS in Transparent mode, make sure you meet the following requirements:

- The file extensions appear in the zend_mime_types.ini file and the file is mapped to the correct mime type.
  For example: to serve .mpeg files via the ZDS, add the following line in zend_mime_types.ini:
  video/mpeg mpeg
- In your Apache Server's configuration file, map the file type to PHP.
- For example, to map all .mpeg files to the ZDS in Apache by adding the following line to the Apache Server's configuration:
  AddType application/x-httpd-php .mpeg

Usability Note for Mac OS (In case of persistent problems with this OS please contact Zend Support.)

Mac Players cannot work when the file in the URL has .php extension.

There are three suggested solutions to this issue based on different possible requirements:

1) Use the following line where $new_name is the new file name.:

header("Content-Disposition: attachment; filename={$new_name}");

2) Rename the extension to WMA (and assign the WMA extension to PHP). This will enable these files. However, by assigning the WMA extension to PHP, ZDS would automatically parse it as a download. Therefore the WMA extension should be removed from the mime types file. Please note that, files will now be delivered with default content type, which might have effect on other players.

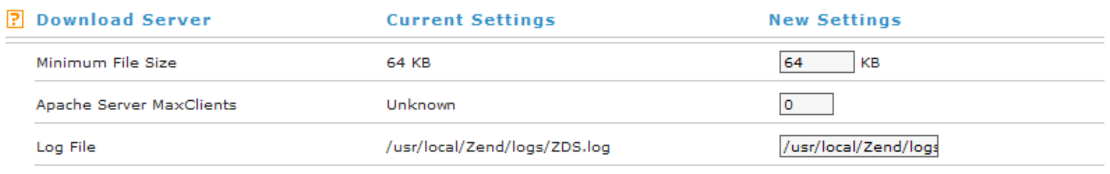
3) Add a condition to not auto-download these files, unless required by zend_send_file"

4) Add a parameter to the zend_send_file with the required mime type.

Both methods (manual mode and transparent mode) ensure that the Web application will continue to work even if, for some reason, you decide to temporarily disable the ZDS, (as long as the ZDS module was loaded).

To Configure the ZDS:

Go to Performance | Settings and go to the Zend Download Server Settings section of the Settings screen.



Download Server	Current Settings	New Settings
Minimum File Size	64 KB	64 KB
Apache Server MaxClients	Unknown	0
Log File	/usr/local/Zend/logs/ZDS.log	/usr/local/Zend/logs

Figure 51 - Zend Download Server Settings

The settings screen contains three general ZDS configuration settings:

- Minimum File Size - The minimum size of files that will be served by the ZDS. Small files need not be served by the ZDS, since performance gain is insignificant. Default: 64Kbytes.
- Server MaxClients - The testing tool (in Platform Administration) uses this value to determine your server's MaxClients. Keep this value updated to the actual number of max clients of your server.
- Log File - The name and location of the log file where the ZDS reports completed downloads. Default: <install_dir>/logs. Make sure the directory exists and that the user who starts the Web server (usually root) has "write" permission.

Server MaxClients Recommendation:

The MaxClients setting depends on your server hardware. To achieve accurate test results the server should be set between 50-150 MaxClients. The MaxClients value must be the same in the Download Server Settings and the Web server's configuration file.

These settings are applied to downloads handled in one of the two handling modes: Manual and Transparent

Testing the ZDS

Once the ZDS has been configured a test can be run to check and analyze the overall efficiency.

- The default ZDS test uses the Manual mode of operation to invoke a PHP script, which sends a file of approximately 300KB.
- The same test tool can be used to check the Transparent mode. Make sure that you correctly map the file type you are checking in your Web server's configuration file - according to the configuration instructions.

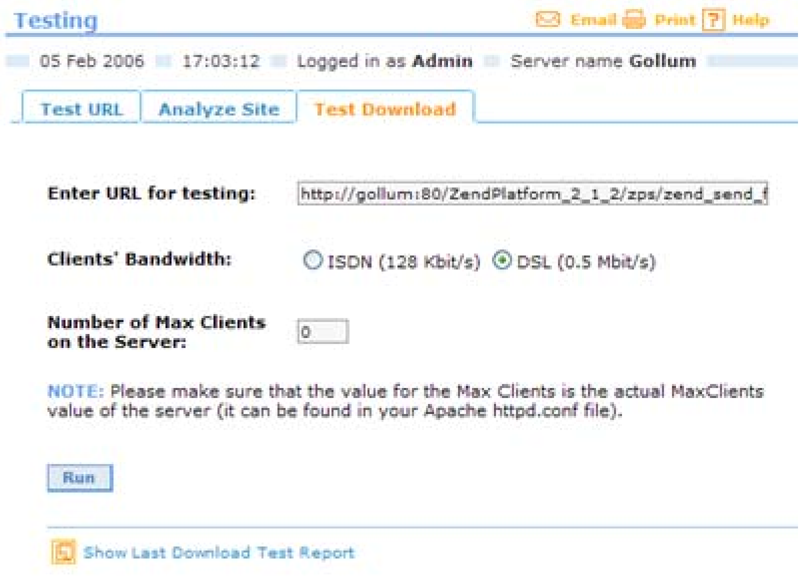
The test simulates multiple requests for a specified URL, with and without ZDS. There are three sets of tests, each test is performed twice (once with the ZDS disabled and once with the ZDS enabled). These tests differ in the number of concurrent clients that simultaneously perform requests to the server.

Note:
It is extremely hard to artificially test ZDS. The main reason is that testing it on a LAN can easily saturate your local network, and if your MaxClients is very high, Apache Benchmark (ab) may have difficulty handling the concurrency. For this reason, it is recommended to test ZDS with a relatively low MaxClients setting (e.g., 50-150) so that you don't reach any of these limits. The ZDS includes a version of ab, which was modified to support bandwidth throttling, which is used by the testing tool.

Caution:
During a test, your Web server will be fully loaded. A test can take several minutes so you should run it on a development machine or on an offline production machine.

To Test the ZDS:

Go to Performance | Testing and go to the Test Download tab.



05 Feb 2006 17:03:12 Logged in as **Admin** Server name **Gollum**

Enter URL for testing: http://gollum:80/ZendPlatform_2_1_2/zps/zend_send_f

Clients' Bandwidth: ISDN (128 Kbit/s) [✓ DSL (0.5 Mbit/s)]

Number of Max Clients on the Server: 0

NOTE: Please make sure that the value for the Max Clients is the actual MaxClients value of the server (it can be found in your Apache httpd.conf file).

Figure 52 - Performance - Testing – Test Download Tab

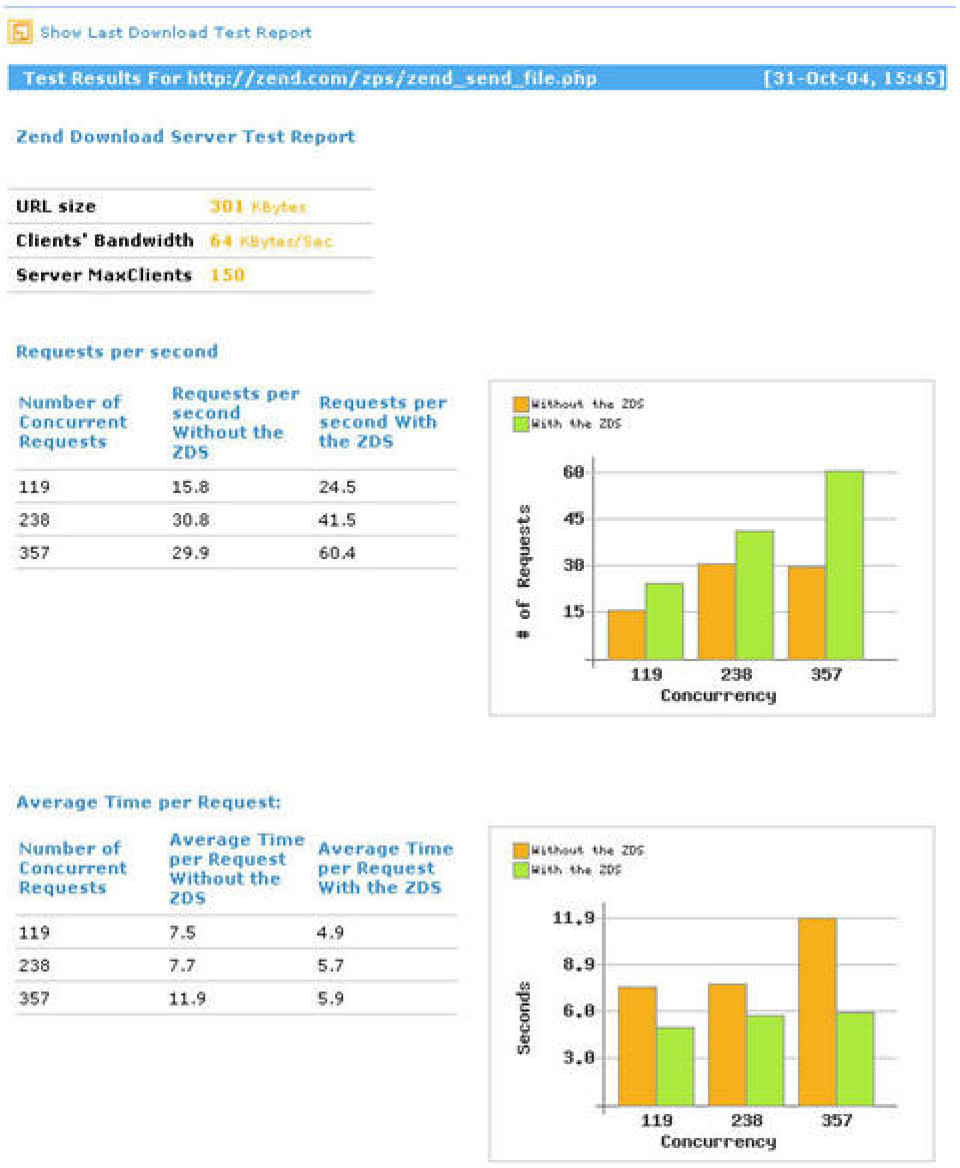
The Test Download Tab consists of two sections: The test options are on the upper section and the test results appear below (after running a test or displaying the last test results).

Running a Test

1. Type in the URL you want to test.
   The default test is a PHP script which uses zend_send_file() to send a 300K zip file (Testing very large files will take a very long time).
2. Choose the bandwidth limit you want to simulate for the clients. For a faster test, select a higher bandwidth (You cannot choose full bandwidth because your network card will be saturated, making the test irrelevant. The test tries to simulate a typical Internet server that has clients connected either by ISDN or DSL).
3. Enter the number of maximum clients that your server can handle.
   Use the precise value by checking the value of the MaxClients directive in your server's configuration file (The ZDS tries to identify your MaxClients value in the installation process, via the httpd.conf file, which is the default value. However, this value can be changed after the installation; and should be double-checked. Using an inaccurate MaxClients value, may not present accurate results).
4. Click Run.

Understanding Test Results

Once the tests have completed, you will see two tables and graphs with results that show Requests per Second and Average Time per Request for each test run.



Show Last Download Test Report

Test Results For http://zend.com/zps/zend_send_file.php [31-Oct-04, 15:45]

Zend Download Server Test Report

URL size	301 KBytes
Clients' Bandwidth	64 KBytes/Sec
Server MaxClients	150

Requests per second

Number of Concurrent Requests	Requests per second Without the ZDS	Requests per second With the ZDS
119	15.8	24.5
238	30.8	41.5
357	29.9	60.4

Average Time per Request:

Number of Concurrent Requests	Average Time per Request Without the ZDS	Average Time per Request With the ZDS
119	7.5	4.9
238	7.7	5.7
357	11.9	5.9

Figure 53 - Zend Download Server - Test Results

Disabling the Zend Accelerator changes the test configuration to cached scripts only.

Note:

If you do decide to run the ZDS Test on a production server, you can watch the log file to see how many concurrent jobs ZDS is handling. This indicates the number of Apache processes that would have been used if the ZDS were not installed.

Download Server Settings

Minimum File Size

The minimum size of files that will be served by the Zend Download Server.

Small files do not need to be served by the Zend Download Server (although they can be) since the gain is insignificant.

Default file size is 64Kbytes.

Apache Server MaxClients

This value is used by the testing tool in Platform Administration to determine the MaxClients of your server.

You should keep this value updated to the actual number of max clients of your server (in Apache, this value can be found in your httpd.conf file).

Recommended: For accurate test results set the server between 50-150 MaxClients (Do not forget to also change the web server configuration).

Log File

The name and location of the log file where the Zend Download Server reports completed downloads.

Note:

Make sure the file's directory exists and is writable by the user who starts the web server (typically root).

Default is /usr/local/Zend/Platform/logs/ZDS.log

Test Download

The Zend Download Server (ZDS) is a transparent process that runs in the background to service large downloads. The performance gain obtained by using the ZDS is measured with the Test Download option.

The Test Download option checks the efficiency of the Download Server. Test Download simulates multiple requests for a specified URL with and without the ZDS, thereby creating its own benchmark for comparison. This feature is currently not applicable for Windows Operating Systems.

To Run a Download Test:

Go to: Performance | Testing and choose the Test Download tab.

1. Enter the URL for Testing. (The default is a proprietary Zend PHP script which uses zend_send_file() to send a file.)

Note:

Testing very large files will take a very long time.

2. Enter the bandwidth limit you want to simulate for the clients (For a faster test, select a higher bandwidth).

Note:

Do not select full bandwidth, doing so will saturate your network card that will make the test irrelevant. The test tries to simulate a typical Internet server that has clients connected either by ISDN or DSL.

3. Enter the Max Clients value defined for your server. Insert the accurate value by checking the MaxClients directive in the httpd.conf file.

Note:
The ZDS installation tries to identify your MaxClients value via the httpd.conf file that is the GUI's default value, but this value can be changed after the installation, so it is important to verify that the value listed is correct.

4. Click "Run" to run the Download Test.

Viewing Download Test Results

Once the tests have completed, you can view the test results both statistically and graphically.

To view the results of the most recent Download Test:

From the Test Download Tab click, " Show Last Download Test Report". The test report includes tables and graphs that display the Requests per Second and Number of Concurrent Requests for each test run.

Note:
The Download Test is a simulation. The most meaningful test results are obtained by running ZDS on the production server and monitoring the log file to see how many concurrent jobs the ZDS is handling.

This completes the Performance Lifecycle chapter of the User Guide. In this section we have described the Performance Lifecycle and how implement the Zend Performance tools in creating optimal performance levels in an organization's environment.
Please refer to "Appendix C – Performance Lifecycle Check List" to read/print the Performance Lifecycle Check List that summarizes the performance optimization tasks.

PART V: INTEGRATION SERVER

The Integration Server includes features that allow Zend Platform users to integrate with external technologies and environments:

Zend Platform's Integration Server includes the following functionality:

- Java Bridge - used to access Java based applications running in a Java application server. Platform's Java Bridge offers significant performance and scalability advantages. Specifically, the memory consumption in the Platform PHP/Java Bridge is constant, regardless of the number of PHP sessions—unlike the equivalent solution, for example, in PHP 5.
- Actuate BIRT Reporting Integration - used to extract reports from Java libraries for generating reports (Uses the Java Bridge).
- SNMP Traps - An additional means of delivering Event information and parameters in a standardized way to a management console. The Event information includes details about occurrences (events) inside the system (For more information on SNMP, see Appendix G).

Java Bridge

IN THIS CHAPTER…

JAVA BRIDGE TAB
ABOUT ZEND'S JAVA BRIDGE TECHNOLOGY
OPERATING AND CONFIGURING ZEND PLATFORM'S JAVA BRIDGE
THE JAVA BRIDGE USER INTERFACE
USABILITY ISSUES
COMMON TASKS

The Zend Platform PHP/Java Bridge is a PHP module that connects the PHP object system with the Java object system. It can be used to access Java based applications running in a Java application server.
Platform's Java Bridge offers significant performance and scalability advantages. Specifically, the memory consumption in the Platform PHP/Java Bridge is constant, regardless of the number of PHP sessions—unlike the equivalent solution, for example, in PHP 5.

The PHP/Java Bridge feature should interest three types of enterprises:

- Companies that have investments in J2EE application servers can take advantage of PHP's Web-enablement capabilities, while preserving the utility of their Java investment.
- PHP-centric companies that want to take advantage of J2EE services that are not present in scripting languages—specifically, PHP.
- Companies that are not highly invested in J2EE and legacy systems can take advantage of Platform's PHP/Java Bridge to interact with plain Java objects.
- Companies that use or want to use Actuate reports.

About Zend's Java Bridge Technology

Zend's Java Middleware module (JavaMW) provides PHP connectivity to Java. The API is analogous to the standard PHP Java API (http://www.zend.com/manual/ref.java.php), however the implementation is different. JavaMW uses a stand-alone Java server process, which allows it to efficiently process Java requests. It adds stability and reliability to the PHP/Java connection. Unlike a standard PHP/Java connector, it uses a single Java virtual machine for all the requests, which makes memory and processor requirements significantly more modest while improving scalability.
The diagram below illustrates Zend Platform's Java Bridge technology:

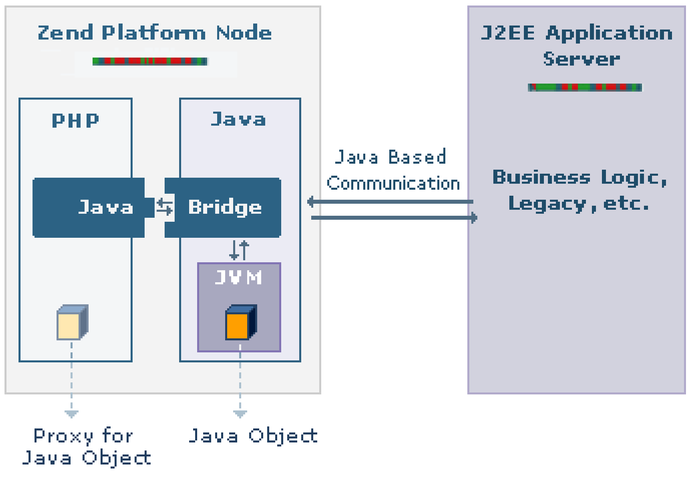


Figure 54 - Java Bridge Process Level

The Java Bridge Process Level diagram illustrates the following:

Zend Platform Node

Zend Platform Nodes include two bridging components: the PHP-side Bridge and the Java-side Bridge. Zend Platform Nodes operate as follows:

1. A JVM (Java Virtual Machine) is installed first—before installing the Platform Node—on the machine that is to be set up with Zend Platform.
   For the Java Bridge to function, you must install a compatible version of JVM. Platform will find the compatible version automatically. Supported versions are SUN J2SE 1.4 or SUN J2SE1.5 (J2SE 5).
2. Zend Platform then installs the two components required—the PHP-side and the Java-side—to create the Java Bridge.
3. A PHP application can call a Java object from any Java library that resides on the Node. For example, JVM can be downloaded with all its component libraries.

When a PHP application calls a Java object over the Java Bridge, a proxy for that object is created in PHP. In the diagram, the Java object is represented as a dark square; the proxy for that object in PHP is shown as a light square.

J2EE Application Server

The J2EE Application Server in its more advanced configuration, allows you to create a PHP/Java Bridge between a Zend Platform Node and an external J2EE Application Server. This type of configuration is typical of companies that have existing Java-based infrastructure. The J2EE Application Server operates as follow:

1. A PHP application can call a Java object from a Java library external to Zend Platform.
2. The Java-side Bridge component communicates with the J2EE Server. It finds objects in the J2EE Server, for example an EJB. The entire process is Java based.
3. The PHP application then calls the Java object over the Java Bridge created between the two Platform bridging components.
4. A proxy for that object is created in PHP. In the diagram, the Java object is represented as a dark square; the proxy for that object in PHP is shown as a light square.

The complete integration of Java and PHP is described in the following diagram:

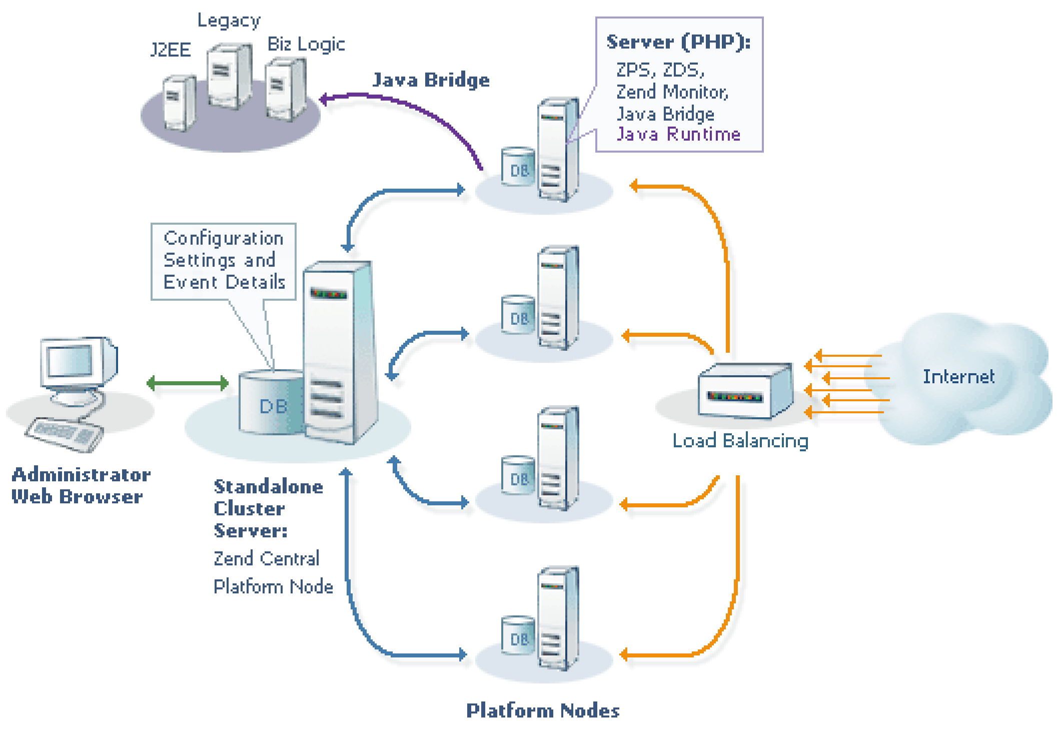


Figure 55 - Java Bridge System Level

The Java Bridge System Level diagram illustrates the following about the network architecture:

- Zend Platform Nodes - In order for a Zend Platform Node to function as a Java Bridge, it must have a properly functioning Java installation. Once Java is installed, Platform installation installs the required components for the Java Bridge, some of which are implemented in Java.
- J2EE Server - The Java-based enterprise that adds Zend Platform will have its own application servers. A J2EE Server is shown in the diagram as part of the Front Office. It can communicate with any of the Platform Nodes that have Java installed on them and which are defined in Java as legitimate accounts.

Added Value

Zend Platform's Java Bridge supports a PHP-Java integration that benefits enterprises on both the business and technical level.

Business Level Benefits:

- Companies with J2EE application servers can begin to realize the advantages PHP offers over other Web-enablement languages, including: shortened development time, shortened 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.

Technical Level Benefits:

- Platform's PHP/Java Bridge provides the ability to interact with plain Java objects.
- Platform's Java Bridge operates without the overhead of a JVM for each Apache process.
- Platform's Java Bridge consumes a finite amount of memory, which is almost disproportional to the amount of activity that's going through it.

Operating and Configuring Zend Platform's Java Bridge

This section describes procedures for operating and configuring Zend Platform's Java Bridge.
Configuring Run-time:
For running JavaMW, the following command can be used:

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

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

```
UNIX, Linux, i5/OS and Mac <install_dir>/bin/javamw.jar
Windows <install_dir>\bin\javamw.jar
```

Java Status Page

The Java Bridge tab provides status information about the Java servers connected to the network. The information displayed in the Java Status page, shows information about a selected server.
To access the Java Bridge tab go to Integration | Java Bridge.

The active buttons on the Java Status page are:

- Stop - Shuts down the Java Bridge daemon.
- Start/Restart - Restarts the Java Bridge daemon.
- Refresh - Refreshes the page for the selected server.
- Help - Opens the Online Help for the Platform Java Bridge.

Windows Vista Note:
The Start/Stop options are not available in Windows Vista. Use the services manager to Start and Stop services.

The Java Status page includes information about:

- Java Environment - The Java Environment includes the Java Version, Java Vendor, OS Name, OS Version, Class Path and Java Home
- Bridge Statistics - Bridge Statistics information includes the Number of connections and Requests.
- Number of connections - The accumulated number of processes holding a connection to the Java Bridge server (the full amount).
- Number of Requests - The number of 'worker threads' available for processing requests to the Java server.

Locating an Existing Java Version

Zend Platform assumes that your existing Java is installed in the standard location:
/QOpenSys/usr/bin/java.
If this is not the case, to use the Java Bridge you must relocate your Java to:
/QOpenSys/usr/bin/java.

Note:
The Zend Platform Java Bridge requires that you have SUN's JRE 1.4 or later installed on your computer. While installing Zend Platform you were prompted to direct the installer to the JRE's location. Therefore, you should already have JRE installed. If you did not choose to setup the Java Bridge in the installation process you can do so after the installation using the Setup Tool (from **UNIX, Linux and Mac**: platform_dir/bin/setup_tool.sh from **i5/OS**: GO ZENDPLAT/ZPMENU from **Windows**: Start | Programs | Zend Platform | Setup Tool).
More information about JRE's and the latest updates can be obtained from SUN Microsystems's website: http://java.sun.com.

Working with the Java Bridge User Interface

To view the Java Status Page for a selected server:

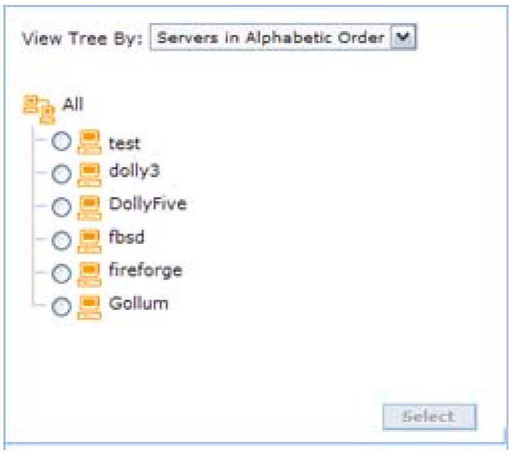
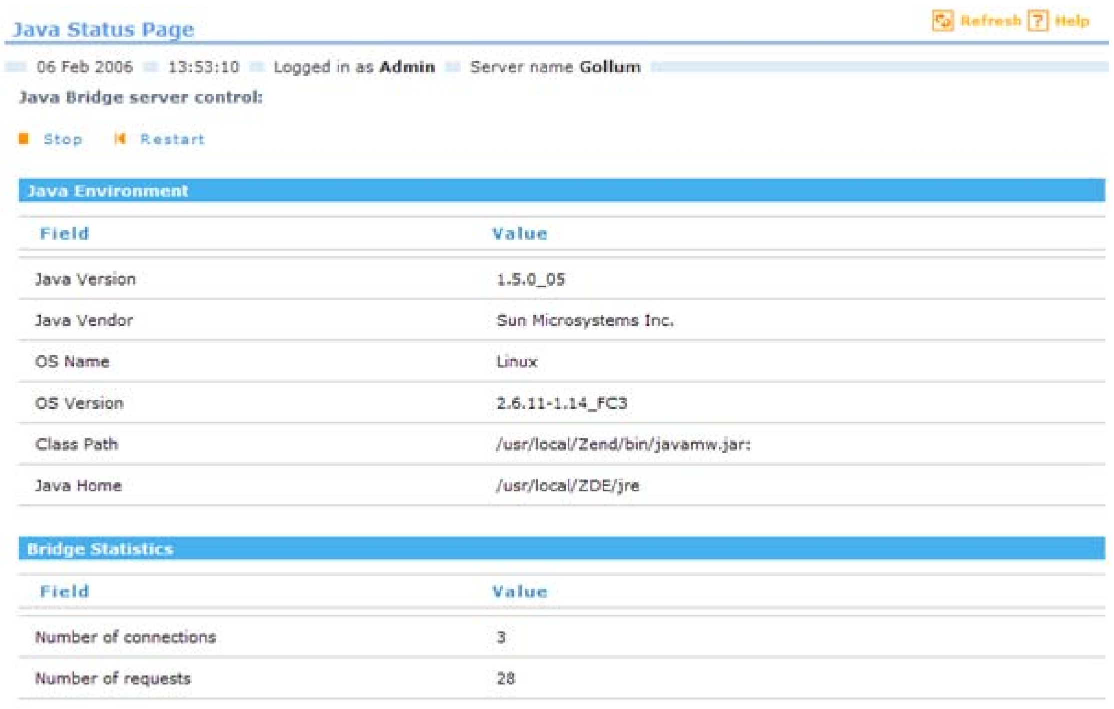


Figure 56 - Select Server to Configure

1. Go to the Java Bridge tab and the “Select Server to Configure” dialog opens.
2. Select a server from the list of servers in the Server Tree and click "OK". The Java Status Page opens for the selected server.



Java Environment

Field	Value
Java Version	1.5.0_05
Java Vendor	Sun Microsystems Inc.
OS Name	Linux
OS Version	2.6.11-1.14_FC3
Class Path	/usr/local/Zend/bin/javamw.jar:
Java Home	/usr/local/ZDE/jre

Bridge Statistics

Field	Value
Number of connections	3
Number of requests	28

Figure 57 - Java Status Page

Note:
Statistical information is gathered on the Java server. Therefore, even after restarting the Web server, the statistics are maintained. This naturally does not apply to restarting the Java server, which will restart the statistics collection from zero.

Using the command buttons provided on the Java Bridge user interface, you can stop the Java Bridge, start the Java Bridge, or refresh the Status Information shown on-screen, for the selected server.

To stop the Java Bridge:

1. Click "Stop".
   Platform opens an information screen that tells you that the Java Bridge was successfully stopped.
2. Click "OK" to close the window.
   Platform closes the window and returns to the Java Bridge main screen.

Configuring the Java Bridge

Zend Platform's Java module has two configuration parameters:

- zend.javamw.threads - Specifies how many worker threads the server is using; allowing this number of concurrent requests to be executed. The default value is 20.
- zend.javamw.port - Specifies the TCP port on which the server is listening. The default value is 10001.

Example Script

The following example is the shell script for running JavaMW (this script should be customized when necessary):

```
export CLASSPATH=$CLASSPATH:`pwd`/javamw.jar
java -Dzend.javamw.threads=20 -Dzend.javamw.port=10001 com.zend.javamw.JavaServer
```

Note:
Add other entries into CLASSPATH if you use non-standard Java packages.

PHP Configuration

The PHP module uses the following configuration directives:

- java.server_port - Specifies the TCP port on which the server is listening. The default value is 10001.

Note:
This must be the same as zend.javamw.port for the server.

- Java.ints_are_longs - converts PHP's integer to Java's java.lang.Long. By default and if this option is off, the PHP's integers are converted to java.lang.Integer.

Common Tasks

This section describes some of the common uses for the Zend Platform 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 Zend Platform's 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. Platform's Java Bridge enables enterprises to keep on using 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's 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. Platform's 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 Platform's 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 a Zend Platform node.
- J2EE is a structured framework for application scripts developed for J2EE. It is preferable that J2EE servers be left intact. (See the Zend Platform System Diagram above.)

Example 1: Typical Code

The code sample below is a functional example—you can run it! The example demonstrates the interaction between the PHP application and Java objects that occurs in the Java Bridge implementation.

```
<?
// 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";
?>
```

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 code creates the Java object—"java.lang.System"—which is the PHP proxy.
3. The purpose of the PHP code is to print the date and system information; however, it does so through the Java object.

Example 2: A Case Study Java Bridge Performance

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 web site 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 found 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 Platform’s Java Bridge applies to this business scenario and others like it:

```
<?
// 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.

The Typical Java Bridge Implementation and the Zend Platform's Java Bridge Implementation diagrams below show how Forever Times' concern about performance is addressed, through the Zend Platform Java Bridge architecture. The diagrams focus on how problems in scalability arise in a typical Java Bridge Implementation.

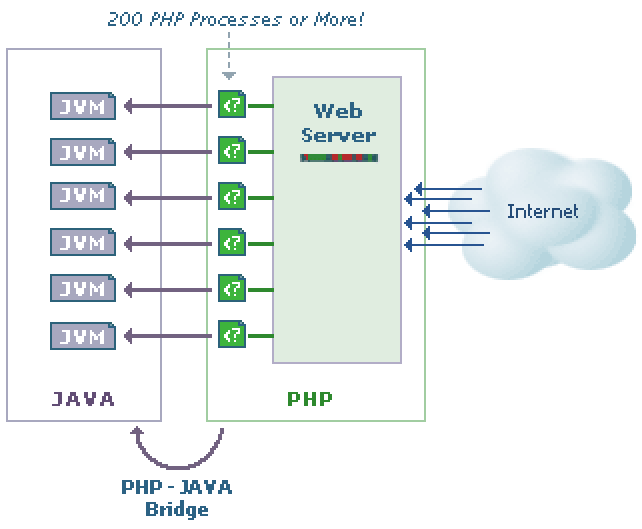


Figure 58 - Typical Java Bridge Implementation

Zend Platform's Java Bridge Implementation diagram shows how scalability issues are addressed in the Zend Platform Java Bridge.

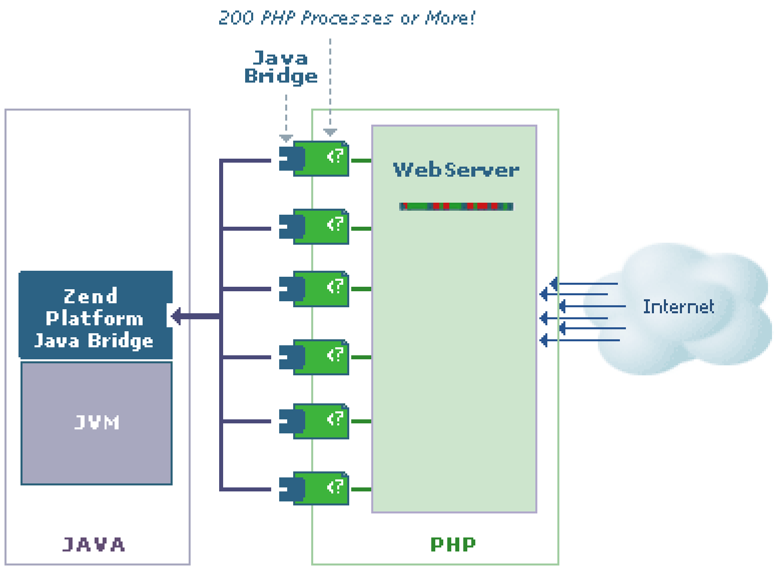


Figure 59 - Zend Platform's Java Bridge Implementation

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 Platform'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 3: Case Study in Management Integration

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, and its various enterprise applications were written in PHP.
Yourflowers.com 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.
Platform's Java Bridge offers a solution. Using Zend Platform, 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 Platform’s Java Bridge can apply to this business scenario and others like it:

```
<?
// 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"));
// $hw is our bean object
$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: “jnp://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.

Usability Issues

The Java Bridge's PHP 4 module has a number of usability issues compared to native Java code, which stem from the limitations imposed by the PHP 4 language. Most of these limitations do not exist in PHP 5 and those that do will be mentioned as such.

Chain Functions Call

In pure Java, you can program the following chain function:

```
result = object.Method1().Method2().Method3();
```

In this example, the result of one method becomes the object for another method. PHP 4's Java module, however, does not allow you to write a chain function in a similar way, as per example:

```
$result = $java_object->Method1()->Method2()->Method3();
```

This is due to the fact that PHP 4 disallows chaining method calls. Instead, a chain function must be expressed as follows:

```
$result1 =  $java_object->Method1();
$result2 = $result1->Method2();
$result = $result2->Method3();
```

Note:
In PHP 5, you can use chaining.

Exceptions

Since PHP 4 has no concept of exception, you may not include Java exceptions in your PHP code. However, you can use functions to deal with exceptions.

```
java_exception_get:
http://www.zend.com/manual/function.java-last-exception-get.php
java_last_exception_clear:
http://www.zend.com/manual/function.java-last-exception-clear.php
```

PHP 5 has a concept of exceptions and therefore can handle Java exceptions and translate them into PHP exceptions.

The following examples display the different exception scenarios and what they look like:

How Exceptions Work:

In this example exceptions are inherited from an exception class.

```
<?
try {
  $stack=new Java("java.util.Stack");
  $stack->push(1);
  $result = $stack->pop();
  print "$result\n";
  $result=$stack->pop();
} catch(Exception $ex) {
  print "Exception in pop: ";
  print $ex->getCause()->toString();
  print "\n";
}
?>
```

Caught Exceptions:

This example shows what an exception looks like when a Java Code exception is caught. This is an example of a typical exception that will appear instead of the expected PHP output when specified in the code i.e using print_r ($exception) or var_dump ($exception).

```
JavaException Object
(
    [message:protected] => Java Exception java.util.EmptyStackException:
java.util.EmptyStackException
        at java.util.Stack.peek(Stack.java:79)
        at java.util.Stack.pop(Stack.java:61)
    [string:private] =>
    [code:protected] => 0
    [file:protected] => /vector.php
    [line:protected] => 7
    [trace:private] => Array
        (
            [0] => Array
                (
                    [file] => /vector.php
                    [line] => 7
                    [function] => pop
                    [class] => java.util.Stack
                    [type] => ->
                    [args] => Array
                        (
                        )
                )
        )
    [javaException] => java.util.EmptyStackException Object
)
```

Uncaught Exceptions:

Uncaught exceptions are also reported but because they lack an immediate relation to a specific exception class they are less detailed and can only indicate basic details regarding the occurrence of an exception. This type of exception typically appears in your error log or wherever you have defined your php.ini to store errors.

Note:
Please refer to the PHP manual for more information regarding the php.ini eror reporting definitions.

```
Fatal error: Uncaught exception 'JavaException' with message 'Java
Exception java.util.EmptyStackException:
java.util.EmptyStackException
        at java.util.Stack.peek(Stack.java:79)
        at java.util.Stack.pop(Stack.java:61)
' in /vector.php:7
Stack trace:
#0 /vector.php(7): java.util.Stack->pop()
#1 {main}
  thrown in /vector.php on line 7
```

Java Array / Hashtable Objects

In PHP, arrays and hashtables are used interchangeably. This is because in PHP hashtables are indexed by integers or strings—not by objects. In Java, the key and value must be objects to be associated, so primitive types have to be converted to objects first, before parsing.

In Zend Platform's Java interface, if a method returns array/hashtable, it is immediately translated into a PHP native array/hashtable type. This means that if you want to work with a Java array/hash from PHP you cannot preserve it as a Java object. Of course, the contents are preserved, but the object identity is lost. In such a case, when an array/hash is returned, you will lose the ability to use Java methods since the array/hash loses the object identity and becomes a regular PHP array.

There are several ways to handle Java arrays and hashtable descendants. The following example shows a possible scenario of how Java arrays and hashtable descendants can be converted into PHP arrays by splitting the class pattern method and returning an array of strings which is then converted into a PHP array as follows:

```
<?
$exp = new Java("java.util.regex.Pattern");
$p = $exp->compile(":+"); // Create new patten object
$arr = $p->split("a::b:c:::d:e"); // Use pattern to split string into array
print_r($arr);
?>
```

To deal with Array / Hashtable objects originating in Java:

Implement the code dealing with the array in Java and then call it from PHP, or encapsulate the object in a different class.

Iterators

Iterators are not handled by the Java Bridge in any special way and they are treated like any other Java object.

This completes the Java Bridge chapter of the User Guide. In this section we have described how provide and configure Zend Platform in order to obtain interoperability with other existing legacy or backend applications written in Java.

BIRT Reports

INSIDE BIRT REPORTS

ABOUT BIRT REPORTS
THE BIRT REPORTS TAB
SETTING-UP THE BIRT REPORT ENGINE
BIRT APIS
ZEND PLATFORM BIRT REPORT EXAMPLES

Advanced reporting capabilities, have been integrated into Zend Platform, to provide enterprise users with expandable reporting functionality. Actuate's reporting application is the chosen application. Together with Zend Platform's Java Bridge it can extract reports from Java libraries and generate reports on any information. This solution is essentially a PHP API to the Actuate BIRT 2.0 run time environment that supports both PHP 4 and PHP 5.
To access the BIRT Reports tab go to: Integration | BIRT Reports.

About BIRT Reports

The BIRT reporting framework was developed by Actuate as a project under the eclipse foundation. The BIRT reporting system is a Java reporting tool for building and publishing reports against data sources ranging from typical business SQL databases, to XML data sources, to in-memory Java objects.

Types of reports that can be generated:

- Lists that include sorts, groups, totals, top N reports, averages and summaries
- Statements, invoices, documents, letters, forms and other business correspondence
- Charts including pies, lines, bars, gauges and many more formats

Actuate BIRT reports can combine text, images, rules, charts, tables and other elements in a single document or web page. Actuate BIRT also provides extensive support for the language needs of global application deployment. Using Actuate BIRT, a single report can display strings in various languages and can adapt date and numeric formatting and item widths to global languages.
Using BIRT reports with Zend Platform.

Zend Platform has included Actuate's report run-time environment allowing Zend Platform users to generate and run BIRT reports, directly form Platform Administration. In addition, Zend Studio XE includes the BIRT reports design interface (See the Zend Studio Online Help for more about creating BIRT reports). Both components complement each other in providing an overall solution for developing and creating business intelligence reports.

The BIRT Reports Tab

The BIRT Reports tab, is a BIRT report demo page.
Through this Tab, users can:

- View examples of BIRT reports.
- View exampled of the actual code.

Report Examples

The BIRT Reports tab includes four different examples of reports. These reports demonstrate different types of reports that can be created. To view the generated output of a report click "Render this code...".

The type of output you will see, depends on the output type defined in the code. The options are PDF or HTML (For more about creating reports see BIRT API)

Report Code

When selecting a report, the display beneath the report selection section changes to show the actual code of the selected report. This provides users with a detailed example of the makeup of reports including the Rendering option that allows users to also view the code's output.

Setting-Up the BIRT Report Engine

The Zend Platform installation script includes all the necessary components for rendering the BIRT use cases directly from Platform Administration. These use cases demonstrate the reporting capabilities in terms output types and the necessary functions used in order to gather and render the reports. Assuming the Java Bridge component has already been setup, the BIRT use cases will work out-of-the-box. The Java Bridge component is needed in order to support the interaction between the PHP scripts and the BIRT reporting tool written in Java.
If while installing Zend Platform, the option to setup the Java Bridge was not selected, the Setup Tool can be deployed to setup the Java Bridge.
Using the Setup Tool to Setup the Java Bridge:
Access the Setup Tool, from **UNIX, Linux and Mac**: platform_dir/bin/setup_tool.sh from **i5/OS**: GO ZENDPLAT/ZPMENU from **Windows**: Start | Programs | Zend Platform | Setup Tool.

Note:
The Zend Platform Java Bridge requires that you have SUN's JRE 1.4 or later installed on your computer. More information about JRE's and the latest updates can be obtained from SUN Microsystems's website: http://java.sun.com.

To enable the interaction between Zend Platform and the BIRT reporting tool, first activate the Java Bridge, by going to: Integration | Java Bridge and click "Start".

Zend Platform BIRT Report Examples

The following are the report examples set in the BIRT Reports Tab:

Note:
Please Refer to the chapter "APIs and Directives for more details about the BIRT APIs

Sales Invoice (html)

Prints an invoice for the selected order, including customer and invoice details and products ordered. Demonstrates use of a parameter to select the order to invoice and expressions for several calculated fields, including discount and order total. Uses expression to build customer address string and illustrates suppression of nulls in database fields with javascript function replace. Also shows image inclusion and sophisticated use of grids and tables to organize report content. Finally, the report makes use of styles to simplify maintenance and achieve a consistent look.

```php
<?php
// include Zend Birt report design API
require_once('Zend/Birt_Report/Zend_Birt_Report_Design.php');

// create report design object from rptdesign file
$birt = new Zend_Birt_Report_Design(dirname(__FILE__) . '/usecase1.rptdesign');

// set parameter sample, in this case:
// show only order number 10101
$birt->setParameter('OrderNumber','10102');

// "BIRT_TMP_DIR" represent a path to writable directory, while "birtImage.php?image=" is
// a php script that display the image from its original location
$birt->setImageConfiguration(BIRT_TMP_DIR, 'birtImage.php?image=');

// render a report.
// BIRT_REPORT_FORMAT_HTML - render an html report
echo $birt->renderReport(BIRT_REPORT_FORMAT_HTML, false);
?>
```

Top Selling Products (embedded html)

Displays a pie chart showing revenue by product line. Lists the top selling products, sorted by revenue. Demonstrates use of a chart and sorting a result set. Also shows image inclusion and use of grid and tables to organize report content. Finally, the report makes use of styles to simplify maintenance and achieve a consistent look.

```
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>Zend birt sample</title>
</head>
<body>
    <div style="width:600px;margin:0 auto;">
    <?php
    // include Zend Birt report design API
    require_once('Zend/Birt_Report/Zend_Birt_Report_Design.php');

    // create report design object from rptdesign file
    $birt = new Zend_Birt_Report_Design(dirname(__FILE__) . '/usecase2.rptdesign');

    // "$birt_tmp_directory" represent a path to writable directory, while "birtImage.php?image=" is
    // a php script that display the image from its original location
    $birt->setImageConfiguration(BIRT_TMP_DIR, 'birtImage.php?image=');


    // render html report embedded in html page
    echo $birt->renderReport(BIRT_REPORT_FORMAT_HTML, true);
```

```
        ?>

        </div>

    </body>

    </html>
```

Product Catalog (PDF)

Prints the Classic Models product catalog, grouped by product category. Provides product name, cost and description. Demonstrates one level grouping and using a grid within a table row to structure spacing. Also shows image inclusion and use of the tag in text item to include the content of a database column. Finally, the report makes use of styles to simplify maintenance and achieve a consistent look.

```
<?php

// include Zend Birt report design API

require_once('Zend/Birt_Report/Zend_Birt_Report_Design.php');



// create report design object from rptdesign file

$birt = new Zend_Birt_Report_Design(dirname(__FILE__) . '/usecase3.rptdesign');





// render report to PDF file and save it on a temporary folder

 if (!$birt->renderReportToFile(BIRT_REPORT_FORMAT_PDF, true, BIRT_TMP_DIR .
'/temp_report.pdf')) {

     // display error if renering return false

     echo $birt->getError();

 } else {

     // send pdf file to browser

     // if function for zend_send_file

     zend_send_file(BIRT_TMP_DIR . '/temp_report.pdf', 'pdf');

 }

 ?>
```

Product Catalog (using caching)

Prints the Classic Models product catalog, grouped by product category. Provides product name, cost and description. Demonstrates one level grouping and using a grid within a table row to structure spacing. Also shows image inclusion and use of the tag in text item to include the content of a database column. Finally, the report makes use of styles to simplify maintenance and achieve a consistent look.

```
<?php

// include Zend Birt report design API

require_once('Zend/Birt_Report/Zend_Birt_Report_Design.php');

// include Zend Birt report document API

require_once('Zend/Birt_Report/Zend_Birt_Report_Document.php');



// create report design object from rptdesign file

$birt = new Zend_Birt_Report_Design(dirname(__FILE__) . '/usecase4.rptdesign');



// caching a report using report document file

// savng rptdoc file to a temporary folder

$rptdoc_cache_file = BIRT_TMP_DIR . '/temp.rptdoc';

if (!file_exists($rptdoc_file)) {

    $birt_doc = $birt->generateReportDocument($rptdoc_cache_file);

} else {

    $birt_doc = new Zend_Birt_Report_Document($rptdoc_cache_file);

}

if(!$birt_doc) {

    die($birt->getError());

}



// "$birt_tmp_directory" represent a path to writable directory, while
"birtImage.php?image=" is
```

```
// a PHP script that displays the image from its original location

$birt_doc->setImageConfiguration(BIRT_TMP_DIR, 'birtImage.php?image=');


// rendering report from cache (rptdoc file) to output

$birt_doc->renderReportToOutput(BIRT_REPORT_FORMAT_HTML,false);

?>
```

PART VI: REFERENCE INFORMATION

Zend Platform APIs and Directives

IN THIS CHAPTER...

ZEND PLATFORM APIS:
- ACCELERATOR FUNCTIONS
- OUTPUT CACHE FUNCTIONS
- MONITOR FUNCTIONS
- ZDS (ZEND DOWNLOAD SERVER)
- JOB QUEUE
- BIRT REPORTS

ZEND PLATFORM DIRECTIVES:
- ACCELERATOR DIRECTIVES
- MONITOR DIRECTIVES
- PLATFORM ADMINISTRATION DIRECTIVES
- COLLECTOR CENTER DIRECTIVES
- DEBUGGER DIRECTIVES
  - ZDS DIRECTIVES

This chapter is a reference chapter for Zend Platform APIs and directives.

Zend Platform APIs

Accelerator Functions

accelerator_set_status

void accelerator_set_status(bool status)

- Description: Disable/enable the Code Acceleration functionality at run time.
- Return Values: none
- Parameters: status - if false, Acceleration is disabled, if true - enabled

Output Cache Functions

output_cache_disable()

- Description: Disables output caching for currently running scripts.
- Return Values: none
- Parameters: none

output_cache_disable_compression()

- Description: Does not allow the cache to perform compression on the output of the current page. This output will not be compressed, even if the global settings would normally allow compression on files of this type.
- Return Values: none
- Parameters: none

output_cache_fetch()

string output_cache_fetch(string key, string function, int lifetime)

- Description: Gets the code's return value from the cache if it is there, if not - run function and cache the value.
- Return Values: function's return
- Parameters: key - cache key, function - PHP expression, lifetime - data lifetime in cache (seconds)

output_cache_output()

string output_cache_output(string key, string function, int lifetime)

- Description: If they cache for the key exists, output it, otherwise capture expression output, cache and pass it out.
- Return Values: expression output
- Parameters: key - cache key, function - PHP expression, lifetime - data lifetime in cache (seconds)

output_cache_remove

bool output_cache_remove(string filename)

- Description: Removes all the cache data for the given filename.
- Return Values: true if OK, false if something went wrong
- Parameters: filename - full script path on local file system

output_cache_remove_url

bool output_cache_remove_url(string url)

- Description: Remove cache data for the script with given URL (all dependent data is removed)
- Return Values: true if OK
- Parameters: url - the relative path for the script

output_cache_remove_key

bool output_cache_remove_key(string key)

- Description: Remove item from PHP API cache by key
- Return Values: true if OK
- Parameters: key - cache key as given to output_cache_get/output_cache_put

output_cache_put

bool output_cache_put(string key, mixed data)

- Description: Puts data in cache according to the assigned key.
- Return Values: true if OK
- Parameters: key - cache key, data - cached data (must not contain objects or resources)

output_cache_get

mixed output_cache_get(string key, int lifetime)

- Description: Gets cached data according to the assigned key.
- Return Values: cached data if cache exists, false otherwise

- Parameters: key - cache key, lifetime - cache validity time (seconds)

output_cache_exists

bool output_cache_exists(string key, int lifetime)

- Description: If data for assigned key exists, this function outputs it and returns a value of true. If not, it starts capturing the output. To be used in pair with output_cache_stop.
- Return Values: true if cached data exists
- Parameters: key - cache key, lifetime - cache data validity time (seconds)

output_cache_stop

- Description: If output was captured by output_cache_exists, this function stops the output capture and stores the data under the key that was given to output_cache_exists().

Monitor Functions

monitor_pass_error

void monitor_pass_error(integer $errno, string $errstr, string $errfile, integer $errline)

- Description: Should be called from a custom error handler to pass the error to the monitor. The user function needs to accept two parameters: the error code, and a string describing the error. Then there are two optional parameters that may be supplied: the filename in which the error occurred and the line number in which the error occurred.

monitor_set_aggregation_hint

void monitor_set_aggregation_hint(string $hint)

- Description: Limited in the database to 255 chars, this API is a global variable that can be set anywhere and in any hierarchy. The purpose of this API is to incorporate locations of occurrences in the script. This API is used when there are events that require the location in the script for diagnosing the reason behind the event occurring. For example: Global Events require the application that generated the event. Adding the Hint API can assist in the identification process. This string that is supplied by the user to differentiate between pages that have the same URL but different parameters.
- Return Values: If the user did not supply a hint the default hint is an empty string.
- Parameters: $hint

monitor_custom_event

void monitor_custom_event(string $class, string $text[, integer $severe, mixed $user_data])

- Description: Custom Events are used to generate an event whenever the API function monitor_custom_event() is called from the PHP script. This event type enables the generation of an event on occurrences that are not necessarily built-in Zend Platform events (error and performance issues). Custom Events are used whenever you decide that it is significant to generate an event in a certain situation. Each event type is given a name for easy identification ($type).
- Parameters: $class – helps to define several types of custom events. This description will be showed in the PHP Intelligence, Event List and in the Event Details (report). $text - error text used to describe the reason for the event. This text will appear in the Event Details. $severe - the severity level of the triggered event, default value is Severe. $user_data - adds a PHP

variable that will be viewed in the Event Details screen (in Event Context-> Variables->User Defined). This forms the stored Event Context (similar to the information obtained in a PHP error event).

register_event_handler

boolean register_event_handler($event_handler_func [,$handler_register_name],$event_type_mask])

- Description: Allow you to register a user function as an event handler. When a monitor event is triggered all the user event handler are called and the return value from the handler is saved in an array keyed by the name the event handler was registered under. The event handlers results array is saved in the event_extra_data table.
- Return Value: TRUE on success and FALSE if an error occurs.
- Parameters: The first argument is a callback function that will be call when the event is triggered, object methods may also be invoked statically using this function by passing array ($objectname, $methodname) to the function parameter. The second (optional) argument is name this function is registered under - if none is supplied, the function will be registered under it's own name. The third (optional) parameter is a mask of event types that the handler should be called on by default it's set to MONITOR_EVENT_ALL.

unregister_event_handler

boolean unregister_event_handler(string handler_name)

- Description: Allow you to un-register an event handler.
- Return Value: TRUE on success and FALSE if no handler we registered under the given name.
- Parameters: string handler_name - the name you registered with the handler you now wish to un-register.

ZDS (Zend Download Server)

zend_send_file

bool zend_send_file(string filename[, string mime_type])

- Description: Send a file using ZDS
- Return Value: FALSE if sending file failed, does not return otherwise
- Parameters: filename - path to the file, mime_type - MIME type of the file, if omitted, taken from configured MIME types file

Note:
The ZDS is not currently supported in Windows.

Java Bridge

java_last_exception_get

object java_last_exception_get()

- Description: Return Java exception object for last exception
- Product Version - Buran
- Return Value: Java Exception object, if there was an exception, false otherwise

java_last_exception_clear

void java_last_exception_clear()

- Description: Clear last Java exception object record.

java_set_ignore_case

void java_set_ignore_case(bool ignore)

- Description: Set case sensitivity for Java calls.
- Parameters: ignore - if set, Java attribute and method names would be resolved disregarding case. NOTE: this does not make any Java functions case insensitive, just things like $foo->bar and $foo->bar() would match Bar too.

java_set_encoding

array java_set_encoding(string encoding)

- Description: Set encoding for strings received by Java from PHP. Default is UTF-8.

java_throw_exceptions

void java_throw_exceptions(int throw)

- Description: Control if exceptions are thrown on Java exception. Only for PHP5.
- Parameters: throw - If true, PHP exception is thrown when Java exception happens. If set to false, use java_last_exception_get() to check for exception.

Job Queue

A queue of job is described using the JobQueue class, when you want to manage a queue (or add more than one job to it) you should instantiate a JobQueue object.
The JobQueue object enables several job control functions such as: add/remove/suspend/resume, and some manage/info queue functions like: getJobsInQueue, getHistory, getStatistics etc.
A job is described by a Job class, whenever you want to add/update a job you can handle the Job object using the Job methods.
After a job is in the queue, you can retrieve it to change remove or suspend the Job by using the job id (the job id is assigned when a Job is added to the queue or by querying the jobs in the queue)
To add one job to a queue, for simplicity of usage, you can create the Job object (with it's required attributes) and then add the job directly from the Job object (without instantiating the JobQueue object), using the Job::addJob() function.
To change a Job's attributes, first get it from the queue (JobQueue::getJob() function), change the attributes and then update the queue with the changed Job object (JobQueue::updateJob() functions).

Global Functions and Constants

Constants for Job statuses

define('JOB_QUEUE_STATUS_SUCCESS', 1);	Job was processed and succeeded
define('JOB_QUEUE_STATUS_WAITING', 2);	Job is waiting to be processed (was not scheduled)
define('JOB_QUEUE_STATUS_SUSPENDED', 3);	Job was suspended
define('JOB_QUEUE_STATUS_SCHEDULED', 4);	Job is scheduled and waiting in queue
define('JOB_QUEUE_STATUS_WAITING_PREDECESSOR', 5);	Job is waiting for it's predecessor to be completed
define('JOB_QUEUE_STATUS_IN_PROCESS', 6);	Job is in process in Queue
define('JOB_QUEUE_STATUS_EXECUTION_FAILED', 7);	Job execution failed in the ZendEnabler
define('JOB_QUEUE_STATUS_LOGICALLY_FAILED', 8);	Job was processed and failed logically either because of job_fail command or script parse or fatal error

Constants for different priorities of jobs

define('JOB_QUEUE_PRIORITY_LOW', 0);
define('JOB_QUEUE_PRIORITY_NORMAL', 1);
define('JOB_QUEUE_PRIORITY_HIGH', 2);
define('JOB_QUEUE_PRIORITY_URGENT', 3);

Constants for saving global variable's bit mask

define('JOB_QUEUE_SAVE_POST', 1);
define('JOB_QUEUE_SAVE_GET', 2);
define('JOB_QUEUE_SAVE_COOKIE', 4);
define('JOB_QUEUE_SAVE_SESSION', 8);
define('JOB_QUEUE_SAVE_RAW_POST', 16);
define('JOB_QUEUE_SAVE_SERVER', 32);
define('JOB_QUEUE_SAVE_FILES', 64);
define('JOB_QUEUE_SAVE_ENV', 128);

set_job_failed

set_job_failed($error_string);

- Description: Causes a job to fail logically. It can be used to indicate an error in the script logic (e.g. database connection problem).
- Parameters: @param string $error_string the error string to display

jobqueue_license_info

jobqueue_license_info();

Description: returns an array containing following fields:

- "license_ok" - whether license allows use of JobQueue
- "expires" - license expiration date

Queue Class

```
class ZendAPI_Queue {
var $_jobqueue_url;
```

Queue Class Functions

zendapi_queue($queue_url) {}

Constructor for a job queue connection

- @param string $jobqueue_url - Full address where the queue is, in the form host:port
- @return zendapi_queue object

login($password, $application_id=null) {}

Opens a connection to a job queue

- @param string $password For authentication, password must be specified to connect to a queue
- @param int $application_id Optional, if set, all subsequent calls to job related methods will use this application id (unless explicitly specified otherwise). I.e. When adding new job, unless this job already set an application id, the job will be assigned the queue application id
- @return bool Success

addJob(&$job) {}

Insert a new job to the queue, the Job is passed by reference because
its new job ID and status will be set in the Job object

- @param Job $job The Job we want to insert to the queue (by ref.)
- @return int The inserted job id

getJob($job_id) {}

Returns a Job object describing a job in the queue

- @param int $job_id The job id
- @return Job Object describing a job in the queue

updateJob(&$job) {}

Updates an existing job in the queue with it's new properties. If job doesn't exists,a new job will be added. Job is passed by reference and it's updated from the queue.

- @param Job $job The Job object, the ID of the given job is the id of the job we try to update.
- If the given Job doesn't have an assigned ID, a new job will be added
- @return int The id of the updated job

suspendJob($job_id) {}

Removes a job from the queue

- @param int|array $job_id The job id or array of job ids we want to remove from the queue
- function removeJob($job_id) {}
- Suspend a job in the queue (without removing it)
- @param int|array $job_id The job id or array of job ids we want to suspend
- @return bool Success/Failure

resumeJob($job_id) {}

Resume a suspended job in the queue

- @param int|array $job_id The job id or array of job ids we want to resume
- @return bool Success/Failure (if the job wasn't suspended, the function will return false)

requeueJob($job) {}

Requeue failed job back to the queue.

- @param job $job job object to re-query
- @return bool - true or false.

getStatistics() {}

returns job statistics

- @return array with the following:
  - "total_complete_jobs"
  - "total_incomplete_jobs"
  - "average_time_in_queue" [msec]
  - "average_waiting_time"   [sec]
  - "added_jobs_in_window"
  - "activated_jobs_in_window"
  - "completed_jobs_in_window"
- moving window size can be set through the ini file

isScriptExists($path) {}

Returns whether a script exists in the document root

- @param string $path relative script path
- @return bool - TRUE if script exists in the document root FALSE otherwise

isSuspend() {}

Returns whether the queue is suspended

- @return bool - TRUE if job is suspended FALSE otherwise

getJobsInQueue($filter_options=null, $max_jobs=-1, $with_globals_and_output=false) {}

Returns a list of jobs in the queue according to the options given in the filter_options parameter, doesn't return jobs in "final states" (failed, complete). If the application id is set for this queue, only jobs with this application id will be returned.

- @param array $filter_options Array of optional filter options to filter the jobs we want to get from the queue. If not set, all jobs will be returned. Options can be: priority, application_id, name, status, recurring.
- @param int max_jobs Maximum jobs to retrive. Default is -1, getting all jobs available.
- @param bool with_globals_and_output. Whether gets the global variables data and job output.
- Default is false.
- @return array. Jobs that satisfies filter_options.

getNumOfJobsInQueue($filter_options=null) { }

Returns a list of jobs in the queue according to the options given in the filter_options parameter
If application id is set for this queue, only jobs with this application id will be returned

- @param array $filter_options Array of optional filter options to filter the jobs we want to get from the queue. If not set, all jobs will be returned.Options can be: priority, application_id, host, name, status, recurring.
- @return int. Number of jobs that satisfies filter_options.

getAllhosts() { }

Return all the hosts that jobs were submitted from.

- @return array.

getAllApplicationIDs() { }

Return all the application ids exists in queue.

- @return array.

getHistoricJobs($status, $start_time, $end_time, $index, $count, &$total) { }

Return finished jobs (either failed or successes) between time range allowing paging.
Jobs are sorted by job id descending.

- @param int $status. Filter to jobs by status, 1-success, 0-failed either logical or execution.
- @param UNIX timestamp $start_time. Get only jobs finished after $start_time.
- @param UNIX timestamp $end_time. Get only jobs finished before $end_time.
- @param int $index. Get jobs starting from the $index-th place.
- @param int $count. Get only $count jobs.
- @param int $total. Pass by reference. Return the total number of jobs satisfied the query criteria.
- @return array of jobs.

suspendQueue() { }

Suspends queue operation

- @return bool - TRUE if successful FALSE otherwise

resumeQueue() { }

Resumes queue operation

- @return bool - TRUE if successful FALSE otherwise.

getLastError() { }

Returns a description of the last error that occurred in the queue object. After every method invoked an error string describing the error is stored in the queue object.

- @return string.

setMaxHistoryTime() { }

Sets a new maximum time for keeping historic jobs.

- @return bool - TRUE if successful FALSE otherwise

Job Class

This class describes a job in a queue
In order to add/modify a job in the queue, a Job class must be created, retrieved and than saved in a queue or , a job can be added directly to a queue without creating an instant of a Queue object.

```
class ZendAPI_Job {
```

var $_id;

- Description: Unique id of the Job in the job queue.
- @var int

var $_script;

- Description: Full path of the script that this job calls when it's processed.
- @var string

var $_host;

- Description: The host from where the job was submitted.
- @var string

var $_name;

- Description: A short string describing the job.
- @var string

var $_output;

- Description: The job output after executing.
- @var string

var $_status = JOB_QUEUE_STATUS_WAITING;

- Description: The status of the job, by default, the job status is waiting to being executed. The status is determined by the queue and can not be modified by the user.
- @var int

var $_application_id = null;

- Description: The application id of the job. If the application id is not set, this job may get an application id automatically from the queue (if the queue was assigned one). By default it is null (which indicates no application id is assigned).
- @var string

var $_priority = JOB_QUEUE_PRIORITY_NORMAL;

- Description: The priority of the job, options are the priority constants. By default the priority is set to normal (JOB_QUEUE_PRIORITY_NORMAL).
- @var int

var $_user_variables = array();

- Description: An array holding all the variables that the user wants the job's script to have when it's called.
- The structure is variable_name => variable_value i.e. if the user_variables array is array('my_var' => 8), when the script is called, a global variable called $my_var will have the int value of 8. By default, there are no variables that we want to add to the job's script.
- @var array

var $_global_variables = 0;

- Description: A bit mask holding the global variables that the user want the job's script to have when it's called.
- Options are prefixed with "JOB_QUEUE_SAVE_" and may be: POST|GET|COOKIE|SESSION|RAW_POST|SERVER|FILES|ENV. By default there are no global variables we want to add to the job's script i.e. In order to save the current GET and COOKIE global variables, this property should be JOB_QUEUE_SAVE_GET|JOB_QUEUE_SAVE_COOKIE (or the integer 6). In that case (of GET and COOKIE), when the job is added, the current $_GET and $_COOKIE variables should be saved, and when the job's script is called,those global variables should be populated.
- @var int

var $_predecessor = null;

- Description: The job may have a dependency (another job that must be performed before this job). This property holds the id of the job that must be performed. If this variable is an array of integers, it means there are several jobs that must be performed before this job. By default there are no dependencies.
- @var mixed (int|array)

var $_scheduled_time = 0;

- Description: The time that this job should be performed, this variables is the UNIX timestamp. If set to 0, it means that the job should be performed now (or at least as soon as possible). By default there is no scheduled time, which means we want to perform the job as soon as possible.
- @var int

var $_interval = 0;

- Description: The job running frequency in seconds. The job should run every _internal seconds. This property only applies to recurrent jobs. By default, its value is 0 e.g. run it only once.
- @var int

var $_end_time = null;

- Description: A UNIX timestamp of the last time this job should occur. If _interval was set, and _end_time was not, then this job will run forever. By default there is no end_time, so recurrent jobsl run forever. If the job is not recurrent (Occurs only once) then the job will run at most once. If the end_time has reached and the job was not yet executed, it will not run.
- @var int

var $_preserved = 0;

- Description: A bit that determines if the job can be deleted from history. When set, removeJob will not delete the job from history.
- @var int

function ZendAPI_Job($script) {}

- Description: Instantiates a Job object, describing all the information and properties of a job.
- @param script $script relative path (relative to document root supplied in the ini file) of the script this job should call when it's executing.
- @return Job

function addJobToQueue($jobqueue_url, $password) {}

- Description: Adds the job to the specified queue (without instantiating a JobQueue object). This function should be used only when adding a single job, to insert more than one job and/or manipulate other jobs (or job tasks) create and use the JobQueue object. This function creates a new JobQueue and logs in to it (with the given parameters), adds this job and logs out
- @param string $jobqueue_url Full address of the queue we want to connect to.
- @param string $password For authentication, the queue password.
- @return int The added job id or false on failure.

function setJobPriority($priority) {}
Description: Set a new priority to the job.
@param int $priority, priority options are constants with the "JOB_QUEUE_PRIORITY_" prefix

All properties SET functions

function setJobName($name) {}
function setScript($script) {}
function setApplicationID($app_id) {}
function setUserVariables($vars) {}
function setGlobalVariables($vars) {}
function setJobDependency($job_id) {}
function setScheduledTime($timestamp) {}
function setRecurrenceData($interval, $end_time=null) {}
function setPreserved($preserved)

function getProperties() {}

- Description: Get the job properties.
- @return array The same format of job options array as in the Job constructor.

function getOutput() {}

- Description: Get the job output.
- @return An HTML representing the job output.

All properties GET functions

function getID() {}

function getHost() {}

function getScript() {}

function getJobPriority() {}

function getJobName() {}

function getApplicationID() {}

function getUserVariables() {}

function getGlobalVariables() {}

function getJobDependency() {}

function getScheduledTime() {}

function getInterval() {}

function getEndTime() {}

function getPreserved() {}

function getJobStatus() {}

- Description: Get the job's current status.
- If this job was created and not returned from a queue (using the JobQueue::GetJob() function), the function will return false. The status is one of the constants with the "JOB_QUEUE_STATUS_" prefix.E.g. job was performed and failed, job is waiting etc.
- @return int

function getTimeToNextRepeat() {}

- Description: Get how many seconds until the next time the job will run.
- If the job is not recurrence or it past its end time, then return false@return int

function getLastPerformedStatus() {}

- Description: For recurring jobs get the status of the last execution. For simple jobs, getLastPerformedStatus is equivalent to getJobStatus. Jobs that haven't been executed will return STATUS_WAITING.
- @return int

BIRT Reports

APIs and Directives

Zend has created a PHP API in PHP that uses the Java Bridge to communicate with BIRT classes and generate reports.
In order to begin benefiting from the advanced reporting capabilities, the BIRT APIs have to be incorporated into the PHP application's code.
The Use Cases in Platform Administration provide a demonstration of how the APIs work and how to insert them in the code. These Use Cases are viewed from Integration | BIRT Reports. In the same page is a download option "Download BIRT API and Samples" this option is used to download the BIRT APIs.

To incorporate Zend BIRT APIs in PHP Applications:

1. Download the BIRT API by clicking: “Download BIRT API and Samples”
2. Extract the files into your PHP application’s directory
3. Include these files to the project.

Once the files are incorporated into your PHP application’s project the different APIs can be inserted into your code to generate various reports.

The following is a detailed description of the BIRT API:

The Zend_Birt Class is a base class for Zend_Birt_Report_Document and Zend_Birt_Report_Design.

- Zend_Birt_Report_Design is used for creating reports from a design file
- Zend_Birt_Report_Report is used for creating reports from report document.

The Zend_Birt_Report_Design class, sets a design file in its constructor that can run to create a report document from a design file. The report's output can be rendered as a document in PDF or HTML format. Or the report document can be saved in a file and used later by the Zend_Birt_Report_Report object. The class also has runAndRenderReportToStream/runAndRenderReportToFile functions that will run and render a report in one step. The 'running report" functions get a parameter array that contains key=>value parameter.
The createReport function knows to run a design report and return the Zend_Birt_Report_Report object that is set with the created report document.

The Zend_Birt_Report_Document class sets a report document in its constructor and renders a report from the report document in HTML or PDF format.
This class inherits from Zend_Birt and handles the BIRT Report object created from a report document. It also gets an array of bookmarks, TOCs, report
information, and an html page of bookmarks and the html page count.

There are three basic actions that should be done in order to create BIRT Reports:

1. Instantiate the report design that contains the information and display type.
2. Define the parameters .
3. Render. the report.

Zend Platform Directives

Accelerator Directives

"zend_accelerator.max_wasted_percentage"

Description: Max percentage of "wasted" memory until restart is scheduled

"zend_accelerator.max_warmup_hits"

Description: How many hits are considered 'warmup' (for statistics)

"zend_accelerator.consistency_checks"

Description: Check cache's checksum each N requests

"zend_accelerator.force_restart_timeout"

Description: Time to wait for cache being unused when restart is scheduled (seconds)

"zend_accelerator.perform_timings"

Description: Collect performance statistics

"zend_accelerator.validate_timestamps"

Description: Check file timestamps

"zend_accelerator.max_cached_filesize"

Description: Max cached size for content cache (Kbytes)

"zend_accelerator.revalidate_freq"

Description: How often to check file timestamps on Windows (seconds)

"zend_accelerator.min_free_disk"

Description: Min disk space to leave free for content cache (in M or %)

"zend_accelerator.php_extensions"

Description: List of extensions to consider for content cache when directory is configured

"zend_accelerator.user_blacklist_filename"

Description: Path for a file that contains a list of files not to accelerate

"zend_accelerator.compress_blacklist_filename"

Description:Path for a file that contains a list of files not to compress

"zend_accelerator.compression"

Description: Enable compression for content cached files

"zend_accelerator.compress_all"

Description: Enable compression for accelerated files

"zend_accelerator.enabled"

Description: Enable acceleration

"zend_accelerator.output_cache_enabled"

Description: Enable content caching

"zend_accelerator.max_accelerated_files"

Description: Maximum number of keys (scripts) in accelerator hash table

"zend_accelerator.mmap_base_file"

Description: Windows: location of map address file

"zend_accelerator.httpd_uid"

Description: UID of the httpd process

"zend_accelerator.memory_consumption"

Description: Accelerator shared memory block size (Mbytes)

"zend_accelerator.allow_noshm"

Description: Allow running in "no shared memory mode" (CGI, CLI)

"zend_accelerator.output_cache_config"

Description: Content cache configuration file

"zend_accelerator.output_cache_dir"

Description: Content cache storage directory

"zend_accelerator.use_cwd"

Description: Use current directory as a part of script key

"zend_accelerator.preferred_memory_model"

Description: Shared memory model to use

"zend_accelerator.dups_fix"

Description: Use hack to prevent "duplicate definition" errors

"zend_accelerator.cgi_base_shm_address"

Description: The base address for the CGI/CLI shared memory block

Monitor Directives

"zend_monitor.max_var_len"

Description: Maximum variable length for collected data in POST/SERVERS. Limit applies to each single value.

"zend_monitor.warmup_requests"

Description: Number of requests until monitor would use averaging statistics to produce events

"zend_monitor.load_sample_freq"

Description: Frequency of checking for load events (seconds)

"zend_monitor.rotate_freq"

Description: Frequency for rotating monitor internal log files (seconds)

"zend_monitor.reconnect_timeout"

Description: How long monitor will wait until trying to restore broken connection to central (seconds)

"zend_monitor.watch_functions"

Description: List of functions to watch for time events (@file reads list from file)

"zend_monitor.watch_results"

Description: List of functions to watch for failure return events (@file reads list from file)

"zend_monitor.collector_host"

Description: Hostname for central

"zend_monitor.collector_port"

Description: Port for central

"zend_monitor.log_dir"

Description: Directory where monitor logs will be kept

"zend_monitor.server_key"

Description: Filename for local SSL key

"zend_monitor.server_cert"

Description: Filename for local SSL certificate

"zend_monitor.collector_cert"

Description: Filename for central SSL certificate

"zend_monitor.enable"

Description: Monitoring is enabled

"zend_monitor.error_level"

Description: Errors reported as events

"zend_monitor.error_level.severe"

Description: Errors reported as severe events

"zend_monitor.silence_level"

Description: If 1, does not report errors when error reporting is 0. If 2, doesn't report errors only if @ is used.

"zend_monitor.max_script_runtime_load_cutoff"

Description: Load value which would suppress time-related events

"zend_monitor.report_variables_data"

Description: Which variables to report (*)

"zend_monitor.max_script_runtime"

Description: Script runtime above which event is produced (ms)

"zend_monitor.max_function_runtime"

Description: Function runtime above which event is produced (ms)

"zend_monitor.max_memory_usage"

Description: Memory usage above which event is produced (K)

"zend_monitor.max_load"

Description: Load above which event is produced

"zend_monitor.max_script_runtime.severe"

Description: Script runtime above which severe event is produced (ms)

"zend_monitor.max_function_runtime.severe"

Description: Function runtime above which severe event is produced (ms)

"zend_monitor.max_memory_usage.severe"

Description: Memory usage above which severe event is produced (K)

"zend_monitor.max_load.severe"

Description: Load above which severe event is produced

"zend_monitor.max_time_dev"

Description: Deviation from average script runtime above which event is produced (%)

"zend_monitor.max_output_dev"

Description: Deviation from average output size above which event is produced (%)

"zend_monitor.max_mem_dev"

Description: Deviation from average memory usage above which event is produced (%)

"zend_monitor.max_time_dev.severe"

Description: Deviation from average script time above which severe event is produced (%)

"zend_monitor.max_output_dev.severe"

Description: Deviation from average output size above which severe event is produced (%)

"zend_monitor.max_mem_dev.severe"

Description: Deviation from average memory usage above which severe event is produced (%)

"zend_monitor.mem_threshold"

Description: If memory usage below this value, no deviation events are produced

"zend_monitor.time_threshold"

Description: If script runtime below this value, no deviation events are produced

"zend_monitor.output_threshold"

Description: If output size below this value, no deviation events are produced

"zend_monitor.event_overload_threshold"

Description: If more then 1000 events happen in this time (seconds), extra events will be dropped

"zend_monitor.disable_script_runtime_after_function_runtime"

Description: Disable "script slow" event after "function slow" event happened

"zend_monitor.tmp_dir"

Description: Directory where monitor temp files are written

(*) G - GET, P - POST, C - COOKIE, R - RAW_POST_DATA, E - ENV, V - SERVER, S - SESSION, F - FILES

Zend Monitor Event Types

For each event type there is a zend_monitor.<event_type>.
zend_monitor.<event_type> - can be set to off then this event type will not be reported. E.g.: zend_monitor.memsize.enable = Off however the same settings can be easily defined from PHP Intelligence | Event Triggers.

The following list displays the event types and the respective directive for enabling and disabling Events:

- Slow Script Execution Absolute - zend_monitor.longscript.enable
- Slow Script Execution Relative - zend_monitor.devscript.enable
- PHP Error - zend_monitor.zenderror.enable
- Function/Database Error - zend_monitor.funcerror.enable
- Slow Function Execution/Slow Query Execution - zend_monitor.longfunction.enable
- Excess Memory Usage (Absolute and Relative) - zend_monitor.devmem.enablezend_monitor.memsize.enable
- Inconsistent Output Size - zend_monitor.outsize.enable
- Load Average - zend_monitor.load.enable
- Custom Event - zend_monitor.custom.enable

Note:
All event types are enabled, by default. zend_monitor.enable when turned off will disable all event reporting activity.

Platform Administration Directives

zps.install_dir
Description: The place the Zend directory was installed to (Platform/ZPS)
studio.install_dir
Description: The place the Zend directory was installed to (Studio Server)
zend_gui.language
Description: Language code Platform Administration uses for texts (i.e. en for English)
zend_gui.language_charset
Description: If set, all the Platform Administration files will send a Content-Type header with this charset also used to send specific charset in Email (should be used in the Japanese version)
zend_gui.ini_modifier
Description: The path where the ini_modifier util is zend_central.error_logging
Description: If enabled, Platform Administration will log errors into the file'zend_central_error_log' that is located in the <install-dir>/logs directory.
zend_central.gui_address
Description: The full address of Platform Administration, this address is used by the node to access Platform Administration (login process) The address is in http(s)://host:port/path format
zend_central.node_address Each node has this directive set with his address, the same address that he gave the central during installation (this is the way the central identify the server in the DB) The address is ONLY the hostname/IP address

zds.your_servers_max_clients

Description: Use for the ZDS tests in Platform Administration (Performance section in Platform), to "know" what is the value of the maxClients of the server

Collector Center Directives

zend_monitor.collector_cert

Description: Certificate file for the CC

zend_monitor.collector_key

Description: Private key file for the CC

zend_monitor.collector_port

Description: Port to listen

zend_monitor.server_key_dir

Description: Dir to store node keys

zend_monitor.events_db

Description: Event DB URI

zend_monitor.pull_freq

Description: How often to pull node data (seconds)

zend_monitor.ping_freq

Description: How often to check node availability (seconds)

zend_monitor.log_dir

Description: Dir to store logs

zend_monitor.gui_dir

Description: Dir where Platform Administration files reside

zend_monitor.event_lifetime

Description: How long until event is considered too old (seconds)

zend_monitor.cleanup_freq How often to clean up old events (event count)

Debugger Directives

"zend_debugger.allow_hosts"

Description: Hosts allowed to connect (hostmask list)

"zend_debugger.deny_hosts"

Description: Hosts denied to connect (hostmask list)

"zend_debugger.allow_tunnel"

Description: Hosts allowed to use tunnel process (hostmask list)

"zend_debugger.expose_remotely"

Description: Which client can know debugger is installed

"zend_debugger.max_msg_size"

Description: Maximum message size accepted by Debugger

"zend_debugger.httpd_uid"

Description: UID for the httpd process

ZDS Directives

"zds.enable" Enable

Description: ZDS file serving

"zds.mime_types_file"

Description: Location of the MIME types file

"zds.log_file"

Description: Log file

"zds.min_file_size"

Description: Minimal file size to serve via ZDS process (smaller files served via Apache)

"zds.disable_byterange"

Description: Disable handling byte-range requests (all requests would return entire file)

"zds.mmap_chunk" Memory chunk to map when serving file, in K. Bigger chunks imply higher memory usage by ZDS.

"zds.nice"

Description: Priority of ZDS server process. Higher number means lower priority.

"zds.child_max"

Description: Maximum number of ZDS sub-processes

"zds.poll_delay"

Description: Delay between poll invocations, in order to enable other processes to run better

"zds.uid" UID of the ZDS file server

Zend Platform Built-In Services and Extensions

About

Services are the backbone of Zend Platform's production level features. To allow a wider control over the production level features, in this chapter we will describe the different services used by Zend Platform and how they can be enabled and disabled.

Note:

Before disabling Services and removing extensions from your php.ini make sure you are aware of the following dependences:

In Unix, Linux, i5/OS **and Mac**

A message will be added to the log indicating that an extension with dependences was disabled and the implications of this action.

In Windows“Zend Platform Job Queue” depends on “Zend Platform MySQL”

“Zend Platform Pinger” depends on “Zend Platform MySQL”

“Zend Platform Collector Center” depends on “Zend Platform MySQL”

“Zend Platform Action” depends on “Zend Platform Collector Center”

Stopping a service will result in a message that all its dependants’ will be also stopped, for example stopping ZPMySQL will display that ZPJobQ, ZPPinger, ZPCollector and ZPAction will be stopped.

Warning: Stopping ZPMySQl will disable the Administration User Interface for all operating systems.

Services are part of the installed package and in some situations are already running out of the box. This is determined according to the chosen installation method. The Zend Platform Installer has two types of installation methods, Custom and Express. The Custom installation prompts users to decide which services to run and the Express method only runs three essential components (PHP Extensions: Optimizer, Download Server, Accelerator, Monitor, Zend Cache and Debugger).

Naturally, services only run according to selected license type. Therefore, Development and Enterprise Licenses will include the Job Queues and Java Bridge services. These services will be applicable either when selected in the Custom installation or available from the Setup Tool, if not selected in the Custom installation or after installing using the Express method.

Note:

If you are using Zend Platform as a debugger (Remote Debugger) for use with Zend Studio the recommended installation method is express. This will provide only the essential services for using the debugger.

Setup Tool

The Setup Tool provides users with an easy way to activate services and change settings. The Setup Tool also is a means to configure options that were not setup in the installation process.

Running the Setup Tool

The setup Tool is a separate component and resides outside the Zend platform user interface.

To run the Setup Tool:

In **UNIX, Linux and Mac** run /usr/local/Zend/Platform/bin/setup_tool.sh from the shell.

In **i5/OS** run: GO ZENDPLAT/ZPMENU in the i5/OS command line

In **Windows** go to the Start menu and select Programs | Zend Platform | Setup Tool (in Windows a welcome screen will be displayed, choose the option "Modify" to access the configuration options).

The Setup Tool includes the following options:

2. Setup Java Bridge - The Java Bridge access Java based applications running in a Java
4. Setup Job-Queues - The Job Queues server services Job Queues
5. Register a Node – Register a server to belong to the Central server and be part of a cluster.
6. Change Platform Administration Password

For more information about the Setup Tool, refer to the Installation Guide.

Services

The following section provides a complete description of each service along with instructions on how to run and stop the services.

Java Bridge

Description	Integrates Java libraries and classes within PHP applications.
To Enable	Run the setup tool and select option number, 2 "Setup Java Bridge" and input the required data.

To Disable	▪ In Unix, Linux and Mac: Execute: /usr/local/Zend/Platform/bin/javamw.rc stop.
	▪ In i5/OS Execute GO ZENDPLAT/ZPMENU option 2 and then option 3.
	▪ In Windows: Go to Services (Start | Settings | Control Panel | Administrative Tools | Services) and stop the service (double-click on the service name and in the "Startup Type" field select the option "Disabled') "Zend Platform Java Bridge", you can also run one of these commands in the command line (Start | Run, enter CMD in the text area and press enter to open the command line):
	`net stop ZPJava`
	`or`
	`net stop "Zend Platform Java Bridge"`
	*Quotes are mandatory.
	
	To completely disable the Java Bridge, remove the directive zend_extension_manager.java_bridge from the php.ini to prevent loading a redundant shared object (in UNIX, linux i5/OC and Mac .so in Windows .dll).

Resulting Outcome	If stopped, integration with Java libraries and classes within PHP applications will be not be available.

Job Queues

Description	Job Queues reroutes and delays the execution of processes and to improve response times during interactive Web sessions.
To Enable	Run the setup tool and select option number, 3 “Setup Job Queues”

To Disable	▪ In Unix, Linux and Mac: Execute: /usr/local/Zend/Platform/bin/jqd.sh stop and remove the directive zend_extension_manager.jobqueue_client from the php.ini to prevent loading a redundant SO.
	▪ In Windows: Go to Services (Start | Settings | Control Panel | Administrative Tools | Services) and stop the service (double-click on the service name and in the "Startup Type" field select the option "Disabled') “Zend Platform Job Queues”, you can also run one of these commands in the command line (Start | Run, enter CMD in the text area and press enter to open the command line):
	`net stop ZPJobQ`
	`or`
	`net stop “Zend Platform Job Queue”`
	*Quotes are mandatory.
	
	To completely disable Job Queues, remove the directive zend_extension_manager.jobqueue_client from the php.ini to prevent loading a redundant shared object (in Unix .so in Windows .dll).

Resulting Outcome	If stopped, the ability to reroute and delay execution of jobs will be disabled (jobs that are already running will not be stopped).

Note:

To ensure the Java Bridgeand Job Queue services do not start again next time the server is booted, erase the following:

1) Open /usr/local/Zend/Platform /etc/rc.d/, and remove these symbolic links:

- S10mysql.sh -> /usr/local/Zend/Platform/MySQL/bin/mysql.sh
- S20jqd.sh -> /usr/local/Zend/Platform/bin/jqd.sh
- S30scd.sh -> /usr/local/Zend/Platform/bin/scd.sh
- S40javamw.rc -> /usr/local/Zend/Platform/bin/javamw.rc

*Not all will appear depending on your configuration preferences.

2) Delete platform_init.sh.

In Windows, open the Service Manager (Start | Settings | Control Panel | Administrative Tools | Services and set them to Disabled.

When reactivating these services this information will be restored automatically.

Cache Cleaner

Description	Cleans old files from the cache directory and maintains cache size below the limits. It runs as a stand-alone background program (daemon), it may be signaled by the user manually, or from the crontab, or from httpd when the cleanup is needed.
To Enable	Automatically configured during installation

To Disable	▪ In Unix, Linux Mac: Run the command:
	`#crontab -u $apache_user -e as root and remove the following line:`
	`“*/10 * * * *  /usr/local/Zend/Platform/bin/cache_clean -l /usr/local/Zend/Platform/etc/zend.ini &>/dev/null"`
	▪ In i5/OS Execute GO ZENDPLAT/ZPMENU option 2 and then option 5.
	▪ In Windows: Go to Services (Start | Settings | Control Panel | Administrative Tools | Services) and stop the service (double-click on the service name and in the "Startup Type" field select the option "Disabled') “Zend Platform Cache Cleaner”, you can also run one of these commands in the command line (Start | Run, enter CMD in the text area and press enter to open the command line):
	`net stop ZPCache`
	`or`
	`net stop “Zend Platform Cache Cleaner”`
	*Quotes are mandatory.

Resulting Outcome	This stand-alone background program cleans the cache directory from old files and maintains cache size below the limits. Deactivating it should only be done in the event you choose not to use Platform's caching abilities.

Collector Center

Description	Collects PHP events for PHP Intelligence.
To Enable	Automatically configured during central installation. In Windows the node collector service (that is part of the four collector services) is also installed in the Node installation.

To Disable	▪ In Unix, Linux and Mac: Run the command:
	`#crontab -u $apache_user -e as root and remove the following line:`
	`*/2 * * * *  /usr/local/Zend/Platform/bin/collector_center /usr/local/Zend/Platform/etc -D`
	▪ In i5/OS Execute GO ZENDPLAT/ZPMENU option 2 and then option 2.
	▪ In Windows: Go to Services (Start | Settings | Control Panel | Administrative Tools | Services) and stop the services (double-click on the service name and in the "Startup Type" field select the option "Disabled') “Zend Platform Action”, “Zend Platform Pinger” “Zend Platform Collector Center” “Zend Platform Node Collector” you can also run one of these commands in the command line (Start | Run, enter CMD in the text area and press enter to open the command line):
	`net stop ZPAction`
	`net stop ZPPinger`
	`net stop ZPCollector`
	`net stop ZPNodeCollector`
	
	`or`
	`net stop “Zend Platform Action”`
	`net stop “Zend Platform Pinger”`
	`net stop “Zend Platform Collector Center”`
	`net stop “Zend Platform Node Collector”`
	*Quotes are mandatory.

Resulting Outcome	If deactivated, PHP events will not be captured. This should be done in the event you do not wish to utilize PHP Intelligence's monitoring abilities.

Extensions

Zend Platform extensions provide additional functionality. In order to disable an extension, open you php.ini with a text editor and remove the extension lines (if present) or in Windows go to Services (Start | Settings | Control Panel | Administrative Tools | Services) and stop the service (double-click on the service name and in the "Startup Type" field select the option "Disabled'):

zend_extension_manager.optimizer

Description	Controls the Zend Optimizer component.
To Enable	Automatically configured during installation.
To Disable	Remove the line from your php.ini
Resulting Outcome	PHP code will not be optimized and files encoded with Guard will not run. Users may notice a decrease in performance after disabling the Optimizer.

zend_extension_manager.download_server (not applicable in Windows)

Description	Controls the Zend download server component.
To Enable	Automatically configured during installation.
To Disable	Remove the line from your php.ini
Resulting Outcome	Files defined in the mime_types file will not be serviced by the download server and will start consuming additional bandwidth.

zend_extension_manager.platform

Description	platform.so or platform.dll (in Windows) includes the components; Accelerator, Monitor and Zend Cache.
To Enable	Automatically configured during installation.
To Disable	Remove the line from your php.ini
Resulting Outcome	Disables Monitoring (PHP Intelligence) and Acceleration and can be used to stop both components at once instead of disabling each component individually.

Zend Platform Action (Windows Only)

Description	Checks if there are actions associated with an Event when an Event is added to the DB and executes them.
To Enable	Automatically configured during installation.
To Disable	Remove the line from your php.ini
Resulting Outcome	Actions associated with events will not be executed.

Zend Platform Pinger (Windows Only)

Description	This process periodically makes a query to nodes to verify activity and status (OS, PHP version, etc.)
To Enable	Automatically configured during installation.
To Disable	Remove the line from your php.ini
Resulting Outcome	Nodes will not be checked for activity and status and should be manually checked.

Zend Platform Collector Center

Description	Collects and aggregates information from nodes in the cluster that is displayed in the Zend Platform PHP Intelligence module.
To Enable	Automatically configured during installation.
To Disable	Remove the line from your php.ini
Resulting Outcome	Event information will not be collected from Nodes, this is equivalent to disabling the cluster configuration and deactivating PHP intelligence

Zend Platform Node Collector

Description	A daemon process that receives Events from Zend platform Nodes.
To Enable	Automatically configured during installation.
To Disable	Remove the line from your php.ini
Resulting Outcome	Event information will not be collected from nodes, this is equivalent to disabling the cluster configuration and deactivating PHP intelligence.

Tutorials

IN THIS CHAPTER...

INTEGRATING EXISTING AND LEGACY APPLICATIONS
CALLING AN EJB ON WEBSPHERE FROM PHP
PARTIAL AND PREEMPTIVE PAGE CACHING

This section of the User Guide is dedicated to tutorials on different subjects.

Tutorial Feedback

Please send us your opinion and suggestions for new tutorials by e-mail to: documentation@zend.com.

Integrating Existing and Legacy Applications

This tutorial details the integration of Zend Platform's Event Details screens with other legacy applications.

Reproducing and resolving bugs, one of the most problematic challenges of development, is often time consuming, and in most cases, almost impossible when information is not collected at the time of the occurrence. PHP Intelligence is an event driven system that provides real-time analysis of PHP applications. By enabling you to obtain immediate insight into your PHP applications, PHP Intelligence provides a fast and efficient means to reproduce and resolve problems, while maintaining a complete audit trail of the occurrence's details.

PHP Intelligence proactively alerts you to problematic occurrences in your application. This means that if you are a Developer or System Administrator you will not need to monitor the Zend Platform console at all times—instead, the information comes to you! An event, containing the audit trail of an occurrence, can be made known to you through an Email Notification. If you require full event details available outside of the Zend Platform console, an Event Details screen can be published to a URL in XML format. Both Event Details screens contain aggregated information relevant to the occurrence of an event, or in other words "Full Problem Context": Event type, Event ID, Timestamp, Severity, number of occurrences, etc.

Full Problem Context provides valuable information for the entire PHP application lifecycle (development, production and deployment). Exposing the source of an occurrence along with the ability to drill-down and investigate details pertaining to an event's location, time and context, provides in-depth insight to the reasons why the event occurred and a basis for resolving the issue.

PHP Intelligence includes the following Event Details screen Types: Slow Script Execution, PHP Errors, Function Errors, Memory Usage, Database Errors, Query Execution, Output Sizes, Load Averages and more...

Each Event Details screen Type includes basic and event-specific details such as: event type, event ID, Timestamp, Severity, number of occurrences, error type, error text, triggered value, load average, Source File Line, Script Name, Host URI, Vardata Type & Name, Function Name, Argument Numeric Value, Function, Included Files, Backtrace, etc.

Contents of the XML output can be easily utilized and integrated to provide an information feed to various legacy systems such as: Bug tracking systems for development and QA, CRM applications for managing customer care, management systems such as Tivoli and HP OpenView for system health information, and most commonly, generic monitoring systems such as Nagion or BigBrother that only provide OS service information.

Using event information, developers and administrator teams have a single point of reference to streamline the maintenance workflow. You can further enhance your development lifecycle by debugging your PHP code referenced in Event Details screens directly through the built-in integration with Zend Studio. This feature includes debug capabilities that enable you to add watches, define conditional breakpoints, view the stack trace and step into the source code to immediately debug the problem.

Zend Platform's XML output enables information to be easily interchanged. Using “Event Details”, developers can be sure that information pertaining to code, database and performance issues can be easily reused in a multitude of applications. Examples of this use include sending SMS messages containing event details, or triggering a mailing system to send a promotional gift to a customer who encountered a performance problem. Done by, extracting customer ID information provided to you in the Event Details screen (cookies).

Event Details screens are delivered as XML, by defining the relevant action (“Submit Report to the Specified URL”) for an event. The report information is submitted as XML data to the specified URL. Submission is done using the POST method, and the data is supplied through a variable named 'event_data'. This variable is accessible in PHP through $_POST['event_data'].

XML reports are structured as follows:

Each attribute is included if it exists in the Event Details screen:

```
<?xml version="1.0" ?>
<event type event_id class  timestamp time severity>
```

If there is an error:

```
    <error type>error text</error>
    <stats triggered_value avg load_average/>
```

If there is a source file:

```
    <source file line/>
    <script name host uri>
        <vardata type name value/>
    </script>
```

If there is a function:

```
    <function name>
        <args>
            <arg num value/>
        </args>
    </function>
```

If there are included files:

```
    <included_files>
        <file name\>
    </included_files>
```

If there is a backtrace for this event:

```
    <backtrace>
        <call depth function file line/>
    </backtrace>
</event>
```

By viewing the XML tagged file as fielded text, the fielding makes it possible to break Event Details screens down to their component parts to any degree of granularity for storage in a database. Once in the database, the data can be utilized by another application.

The following example shows how Event Details data can be extracted from an XML file and inserted into a database for use in a different system (could be any system based on a database, such as: Bug Tracking, CRM, management or any other application).

```
<?PHP
$event_xml_data = (isset($_POST['event_data']))?$_POST['event_data']:null;
if (!$event_xml_data) {    // no Event Context arrvied
    die();
}
$xml = simplexml_load_string($event_xml_data);
```

Implement different behaviors according to Class

```
$event_type = (string) $xml['type'];
// if this event is a Custom Event, we may implement different behaviors according to
the Custom Event’s class
if ($event_type == 'custom') {
$custom_class = (string) $xml['class'];
switch ($custom_class) {
// different behaviors according to the class
}
}
// get the new event id
$event_id = (int) $xml['event_id'];
// insert a new event with its genreal info (type and timestamp) to the db
insert_new_event_into_db($event_id,$event_type ,(int)
$xml['timestamp']);
// parse the function parameters of the function where the event occured
$function_parameters = array();
foreach ($xml->function->args->arg as $arg) {
    $function_parameters[(int) $arg['num']] = (string) $arg['value'];
}
// insert the function data (function name and parameters, where the
event occured) to the db
update_event_function_data($event_id, (string) $xml->function['name'],
$function_parameters);
/**
 * insert a new event (with some genreal info) to the db
 *
 * @param int $id id of the new event in the ZendPlatform events database
 * @param string $type the event type
 * @param int $timestamp the unix timestamp when the event occured
 */
function insert_new_event_into_db($id,$type,$timestamp) {
}
/**
 * update a specific event function data in the database
 *
 * @param int $id the event id we want to update
 * @param string $function_name name of the function where the event occured
 * @param array $function_params array of the function parameters (num
=> value)
 */
function update_event_function_data($id,$function_name,$function_params) {
}
?>
```

The first part of the example uses a PHP 5 Simple XML extension to parse XML to PHP objects that can be processed with normal property selectors and array iterators. The second part extracted data from the event XML data, and inserted it into a database.

As this tutorial demonstrates, XML Event Details generated by the PHP Intelligence component of Zend Platform, provides Developers and System Administrations a single point of reference for production environments, and to streamline maintenance workflow. In environments where multiple management applications are an everyday reality, Zend Platform provides a flexible information feed to legacy systems, relieving the overhead normally required to integrate with these applications.

Calling an EJB on Websphere from PHP

This tutorial reviews the steps needed to call an EJB on WebSphere from PHP using the Java Bridge. These instructions assume that the developer has a system(s) with WebSphere, PHP, and the Zend Platform installed.

To call an EJB on Websphere from PHP, a script file needs to be created. This script should start the javaMW server with the correct settings to run a Websphere client.

Some of the important things to remember are:

1. Use IBM's java that ships with the Application Client to run the javaMW server.
2. The jars containing the client classes for any EJB that is to be called need to be classpath of the javaMW server.
3. The jars and environment variables for the WebSphere application client runtime need to be on the command line staring the javaMW server.

The following is an example script file for starting the javaMW server with the WebSphere runtime configuration options and the jars needed to call the Basic Calculator Technology sample shipped with Websphere.

Note:
This script is a modified version of the Basic Calculator Thin Client script file shipped with the Websphere Client Install.

```
#!/bin/sh
. /opt/IBM/WebSphere/AppClient/bin/setupClient.sh
# Change the PROVIDER_URL to point to this machine or another server.
if [ "${SERVERPORTNUMBER}" != "" ]
    then
       PROVIDER_URL=iiop://$DEFAULTSERVERNAME:$SERVERPORTNUMBER
else
         PROVIDER_URL=iiop://$DEFAULTSERVERNAME
fi
"$JAVA_HOME/bin/java" $WAS_LOGGING -classpath
/usr/local/Zend/Platform/Bin/javamw.jar:/opt/IBM/WebSphere/AppClient/samples/lib/Tec
hnologySamplesThinClient/BasicCalculatorClientCommon.jar:/opt/IBM/WebSphere/
AppClient/samples/lib/TechnologySamplesThinClient/BasicCalculatorThinClient.
jar:/opt/IBM/WebSphere/AppClient/samples/lib/TechnologySamplesThinClient/Bas
icCalculatorEJB.jar
-Djava.ext.dirs="$WAS_EXT_DIRS" -Djava.naming.provider.url=$PROVIDER_URL
```

```
-Djava.naming.factory.initial=com.ibm.websphere.naming.WsnInitialContextFac
tory
-Dzend.javamw.threads=20 -Dzend.javamw.port=10001 "$SERVER_ROOT"
"$CLIENTSAS" com.zend.javamw.JavaServer
```

Instructions

- Start the javaMW server using the script.
- Write a PHP client, which uses the Java Bridge functionality to call the ejb running on Websphere.

The following is an example PHP client for calling the Basic Calculator Technology sample.

```
<?php
   // Get the provider URL and Initial naming factory
   // These properties were set in the script that started the Java Bridge
   $system = new Java("java.lang.System");
   $providerUrl = $system->getProperty("java.naming.provider.url");
   $namingFactory = $system->getProperty("java.naming.factory.initial");
   $envt = array(
     "javax.naming.Context.PROVIDER_URL" => $providerUrl,
     "javax.naming.Context.INITIAL_CONTEXT_FACTORY" => $namingFactory,);
   // Get the Initial Context
   $ctx = new Java("javax.naming.InitialContext", $envt);
   // find the EJB
   $obj = $ctx->lookup("WSsamples/BasicCalculator");
   // Get the Home for the EJB
   $rmi = new Java("javax.rmi.PortableRemoteObject");
   $home = $rmi->narrow($obj, new
Java("com.ibm.websphere.samples.technologysamples.ejb.stateless.basiccalcula
torejb.BasicCalculatorHome"));
   // Create the Object
   $calc = $home->create();
   // Call the EJB
   $num = $calc->makeSum(1,3);
   print ("<p> 1 + 3 = $num </p>");
?>
```

This tutorial detailed how developers can use EJB on WebSphere from PHP using the Java Bridge. The two steps are: to create a script file to start the JavaMW server and to write a PHP client which uses the Java Bridge functionality to call the EJB running on WebSphere.

Partial and Preemptive Page Caching

This tutorial will review one of Zend Platform’s most powerful features, Partial Page Caching. Partial Page Caching is used in cases where it is impractical or impossible to cache the entire 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 type of usage is when some of the output is a form, that has credit card numbers, addresses and all kinds of information that for security reasons, should not be cached
The following tutorial is a step-by-step guide to mastering Partial Page Caching.

Inside this tutorial, you will find several ways to cache you output:

- Partial Page Caching APIs – a general overview of the Caching APIs with usage examples.
- Action Based Partial Page Caching – Cache using buttons and conditions

Partial Page Caching APIs

Partial Page Caching can also be achieved using the following functions for almost all situations:

Output Caching Functions

Function	Action
output_cache_fetch()	Gets the code's return value from the cache, if it is there.
output_cache_output()	Calls a function and checks if the function exists in the cache.Yes – PrintNo – Puts function output in cache and prints.
output_cache_exists()	Checks if the key exists in the cache.Yes – Print.No – Runs code, output in cache and prints until it reaches the stop command: output_cache_stop().
output_cache_stop()	Indicates the end of a block of code.

Data Caching Functions

Function	Action
output_cache_put()	Enters a single variable into the cache
output_cache_get()	Gets the Variable from the cache at the end of its lifetime.

Invalidate Cache

Function	Action
output_cache_remove_key (string key)	Respectively remove items from the cache according to their type (Key, URL or File)

The partial caching functions are divided into two groups, output caching and data caching. This document will explain each of the two groups and give practical examples of each function call.

Output Caching

The first groups of functions are the output caching functions. These functions capture the output from a function or block of PHP code and cache it. These functions are:

```
output_cache_output()
output_cache_exists()
```

```
output_cache_stop()
```

Output caching functions allow programmers to remove the execution of blocks of code with static output, such as looping over and printing the day's news headlines. This output changes infrequently, so instead of reprocessing it for every user caching allows PHP to skip execution and print the results.

Prototype:

```
void output_cache_output(string key, string code, int lifetime)
```

The first time *output_cache_output()* is called, it will execute the function defined in argument two, and store any output in the cache under the retrieval key specified in argument one. Each subsequent call to *output_cache_output ()* with the same value as argument one will result in the output of this cached data instead of the execution of the function in argument two, until the cache lifetime in argument three expires.

output_cache_output () is typically used to capture the output created by a function call. In order to use it, you would need to wrap a section of code as a function. When you call *output_cache_output ()*, it will call this function and cache it's output.

output_cache_output () takes three arguments:

1. The key value with which this output will be cached.
2. The function call.
3. The cache lifetime in seconds.

output_cache_output() has no return value.

Usage Example

```
<?
function content($time) {
/* Create a function to Wrap the code that produces
   the output */

    print "<p>Cached Time: $time </p>";
    /* The actual value of $time will be printed
       only once every 30 seconds. The output from
       the print statement will be cached, and the
       function call will be ignored until the cache
       lifetime expires */
}
$time = time();
/* Get current time, in seconds */
print "<p>Current Time: $time </p>";
/* Print the real current time */
output_cache_output("Current Time","content($time);",30);
/* Cache all output for the function content() and
   store it based on the key "Current Time" for 30
   seconds */
?>
```

Usage Notes:

output_cache_output() is used to cache the output generated from functions. To utilize it, wrap a section of code (which generates the output you wish to cache) as a function. Any output statements in this new function will be captured into a buffer and stored as cached data with the key specified. The other two functions, *output_cache_exists ()* and *output_cache_stop ()*, are used in tandem to simplify the task of caching output from a given section of code.

Prototype:

```
boolean output_cache_exists(string key, int lifetime)
void output_cache_stop()
```

output_cache_exists() is called from a conditional statement. The conditional statement should wrap the section of code producing the output you are intending to cache.

output_cache_exists() takes two arguments:

1. The key value with which this output will be cached.
2. The cache lifetime in seconds.

output_cache_exists() returns a boolean. TRUE is the key exists, FALSE if it doesn't.

output_cache_stop() takes no arguments.

output_cache_stop() has no return value.

Usage Example

```
<?
if (!output_cache_exists("Some_Key2", 3)) {
    echo time();
    output_cache_stop(); // Stop buffering...
}
?>
```

Data Caching

In cases where caching the output from a script isn't possible, we offer a set of functions for caching data. These functions are output_cache_fetch(), output_cache_put() and output_cache_get().

Data caching allows programmers to skip execution of repetitive database calls, increasing script performance and reducing overhead on the database. Typical uses of data caching include caching user preferences, caching product pricing, or any SQL call which changes infrequently.

Prototype:

```
string output_cache_fetch(string key, string code, int lifetime)
```

output_cache_fetch() works in a similar manner to the output caching function, *output_cache_output()*. The major difference is that instead of caching the output from the function it caches the return value as a string.

***output_cache_fetch()* receives 3 arguments:**

1. Unique identifier string for the data (string)
2. PHP code to be cached (string)
3. Cache lifetime in seconds (integer)

output_cache_fetch() returns a string containing the return value of the cached code section as defined in argument 2. The ID, defined in argument 1, serves to differentiate the code section and give it a name. Lifetime, defined in argument 3, is handled in the same way the Zend Performance module handles cache lifetimes for all cached files -- cached copies older than the lifetime will be refreshed when the function is called.

Note:
Unlike normal caching only the return value of the given PHP code is cached.

Usage Example

```
<?
function get_content($time, $sec, $usec) {
    /* I define an arbitrary function which
       returns data. */

    $data = array();
    $data["time"] = $time;
    $data["sec"] = $sec;
    $data["usec"] = $usec;
    /* Create an array to store the data.
       This is where you would generate the data
       you wish to cache, such as making database
       calls. */
    $ser_data =  serialize($data);
    /* serialize the array for return */

    return ($ser_data);
}
$time = time(); /* get current timestamp */
$micro = microtime();
/* get current timestamp
   including microseconds */
list ($usec, $sec) = explode(" ", $micro);
/* Print the real current time */
print "<p>Current Time: $time, $sec, $usec</p>";
$cached_string = output_cache_fetch("Example: Fetch","get_content($time, $sec,
$usec);",30);
/* Call the function via the 'output_cache_fetch()'.
   If the content key exists and the lifetime hasn't
   expired, the function execution will be skipped
   and the cached data will be returned via the cache
   API call. */
$data = unserialize ($cached_string);
/* unserialize the data */
/* Print the cached time */
print "<p>Cached Time: " . $data["time"] . "," . $data["sec"]. "," .$data["usec"] .
"</p>";
print "<p><b>Refresh to see caching in action!</b></p>";
?>
```

The strength of *output_cache_fetch* is that it allows the developer to offload repetitious database calls by wrapping these calls in a function. The following example illustrates how this would work.

```
<?php
/* Note that this code is kept as simple as possible,
   with no return type checks etc., in order to focus
   on the caching features. */
/* Display greetings and read user info from database -
   this part must remain dynamic */

$user_id=($_SESSION['user_id']);
$result = mysql_query("SELECT name,country,airport
                       FROM users
                       WHERE id=$user_id");
list($name,$country_id)=mysql_fetch_array($result,MYSQL_ASSOC);
echo "<P>Welcome $name ".date("F j, Y, g:i a") ."</P> ";
/* Note that the code to be cached should be wrapped as
   a single function call (see argument 2 in above
   example) - this is to improve readability and code
   reuse (the code we want to cache is usually longer
   than one line.) */
// Display list of destination countries
$sql = "SELECT id,name FROM countries";
echo "<P>Destinations: ";
$destination_str=output_cache_fetch("destinations","GetQuery('$sql')",3600);
$destination_arr = unserialize($destination_str);
$out = "<P>Destination Airport: <SELECT NAME=\"destination_airport\">";
foreach ($destination_arr as $destination) {
        $out .= "<OPTION VALUE=\""
        .$destination['id']."\">"
        .$destination['name']."</OPTION> ";
    }
$out .= "</SELECT></P> ";
echo $out;
echo "</P>";
/* In the first caching example (above), we cache the
   list of destinations. This is the same for every
   user, and so the cache id is a simple string. In
   the second example (below), the list of airports
   depends on the user's country. So, the country_id
   is added in the ID string. This will create a
   different cache copy for each continent. */
// Display list of airports in the user's home country
$sql = "SELECT id,name FROM airports WHERE country='$country_id'";
$airports_str = output_cache_fetch("airports_$country_id","GetQuery('$sql')",3600);
$airports_arr = unserialize($airports_str);
$out = "<P>Departing Airport: <SELECT NAME=\"depart_airport\"> ";
foreach ($airports_arr as $airport) {
    $out .= "<OPTION VALUE=\"".$airport['id']."\">".$airport['name']."</OPTION> ";
}
$out .= "</SELECT></P> ";
echo $out;
/* This function is more general purpose, meant to be
   used with output_cache_fetch () it performs an SQL
   query and returns the results as a serialized string.*/
function GetQuery ($sql_query) {
    $result = mysql_query($sql_query);
    $res_arr = mysql_fetch_array($result, MYSQL_ASSOC));
    $res_str = serialize($res_arr);
    return $res_str;
}
?>
```

Usage Notes:

You can use *output_cache_fetch()* to cache non-string types (e.g. arrays and objects) of PHP variables by using PHP's serialize() and un-serialize() to convert them to strings and vice versa.
output_cache_fetch requires the code which generates the cache data to be wrapped in a function. This allows the cache routine to skip the execution of this code if the data is already cached
output_cache_put() and *output_cache_get()* provide a direct way to store and retrieve data from the cache.

Prototype:

```
void output_cache_put(string key, mixed data)
void output_cache_get(string key, int lifetime)
```

***output_cache_put()* takes two arguments:**

1. The key value with which this data will be cached.
2. The data to be cached. (scalar, string, or serialized data).

output_cache_put() has no return value.

***output_cache_get()* takes two arguments:**

1. The key value with which the data is stored.
2. The lifetime that this data should be considered valid.

output_cache_put() returns the cached data, if it exists and is valid. Otherwise it returns false.

Usage Example

```
<?
if(($result = output_cache_get("TestFunctionResult", 30)) === false) {
    $result = microtime();  /* Current timestamp */
    print "<br><i>Fetching Fresh Content</i><br>";
     /* Should only print this every 30 seconds when
        the content is fetched fresh */
    output_cache_put("TestFunctionResult", "Cached: $result");
}
print "<b>$result</b><br>";
?>
```

Usage Notes:

The put/get routines are a simpler method for caching data than output_cache_fetch. They are typically used for the storage and retrieval of small bits of data.

Action Based Partial Page Caching

Action Based Partial Page Caching pertains to caching part of an output based on the occurrence of an action. This type of Caching is necessary in instances where it is preferable to refresh the Cache when an action occurs rather than time based.
For example: If we have a list of people who are “Currently Online” and we were to use time based caching, we would have to set an extremely short time limit to make sure that the list is updated at all times. We would also waste valuable system resources every time we refresh the cache. Instead, we can adopt a more efficient approach: refreshing the cache based on an action, for instance, every time a member goes online or logs out.
How do we do this?

There are two "Partial Page Cache" options based on an action:

1. Conditional Partial Page Caching
2. Button Based Partial Page Caching

Conditional Partial Page Caching

With this option, we predetermine conditions for caching and invalidating (If X occurs then do Y).
For example: we can cache our list of people who are "Currently Online" based on their log-on action. Whenever someone logs-on, his or her name will be added to the cache. Subsequently, when the same person logs-off we could set another condition will remove the name from the cache.
The following code example demonstrates how to empty the cache when a certain action occurs:

```
if (check_some_condition()) {
output_cache_remove_key (...);
}
```

Button Based Partial Page Caching

With this option, we set a specific button to initiate refreshing the Cache (Pressing button X does Y).
For example: we can cache our list of people who are "Currently Online" based on a specific button that the person logging-in will press (such as: login, next, go, etc). Whenever someone presses the button, his or her name will be added to the cache. Subsequently, when the same person presses a different button, his or her name will be removed from the cache.
The following example demonstrates the button triggered Partial Page Caching technique:

```
<form action="clean_cache.php">
<input type="submit" value="Clear Cache">
</form>
```

The 'action' attribute points to the clean_cache.php script. Therefore, when the user submits the form, clean_cache.php is executed+.
The clean_cache.php file is the same as the one that we put in the Cron Job example (see next chapter): it clears the cache (with output_cache_remove()) and then builds it again
(With fopen("http://..")) - So we get a real cache refresh.

This tutorial details Action Based Partial Page Caching, with conditional or button-oriented options.

Appendixes

Appendix A – Troubleshooting Zend Platform

Web Server

Possible Issue	Recommended Action
Images are not displayed in the Administration User Interface (GUI) when using Apache 2.	Add "EnableSendfile Off" to your httpd.conf. Note: this is not a Zend Platform issue if the Sendfile support is broken, images of other web pages on the same server will suffer from the same problem and this action will fix the problem for the entire server.

Accelerator

Possible Issue	Recommended Action
Received Error Message: Cannot communicate with reporting daemon. Reporting disabled. Please restart httpd to re-enable reporting.	The connection to the monitoring reporting process was broken. This happens when daemon process is either dead or can not timely respond to events sent to it (either stuck or overloaded). Immediate work-around would be to, restart Apache, that will launch a new copy of reporting process. Also examine logs for the evidence of reporting process logging any errors or crashing.

The Communication Tunnel

The Communication Tunnel includes settings in Zend Platform and Zend Studio. The following lists the possible causes and solutions depending on the origin of the problem.

Troubleshooting Zend Studio

If Zend Studio is unable to connect to the target server, you will get an error message with the response from the server. The table below describes the most likely causes and recommended actions for successfully establishing a connection with the target server.

Possible Issue	Recommended Action
The server address or the port you entered is incorrect	Enter the correct server information in the Tunneling Settings dialog.
HTTP authentication is required	Enter authentication information in the Tunneling Settings dialog box; then click the 'Send authentication information' checkbox.
The dummy file content or location on the server is incorrect	The dummy file on the server side was changed or does not exist. You will need to insure that the correct dummy file with the correct content is placed in the correct directory on the target server (The correct dummy file is created and located properly as part of the Installation procedure. The problem here is post-installation).
You are not allowed to connect with the server via the communication tunnel	You must have tunneling permissions in the Zend Platform Allowed Hosts Studio Server | Settings.

Troubleshooting Zend Platform

If Zend Platform is unable to communicate, there are a number of possible reasons. The table below describes the likely reasons and suggests possible solutions.

Possible Issue	Recommended Action
Zend Studio is not running.	Run Zend Studio.
The version of Zend Studio you are using is lower than 4.0.0.	Please install a newer version, if available. Zend Platform’s interface with Zend Studio requires Zend Studio 4.0 (or higher).
Port for auto detection not the same	Check that Zend Studio is listening to the same port as the one to which you are trying to connect.
Some other failure happened in the browser or in the Zend Studio	Use manual settings and if that doesn't work contact Zend Support

Appendix B – Configuration Check List

This Check List details all the Zend Platform configuration tasks in chronological order. This list can be printed and used as an extra aid for setting-up Zend Platform.

1. Configure Clusters and Groups when working in a cluster environment to enable event aggregation over multiple servers.
   Platform |Status | Manage Cluster or use the Shortcut Platform | Dashboard | Manage Cluster
2. Event Triggers, to modify default settings to suit the new environment:
   PHP Intelligence | Event Triggers
   Or use the shortcut Platform | Dashboard | Event Triggers
3. Configure Action Rules, to send Event Details data by e-mail or to a URL:
   Platform | Dashboard | Event Actions
4. Configure Performance, to define initial performance settings for Code Acceleration, Dynamic Content Caching, File Compression and Download optimization:
   Performance | Settings
5. Configure Virtual Hosts and fine tune performance setting per file:
   Performance | File View
6. Setup integration with Zend Studio Server:
   Go to Zend Core Server and configure the settings.
7. Establish a persistent connection with Zend Studio for Debugging Profiling and Editing code:
   Go to Zend Core | Server.
8. Configure PHP settings to customize the php.ini and zend.ini to your environment:
   Configuration | PHP Configuration or use the Shortcut: Platform | Dashboard | Configure PHP Settings
9. Define User and Group permissions
   Platform | User Management

Appendix C – Performance Lifecycle Check List

This Check List details all the Zend Platform performance Lifecycle tasks in chronological order. This list can be printed and used as an extra aid for calibrating Zend Platform.

1. Benchmark Web application, to establish optimization-starting point:
   Performance | Testing | Analyze Site – Run Performance Tool
2. Calibrate Event rules to configure PHP Intelligence events to the Web application's performance parameters:
   PHP Intelligence | Event Triggers
   Or use the shortcut Platform | Dashboard | Event Actions
3. Benchmark Web application, to establish a second optimization-starting point:
   Performance | Testing | Analyze Site – Run Performance Tool
4. Analyze Event Details, to pinpoint performance issues:
   PHP Intelligence | Event List
   Recommended: Focus on the following performance related event types:
   - Slow Script Execution (Absolute and Relative)
   - Slow Query Execution
   - Slow Function Execution

    - Excess Memory Usage (Absolute and Relative)
5. Apply Caching to boost Web application performance:
    - Define Dynamic Content Caching: Performance | File View (or from the Site Analysis results).
    - Apply Partial Page Content Caching APIs (see Tutorial)
6. Configure Acceleration to save code compilation time:
    - Acceleration Settings: Performance | Settings
    - Acceleration Blacklist: Performance | File View
7. Configure Compression to consume less bandwidth:
    - Compression Settings: Performance | Settings
    - Compression Blacklist: Performance | File View

      Important: Deactivate compression entirely if the server is set to handle compression (Performance |Settings | File Compression).
8. Configure Optimization optimize script and detect encoded files:
   Platform | Dashboard | Configure PHP Settings | Zend | Zend Optimizer
9. Benchmark Web application, to view optimization boost:
   Performance | Testing | Analyze Site – Run Performance Tool

Appendix D - Event Aggregation Mechanism

Introduction

This appendix covers the event aggregation mechanism in the Central Server. It will try to answer the fundamental question: "When are two events considered to be of the same origin (or cause) and therefore reported as one?"

Event properties

To answer this question we first have to define the different properties (or attributes) that define an event. Here is a list of the attributes that are used for aggregation along with a short definition:

- Event type - the type of the error that triggered the event (PHP error, Function error etc'). Perhaps the most important property since it also determines which other properties will be compared.
- Source file, Line number - the name of the PHP file and the line that contains the code that triggered the event. This file may not be the file that the user requested. Not all events have code location - e.g., "slow script" events and other events related to the whole script do not.
- Function name - the name of the function that contains the code that triggered the event. If the event happened in the global scope it's reported in the 'main' function.
- Location - one of two: either the server id of the server that triggered the event or the group id if the server belonging to an aggregated group.
- Aggregation Hint - this is a string that is supplied by the user to differentiate between pages that have the same URL but different parameters. If the user did not supply a hint the default hint is an empty string (The limit for Aggregation hints is 255 chars, longer hints will not be aggregated).
- Error text - the error text that was attached to the event.

- Script id - refers to the record for the script that the user requested (i.e., derived from original request URL).
- Severity - the severity of the event - currently, has two levels - regular and severe.

Another property that is taken into account is the event status. Only events that are not closed are aggregated.

Events are not aggregated when they are one of the following:

- Events of different types.
- Events that happened on different non-aggregated servers.
- Events with different aggregation hints.
- Events with different severity.

Zend Error Events

The following properties must be equal for events that are of type "zenderror":

1. Type (note: this is a Zend error type, like E_WARNING, not monitor error type)
2. Source file
3. Line number
4. Function name
5. Location
6. Aggregation hint

The Error text attribute must be 75% similar. (To learn more about text similarity read http://uk.php.net/manual/en/function.similar-text.php)

Function Error Events

The following properties must be equal for events that are of type "funcerror" or "dberror":

1. Source file
2. Line number
3. Function name
4. Location
5. Aggregation hint

If one of the events has an Error text attribute than the Error texts must be the same (not similar!).

Long Function Events

The following properties must be equal for events that are of type "longfunction" or "longquery":

1. Script id
2. Source file
3. Line number
4. Function name
5. Location
6. Aggregation hint
7. Severity

Custom Events

The following properties must be equal for custom events:

1. Type (this is the first parameter user provides)
2. Severity
3. Event text
4. Source file
5. Line number

Additional Events

The rest of the events are aggregated according to following attributes if two conditions are met:

1. The event type is one of the following: "devmem", "memsize", "devscript", "outsize" or "longscript".
2. The event has a script id attribute

For these events the following attributes must be equal:

1. Type
2. Script id
3. Location
4. Severity

Appendix E – Zend Platform Support

Zend Platform Support

Zend Platform Support provides Zend Product owners and prospective Zend Product owners with information regarding: System Requirements, Installation Instructions, General FAQ, Quick Start Guide and much more.

Visit: http://www.zend.com/support/support_platform.php

Zend Support Center

The Zend Support Center is your online destination for information and assistance for Zend's best-of-breed PHP products and technologies:

- Knowledge Base
- Support FAQ
- Submit a Support Ticket

Visit: https://www.zend.com/support/index.php

Support Tool

The Zend Support Tool gathers server configurations and setup information. This is used to aid in the support process to troubleshoot support issues and provide comprehensive and efficient support.
The type of information collected is as follows (partial list):

The type of information collected it as follows (partial list):

- php.ini (content and location)
- httpd.conf (content and location)
- Results of phpinfo() on the server
- Output of 'df'
- Output of 'uname -a'
- All of the various logs our products generate (installation log, etc.)
- Output of 'ls -lR of /usr/local/Zend/Platform' (The install_dir)

Use the support tool wizard to create, gather and send information regarding your Server's configuration and setup.

The information collected by the Support Tool can be stored and distributed in several ways:

1. Submit a ticket to Zend.com support
2. Collect information and save it in an archive
3. Collect information and send it by e-mail

The support tool is accessed from Platform Administration by going to:
Platform | Dashboard | Configure & Management Tools | Support Tool
In case of problems during Installation or later on when using Platform Administration, the Support Tool can be run from:

```
Unix: <Installation_dir>/bin/support_tool.sh
Windows: <install_dir>\bin\support_tool.bat
```

Getting Support

The Zend Support Center allows users to benefit from other user's experiences by viewing Knowledge Base articles, participating or viewing posts made to one of the product User Forums. Easily access the Support Center from the online help by clicking the Support button.

Appendix F – zend.ini Configuration Settings

The following table lists the zend.ini directives. Directives marked with YES do not require restarting the server apply changes.

Accelerator Directives

Directive	Reload
"zend_accelerator.max_wasted_percentage"	YES
"zend_accelerator.max_warmup_hits"	YES
"zend_accelerator.consistency_checks"	YES
"zend_accelerator.force_restart_timeout"	YES
"zend_accelerator.perform_timings"	YES
"zend_accelerator.validate_timestamps"	YES
"zend_accelerator.max_cached_filesize"	YES
"zend_accelerator.revalidate_freq"	YES
"zend_accelerator.min_free_disk"	YES
"zend_accelerator.php_extensions"	NO*
"zend_accelerator.user_blacklist_filename"	NO*
"zend_accelerator.compress_blacklist_filename"	NO*
"zend_accelerator.compression"	NO
"zend_accelerator.compress_all"	NO
"zend_accelerator.enabled"	NO
"zend_accelerator.output_cache_enabled"	NO
"zend_accelerator.max_accelerated_files"	NO
"zend_accelerator.mmap_base_file"	NO
"zend_accelerator.httpd_uid"	NO
"zend_accelerator.memory_consumption"	NO
"zend_accelerator.allow_noshm"	NO
"zend_accelerator.output_cache_config"	NO
"zend_accelerator.output_cache_dir"	NO
"zend_accelerator.use_cwd"	NO
"zend_accelerator.preferred_memory_model"	NO
"zend_accelerator.dups_fix"	YES

Monitor Directives

Directive	Reload
"zend_monitor.max_var_len"	YES
"zend_monitor.warmup_requests"	YES
"zend_monitor.load_sample_freq"	YES
"zend_monitor.rotate_freq"	YES
"zend_monitor.reconnect_timeout"	YES
"zend_monitor.watch_functions"	NO*
"zend_monitor.watch_results"	NO*
"zend_monitor.collector_host"	NO

"zend_monitor.collector_port"	NO
"zend_monitor.log_dir"	NO
"zend_monitor.server_key"	NO
"zend_monitor.server_cert"	NO
"zend_monitor.collector_cert"	NO
"zend_monitor.enable"	YES
"zend_monitor.error_level"	YES
"zend_monitor.error_level.severe"	YES
"zend_monitor.silence_level"	YES
"zend_monitor.max_script_runtime_load_cutoff"	YES
"zend_monitor.report_variables_data"	YES
"zend_monitor.max_script_runtime"	YES
"zend_monitor.max_function_runtime"	YES
"zend_monitor.max_memory_usage"	YES
"zend_monitor.max_load"	YES
"zend_monitor.max_script_runtime.severe"	YES
"zend_monitor.max_function_runtime.severe"	YES
"zend_monitor.max_memory_usage.severe"	YES
"zend_monitor.max_load.severe"	YES
"zend_monitor.max_time_dev"	YES
"zend_monitor.max_output_dev"	YES
"zend_monitor.max_mem_dev"	YES
"zend_monitor.max_time_dev.severe"	YES
"zend_monitor.max_output_dev.severe"	YES
"zend_monitor.max_mem_dev.severe"	YES
"zend_monitor.mem_threshold"	YES
"zend_monitor.time_threshold"	YES
"zend_monitor.output_threshold"	YES
"zend_monitor.event_overload_threshold"	YES
"zend_monitor.disable_script_runtime_after_function_runtime"	YES
"zend_monitor.tmp_dir"	NO
"zend_monitor.longscript.enable"	YES
"zend_monitor.longscript.enable"	YES
"zend_monitor.longfunction.enable"	YES
"zend_monitor.zenderror.enable"	YES
"zend_monitor.devscript.enable"	YES
"zend_monitor.funcerror.enable"	YES
"zend_monitor.devmem.enable"	YES
"zend_monitor.outsize.enable"	YES
"zend_monitor.memsize.enable"	YES
"zend_monitor.load.enable"	YES
"zend_monitor.custom.enable"	YES

Debugger Directives

Directive	Reload
"zend_debugger.allow_hosts"	YES
"zend_debugger.deny_hosts"	YES
"zend_debugger.allow_tunnel"	YES
"zend_debugger.expose_remotely"	YES
"zend_debugger.network_trace"	NO
"zend_debugger.max_msg_size"	YES
"zend_debugger.httpd_uid"	NO

ZDS Directives

Directive	Reload
"zds.enable"	YES
"zds.mime_types_file"	NO*
"zds.log_file"	NO
"zds.min_file_size"	YES
"zds.disable_byterange"	YES
"zds.mmap_chunk"	NO
"zds.nice"	NO
"zds.allow_assert"	NO
"zds.child_max"	NO
"zds.poll_delay"	NO
"zds.uid"	NO

(*) Not reloaded now – maybe in future versions)

Appendix G - Network Port Requirements

Zend Platform utilizes several network ports for regular component operation. The Zend Platform installer automatically defines these ports based on default settings that are located in the zend.ini. To change the port settings, open the zend.ini and locate the port number (ports are assigned a specific directive).

The following table lists the ports by component and zend.ini directive.

Component	**Directive**	**Port Number**	**Description**
Job Queue	zend_jq.port	10003	Incoming connections for data communication, tcp.
	zend_jq.message_server_port	10004	Incoming connections for messaging service, tcp.
Monitoring	zend_monitor.collector_port	10010	Incoming/outgoing traffic. On Central allow incoming traffic from nodes to port 10010, tcp. On Nodes allow outgoing traffic to central, tcp, port 10010.
Debugger	The port number is controlled by the Studio Client preferences.	10000	Direct connections without Tunneling. Allow outgoing connections from the debug server to the <studio client IP>, port 10000, tcp. Note: may require an available outbound tcp connection to the internet.
	zend_debugger.tunnel_min_port (default 1024) zend_debugger.tunnel_max_port (default 65535) (UNIX, LINUX i5/OS and MAC only)	1024-65535	Connections using Tunneling. Allow connections from node to "tunnel server" (specify "return host" in Studio tunneling settings), to "tcp port range" where a "tcp port range" is defined.

General Comments

- The Firewall should be set to not drop idle connections between the nodes/central (internal in the cluster).
- All nodes/central servers should allow connections to themselves (accept connections to 127.0.0.1).

Appendix H - About SNMP

IN THIS APPENDIX…

AVAILABLE OPERATIONS
SNMP TRAP
SNMP MESSAGE STRUCTURE
THE MIB
THE OID
MIB FILE STRUCTURE
ONLINE MIB VALIDATORS
USING NET-SNMP
OTHER SOURCES OF INFORMATION

SNMP (Simple Network Management Protocol) is a method of monitoring devices or applications from a single location, without the need to check each device/application at any given time.
This is done by having an SNMP agent running on each of the monitored devices. A NMS (Network Management Software/Station) is then installed on a central machine to monitor the activity.
The Rational behind this method is to enable the user (in most cases, the system administrator) to use the NMS in order to 'operate' the monitored devices.
The SNMP agent can inform the NMS about any occurrence.
Communication between the SNMP agent and the NMS is done using UDP, which makes SNMP somewhat unreliable and it is up to the SNMP implementation to make sure messages reach their destination.

Available Operations

The SNMP protocol defines five basic operations (later SNMP include more):

1. Get
2. Get-Next
3. Get-Response
4. Set
5. Trap

The first four operations are available to the NMS and are used to control a monitored device.
The last operation (Trap) is used by the SNMP agent in order to inform the NMS about an occurrence the device. Traps are employed by Zend Platform’s Event Actions and therefore, will be further described.

SNMP Trap

The SNMP Trap is used by the SNMP agent in order to inform the NMS about an occurrence on a device (the same device on which it is installed). An occurrence can be anything ranging from: “detecting a new device”, “device shut down”, “application crash” or “computer temperature is high”. This facilitates the purpose of sending a trap, which is to transmit only essential data to indicate there is a problem. The other SNMP commands (like 'set' and 'get') can then be used to investigate the actual details of the occurrence.

Note:
The following section describes the SNMP V2c message structure which is only slightly different than the SNMP V1 message structure.

SNMP Message Structure

All SNMP messages have the same general structure. They are constructed of Message Headers and the PDU (Protocol Data Unit).

Note:
The PDU may differ in some SNMP operations.

Message Headers

Message Headers contain the following information:

- Version number - Specifies the SNMP version in use (for example '2c').
- Community name – Specifies the access environment for a group of NMSs (for example 'public').

Community names serve as a form of authentication (much like a password).
For example, when the NMS gets an SNMP Trap from an SNMP agent, it checks if the community name that was sent with the trap is authorised to send this trap from this agent.

The PDU

The PDU is the format for sending data in an SNMP operation.

SNMP PDU structure is as follows:

- PDU type
- Enterprise OID
- Agent Address
- Generic trap number
- Specific trap number
- Up-time
- Variable-binding

The MIB

MIB (Management Information Base) is a hierarchically organized collection of information definitions.
MIBs are a collection of managed objects that are identified by object identifiers.
A managed object represents an element of a device. Managed objects are a collection of one or more object instances, which are essentially variables.

Why do we need the MIB ?

The MIB server as a common ground for both the NMS and the SNMP agent (MIB is saved in a text file in a specific structure).
The MIB defines the entities that take part in SNMP communication. For example, traps are set in the MIB file with the definition of the data that can be sent in the trap. This allows the NMS that gets the trap from the SNMP agent to read and interpret the data.

The OID

The OID (Object ID) is a set of numbers, separated by '.', that together assemble a unique identifier that identifies a managed object in the MIB hierarchy.
The MIB hierarchy can be depicted as a tree. For example, the OID: 1.3.6.1.4.1.20815 broken down to each of it's elements, means:

- 1 - ISO
- 3 - Identified Organization
- 6 - DOD
- 1 - Internet
- 4 - Private
- 1 - Enterprise
- 20815 - Zend

The first six elements in the OID are constant. Each organization can register itself with an institute called 'IANA' (Internet Assigned Numbers Authority). Registration with IANA is not mandatory, but strongly recommended in order to maintain order in the Hierarchy (making sure each organization uses it's own unique OID).
Registration with IANA is free from: http://www.iana.org/cgi-bin/enterprise.pl).
Each element in the MIB file can be (and is eventually) represented by OID. However, using a MIB file allows enables to represent each element in a more 'human readable' way.

MIB file structure

The following is an example of a basic MIB file:

```
MIB-NAME DEFINITIONS ::= BEGIN
IMPORTS
enterprises
FROM RFC1155-SMI
OBJECT-TYPE, NOTIFICATION-TYPE
FROM RFC-1212;
zend OBJECT IDENTIFIER ::= { enterprises 20815 }
SimpleTrapExample NOTIFICATION-TYPE
    STATUS current
    DESCRIPTION "This is a simple trap example"
::= { zend 1 }
ComplexTrapExample NOTIFICATION-TYPE
    STATUS current
    DESCRIPTION "This is complex trap example with variable-binding in it"
    OBJECTS { IntVariable, StringVariable }
::= { zend 2 }
IntVariable OBJECT-TYPE
    SYNTAX INTEGER
    MAX-ACCESS accessible-for-notify
    STATUS current
    DESCRIPTION "This is an integer variable"
::= { ComplexTrapExample 1 }
StringVariable OBJECT-TYPE
    SYNTAX STRING
    MAX-ACCESS accessible-for-notify
    STATUS current
    DESCRIPTION "This is a string variable"
::= { ComplexTrapExample 2 }
END
```

The first line defines the name of the MIB (in the sample file, it's called 'MIB-NAME'). Then, the required imports are set (the 'enterprise' import actually sets all the required data up to the 'enterprise' level, which is one level above our organization level, in this case - one level above Zend). From this point, in order to mention the '1.3.6.1.4.1' OID level, all we need to write is 'enterprise', as illustrated below.
It also imports two 'element' definitions: the NOTIFICATION-TYPE (this is actually a 'trap-type' that defines a trap) and OBJECT-TYPE (which defines variables).
Once the required data is imported, we can start setting our MIB.
Now we define 'zend' to be the OID 1.3.6.1.4.1.20815 by stating zend OBJECT IDENTIFIER ::= { enterprises 20815 }
From this point on, we start setting the trap types and the variables.
More detailed information on how to write MIB files can be found in RFC-1212 (http://www.faqs.org/rfcs/rfc1212.html)

Online MIB Validators

Once an MIB file is written it is prudent to pass the MIB file through a MIB validator engine. The validator engine will check to see if the MIB file is written correctly and validates to content.
There are several online validator engines such as*:

- Online MIB Validator (http://www.muonics.com/Tools/smicheck.php)
- MIB module validation (http://www.simpleweb.org/ietf/mibs/validate/)

*Zend Technologies does not indorse or recommend these sites and the links herein are simply provided as examples of online providers.

Using NET-SNMP

NET-SNMP is a free implementation of the SNMP protocol. NET-SNMP that can be obtained from http://www.net-snmp.org/.

Sending an SNMP Trap

Sending a trap using NET-SNMP is quite easy once you know the parameters you need to send and how.
SNMP traps are sent using the snmptrap command. The syntax is:

```
snmptrap -v 2c -c <community string> -M <MIB directory> -m <MIB name> <NMS
address:port> <uptime>
         <<MIB name::>Trap name> <<MIB name::>var> <type> <value>
```

The following describes the different parameters:

- -v - The version of the SNMP protocol to use in order to send the trap.
- -c - The community string to be sent with the trap.
- -M - TThe directory where the MIB is located. If all you want is to add another directory to the MIB dir list, you should place the plus sign (+) before the directory, like this: -M +/my/MIB/dir. Multiple MIBs directories are seperated by ':'.

- -m - The MIBs to be used. If all you want is to add another MIB to the list of MIBs used, you should place the plus sign (+) before the MIB name, like this: -m +MY_MIB_NAME. Multiple MIBs are separated by ':'.
- NMS address:port - The address and port of the target NMS machine.
- uptime - The time passed since the machine came up or since something happened on the machine. You can send an empty string as the uptime.
- Trap name - The name / type of the trap you are sending. Generic trap is only used in SNMP V1, therefore, we only specify the 'specific trap' type. Specifying the MIB name before the trap name (followed by '::') is used to avoid instances where two traps have the same name in different MIBs.
- Variable binding variables - The following elements must be sent together or not at all ! (Several variables can be sent in each trap).
- var -The name of the variable in the MIB. Specifying the MIB name before the variable name (followed by '::') is used to avoid instances where two variables have the same name in different MIBs.
- type - The type of the variable being sent. Several types are available:
- i - for INTEGER
- u - for UNSIGNED
- c - for COUNTER32
- s - for STRING
- x - for HEX STRING
- d - for DECIMAL STRING
- n - for NULLOBJ
- o - for OBJID
- t - for TIMETICKS
- a - for IPADDRESS
- b - for BITS
- value - The value for the variable being sent.

Catching an SNMP Trap (emulating NMS)

In order to catch a trap, you need an NMS, and in case you don't have one installed, you can use the 'snmptrapd' command.

The command syntax is as follows:

```
snmptrapd -P -M <MIB directory> -m <MIB name>
```

The following describes the different parameters:

- -P - This parameter prints formatted incoming traps to stderr (resulting in 'interactive' output). This option is deprecated and can (should) be replaced with -f -Lo.
- -M - Same as in the snmptrap command.
- -m - Same as in the snmptrap command.

Other Sources of Information

- SNMP tutorial - Explains what is SNMP and how it works.
- NET-SNMP project home page (http://www.net-snmp.org/)
- Some short tutorial about SNMP (http://www2.rad.com/networks/1999/snmp/index.htm)
- Another, more detailed, tutorial about SNMP

Index

#
# of Requested ... 123
# of Served ... 123
A
Absolute ... 25, 29
accelerated ... 41, 49
acceleration ... 41, 50
Acceleration Blacklist ... 50
Acceleration Blacklist Files ... 41
Accelerator Memory ... 39
Accelerator Performance Tuning Level ... 111
Action ... 33
Action Rule parameters ... 33
Action Types ... 33
Actions Rules ... 33
Add Jobs ... 123
Add/Edit Users ... 67
Additional Rules ... 25, 26, 30
admin ... 70
Aggregate Hint ... 84
Aggregate Hints ... 92
Aggregate servers ... 79
Aggregation ... 31
Aggregation Rules ... 84
Alert Rule ... 25
Alert Rule event types ... 25
Alert Rules ... 22
Alert Window ... 79, 81
Alerts ... 81
All ... 79
All Files ... 39, 41, 49
ALLGET ... 45
allowed ... 66
Allowed Hosts ... 61
Allowed Hosts for Tunneling ... 61
ALLSESSION ... 45
Always ... 61
Always Report Errors ... 26
Analyze Site ... 100, 102
Apache ... 30
Apache Server MaxClients ... 39
API ... 25, 31, 92
Array ... 45
audit trail ... 84
authentication ... 58
Authentication Information ... 58
Auto Detection Port ... 57
Automatically Connect on Startup ... 58
average ... 25
Average Time in Queue ... 123
Average Waiting Time ... 123
B
Backtrace ... 84
bandwidth ... 140
Bar graphs ... 94
Benchmark ... 101
Blacklist ... 49, 50
Blacklisted ... 41
bottlenecks ... 25, 28
Bridge ... 146
Broadcasting Port ... 58
C
Cache ... 41
Cache Cleaner ... 45
Cache Lifetime ... 45
Cached ... 39, 41, 49
cached files ... 41, 49
Caching ... 45
caching conditions ... 41, 45
Cancel ... 41
central machine ... 70
central registration ... 70
Change Event Report Information ... 90
Change Server ... 111
Choosing and Defining Alert Rules ... 25
Clean ... 41
Client ... 56, 57, 61
Client Debug Port ... 58
Client IP ... 58
Close Event ... 84, 93
Close Selected ... 81
Code Acceleration ... 39, 101
Code Acceleration Enabled ... 39
Code Acceleration Settings ... 39
communication ... 57
communication tunnel ... 56, 57, 58
Completed Jobs ... 123
compressed ... 41, 49
compression ... 41, 49, 50
Compression Blacklist ... 50

Compression Blacklist Files........................41
Compression Test 100, 101, 102
Compression Test results 102
concurrent jobs..................................... 140
Concurrent Requests 140
configuration 63
configurations...41
Configure Alert Rules.......................... 22, 79
Configure communication..........................57
Configure performance 39
Configure Preferences for Tunneling 57
Configuring Zend Studio Client Tunneling
Settings 58
Confirm Password 67
Connection errors 30
Content 45
Content Caching 101
Cookie 45, 57, 101
CPU...25
CPU Load 84
create a New Group 68
critical situations.....................................31
Current Blacklisted Files...........................50
Current Existing Users 66
Current Jobs... 123
Custom.. 111
Custom Event...25
Custom events 31, 84
D
daemon 146
data 39
Database 30, 79
database cleanup....................................81
Database Error 25, 30
database function errors 30
Database Maintenance.............................93
Database selection errors.........................30
databases 30
Deactivate compression 49
debug... 56, 57
Debug Mode 58
Debug Preferences 58
Debug URL..84
Debugger...................................58, 61, 84
Debugger Server URL 58
Default Cache Lifetime........................ 39, 45
default configurations 22
Default Dynamic Caching Conditions 39
Define Action Types 33
Define Alert Rules................................... 22
Define Cache 41
Define Caching Settings 41
Delete Event 84, 93
Delete Selected 81
Denied Hosts 61
deny tunneling 61
diagnostic.. 84
directives... 63
Disable an Action Rule............................. 33
Dismatch 45
Do not Cache 41
document root 41
Download 140
Download Server 140
Download tab 140
Download Test 140
Dummy File 58
Duration 101
Dynamic.. 45
Dynamic Caching Enabled....................39, 45
Dynamic Content Caching....... 39, 45, 49, 101
Dynamic Content Caching Settings 39
E
edit..56, 66
Editing a User 67
Editing Groups 68
Editing Users.. 67
e-mail 33, 84, 100
Email Report 33
Encrypt Communications using SSL 58
End User 79
Enhanced 111
Environment 146
Error Description 84
error-reporting criteria 26
Errors 26
establish.. 56
establish a communication tunnel 57
Event 26, 27, 28, 29, 30, 31
Event Data 84
Event List 81
Event Name 79
Event Occurrence Info............................. 84
Event Report............................... 33, 84, 90
Event Reports 33, 84
Event Summary 79

Event Summary table 79
Event Type...................................... 22, 81
Event Types 22
events 30, 31, 79, 81, 92
Events From 22, 81
Excess Memory...................................... 29
Excess Memory Usage 25, 29
expandable lists..................................... 63
extensions 39
Extreme... 111
F
File Compression........................ 39, 49, 101
File Settings 41
File View 41
Filtering 79, 81
Filtering Alert Rules 22
Filtering Table Data 79
Fine Tune Caching Conditions 41
firewall 57
Full Filepath Identification 39
Full Page Content Caching 45
function 30
Function Data 84
Function Error 25
function errors................................. 27, 30
Function Execution 28
Function Usage 31
functions... 27
G
General database function errors 30
Get 45, 57, 101
Getting Support 209
Global Events 92
global variable 92
graphical representatio 79
graphical representation 79, 94
Graphs 79
Group... 66, 79
Group Name 66
GUI 70
gzip.. 49
H
Handle Groups 66
Hard errors 26
health... 31
Host 79
HTTP Authentication 58, 101
HTTP Server 41
I
IDE 56
Ignore Event 84, 93
Ignore Selected 81
Included Files....................................... 84
Inconsistent Output Size 25, 31
information 209
Integration 57, 84
investigating 84
IP 56, 61
J
Java Bridge 146
Java Environment 146
Java Home.. 146
Java Status 146
Java Vendor 146
Java Version 146
Job-Queue 120
Jobs 120
L
Last Download 140
Load Average 25, 31
load balancer 57
Log 39
Log File 39
login 66
M
management 66
Manual Override 93
Match... 45
Max Clients 140
MaxClients 39
Maximum Accelerated Files 39
Maximum Cache Size 39, 45
Maximum Cached File Size.................. 39, 45
Memory 29
Memory Reclaim Threshold 39
Minimum File Size.................................. 39
Minimum Free Disk space 45
Minimum Free Diskspace 39
Modified File Detection 39
Monitor 22, 31
monitor_set_aggregation_hint 92
Monitoring 39, 79
multiple users 66
N
N/A Test Results................................... 102
Net Mask 61

Never 61
New Group 68
New Users 67
node 39, 61
None 39, 49
Normal 111
Notices 26

O

on demand 56
On-Demand 56
On-Demand Communication 56
Only Cached Files 39, 49
operator 26
OS Name 146
OS Version 146
output site 79
Output Size 25

P

parameters 26, 57
Partial Page Content Caching 45
Password 58, 63, 67, 70
Password Administration 70
Password Specifications 70
Password Structure 70
Performance 41, 100, 101, 102, 120
performance configurations 41
Performance Lifecycle 22
Performance Monitoring Event .. 25, 26, 28, 29
Performance Test 102
Performance Test results 102
Performance Tuning 111
Permission Group 66
permissions 67, 68
Permissions Group 67
PHP 63, 79
PHP Error 25, 26
PHP events 79
PHP function 28
PHP Intelligence 79
PHP Section 63
PHP Settings 63
php.ini 63
Pie 94
Port 56, 58
Preserve Event 84
Preserve Selected 81
Previous Test Results 102
processes 25, 26, 31, 120
Product directives 63
Production 27
profile 56, 57, 84
Profile URL 84

Q

QA 26, 27
Query Execution 25, 30
Queue 123
Queues 120

R

Refresh 146
regexp 45
registration 70
Regular Expression 45
Regular Expressions Format 45
Relative 25, 26, 29
Relative Events 29, 31
Remove 66
Reopen Event 84
Reopen Selected 81
reports 39
REQUEST 45
Requested URL 84
requests 26
resolving events 84
Restart 146
Restore Defaults 41
rule-based notification 33
Rules 22, 33

S

Save 41
scheduling 120
script 101
scripts 29, 31
Selective 61
Send a report via e-mail 33
sending the information 33
Server 45, 79
Server by name 79
Server Name 79
Server Response Timeout 58
SESSION 45, 101
settings 39, 61, 120, 123
severe event 27
Severity 81
Severity Level 84
Show Source Code 84
silence operator 26

silenced 26
simulation 140
Site Analysis 102
Site Analysis Report 102
slow function 25
Slow Function Execution 25, 28
slow queries 25
Slow Query 30
Slow Query Execution 25, 28, 30
Slow Script Execution 25, 26
Slow Script Execution Absolute 25
Slow Script Execution Relative 25
solve problems 209
source code 84
Source Info 84
Specify Return Host 58
SSL 58
SSL encryption 58
Start 146
Statistics 123, 146
Status 81, 84, 146
Stop 146
Structure 70
Studio 56
Studio Client 56, 58, 61
Studio Client Tunneling 58
Studio Server 61
Studio Server parameters 57
Studio Server Settings 61
Submit a report to a Specified URL 33
Support 209
Support Center 209
Support Tool 209
supported databases 30
Suppress 25
Suspend 123
System Passwords 70

T

table 79
Test 57, 101, 140
Test Download 100, 140
Test Report 140
Test Results 101, 102, 140
Test URL 84, 100, 101
Testing 100, 102, 140
Time Filter 81
Title 84
Tool 209
Tree View 41
Tuning 111
Tunnel Target Host 58
Tunnel Target Port 58
Tunneling 56, 57, 58, 61

U

Undo 41
Ungrouped 79
un-zipped 49
Update Blacklists 41
Update the Blacklist 50
URL 33, 101, 140
User 66
User Forums 209
User Management 66, 68
User Name 58, 67
users 67

V

variable matches 45
Variables 45, 84, 90, 101
variables data 90
vhost 84
Virtual Host 81
Virtual Hosts 41
Virtual Hosts List 41

W

Warnings 26
Watched Functions 27
Web application 101
wildcards 61

X

XML 84
XML data 33
XML Report 33

Z

ZDS 140
Zend Download Server 39, 100
Zend Error 84
Zend Monitor 22
Zend Product directives 63
ZEND Section 63
Zend Studio 58, 84
Zend Studio Client 56, 58
Zend Studio IDE 57
Zend Studio Integration 57
zend.ini 63