User Guide:

Zend Core 2.5 for i5/OS

By **Zend Technologies**

Zend
The *php* Company

Disclaimer

The information in this document is subject to change without notice and does not represent a commitment on the part of Zend Technologies Inc. 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 in accordance with the Zend Core for i5/OS End User License Agreement (EULA).

Zend Core for i5/OS User Guide issued November 2007.
Product Version: 2.5

DN: ZCI5OS-UG-271107-2.5-006

In This Guide

This guide is intended for System Administrators and PHP developers who manage the PHP Web Servers in their organization. This guide covers the functional aspects of Zend Core for i5/OS along with in-depth explanations on how to get support and other services from Zend Technologies.

Throughout this guide are instructions for guiding the reader to extra reference information about various products featured or implemented in Zend Core for i5/OS.

Table of Contents

Introduction

Zend Core™ supports businesses using PHP and managing database information for mission critical web applications. It provides a seamless out-of-the-box experience delivering a stable, easy to-install and supported PHP development and production environment.

Presented in a browser-based environment, Zend Core provides a highly stable and efficient means for installing and managing PHP servers. Resources and reference information are bundled into Zend Core for “one click” access to a wide range of information, configurations and reference documents.

Using Zend Core ensures that organizations work with a stable, certified, binary distribution of PHP. In other words, Zend Core provides a constantly supported and updated generic code base. An organization’s PHP will therefore be easily understood so that newcomers or external consultants can quickly get up to speed with the new environment.

Installing Zend Core

To install Zend Core for i5/OS in 'silent' mode (no interactive dialogs):

When the SAVF is loaded into the i5 QGPL library run the following command:

```
SBMJOB CMD(RSTLICPGM LICPGM(1ZCORE5) DEV(*SAVF)
SAVF(QGPL/ZCORESAVF))
```

Zend Core for i5/OS will be automatically installed without interactive dialogs being displayed.

To install Zend Core for i5/OS in 'interatctive' mode:

1. When the SAVF is loaded into the i5 QGPL library, run the following command:

```
RSTLICPGM LICPGM(1ZCORE5) DEV(*SAVF) SAVF(QGPL/ZCORESAVF
```

2. Follow the on-screen instructions.

Refer to the Zend Core for i5/OS Installation Guide for complete installation instructions. This is located in the Zend Core for i5/OS Installation Package or can be downloaded from the Zend Core Resources page, located at http://www.zend.com/en/products/core/for-i5os.

Updating Zend Core

Users who have purchased a Zend Core Support Subscription (Silver, Gold or Platinum) can get periodic Zend Core Updates, including bug and security fixes. By downloading Zend Core for i5/OS, you have received a one year, first-level Silver Support Subscription.
For more on Zend Support Subscriptions, and to register for other programs, see the Zend Core Support page at http://www.zend.com/en/products/core/support.

You can download and install one or more Updates using the Zend Core Setup Tool.
This method is only applicable for systems with a direct internet connection.

Note:
Updates and Zend Update Packages are only available to Zend Core Support Subscribers. You must therefore ensure that your correct Zend Core Support Network User ID and Password (or Zend account login details) are configured in your Zend Core. To configure these settings, open the Zend Core Setup Tool and select Update via Zend Network Menu | Change Network ID user/password.

This procedure describes how to download and install Updates using the Zend Core Setup Tool.

To download and install Updates using the Zend Core Setup Tool:

1. Open the Zend Core Setup Tool by running the command "go zendcore/zcmenu" in your i5/OS emulation screen.
2. Select Update via Zend Network Menu | Update Zend Core components | Zend Network Update.
3. A list of available Updates will be displayed.
4. To install all available Updates, press F10.
   To install only certain Updates, select the required components and press F8.

The Updates will be downloaded and installed.

Note:
It is recommended to test the Updates on a test environment before applying them to a production environment.

Consistency Checking

The Zend Core Setup Tool performs consistency checks to prevent you from installing Updates which are dependent on other, uninstalled Updates.
However, if necessary you can force the download of these Updates through the Zend Core Setup Tool. (Not recommended.)

Rollbacks (Removing previous Updates)

Update transactions are preserved so that Updates can be removed and the system can be 'rolled back' to a previous state. This action is referred to as a Rollback. Rollbacks are stored and displayed with the date and time that your system was modified and Updates installed. Selecting a Rollback will revert your system back to the state it was in on the selected time/date.

Rollbacks can be performed through the Zend Core Setup Tool.
This procedure describes how to execute a Rollback to a specific recent system state, using the Zend Core Setup Tool.

To execute an attended Rollback:

1. Open the Zend Core Setup Tool by running the command "go zendcore/zcmenu" in your i5/OS emulation screen.
2. Select Update via Zend Network Menu | Rollback Options | Rollback Components.
   A list of previous system states will be displayed, listed according to the date and time that new Updates were installed.
3. Select a required system state and click OK.

Any Updates installed since this version will be deleted.

Note:

Rollback information is stored so that your system can be restored to a previous state.
Two parameters have been defined to ensure that the rollback feature consumes the least disk space possible:
1. Maximal disk space allocated for backups (Default maximum disk space consumption = 100Mbytes).
2. Only the last ten update transactions will be stored, provided they do not exceed 100Mbytes.
In addition, you can choose to delete Rollback information (see below).

To delete Rollback information:

1. Open the Zend Core Setup Tool by running the command "go zendcore/zcmenu" in your i5/OS emulation screen.
2. Select Update via Zend Network Menu | Rollback Options | Delete Rollback Information.
   A list of available Rollbacks will be displayed.
3. Select a Rollback and click OK.

System state information contained in the Rollback will be deleted.

Zend Core Setup Tool

The Zend Core for i5/OS Setup Tool allows you to configure all aspects of your Zend Core system, and lets you download and install Updates and additional components.

The Zend Core for IBM i5/OS Setup Tool can be opened by logging into your emulation screen and running the following command:

```
go zendcore/zcmenu
```

The Zend Core Setup Tool has 6 main options:

```
  ZCMENU              Zend Core for IBM i5/OS Setup Tool
                            System:   I5RND5R3
Select one of the following:
1. Set Zend Core Web Administration Console password
2. Update via Zend Network menu
3. Run Support Tool
5. Service Management menu
6. MySQL Management menu
7. System Information and Server IDs
90. Signoff

Selection or command
===>
F3=Exit   F4=Prompt   F9=Retrieve   F12=Cancel
 F23=WRKUSRJOB
```

Figure 1 - Zend Core Setup Tool

1. Set the Zend Core Web Administration Console Password - Provide a password for accessing the Zend Core Web Administration GUI.
2. Update via Zend Network menu - The Zend Core for i5/OS update mechanism is used to upgrade installations. The Zend Network update mechanism enables automatic downloads and installation of Updates.
3. Run Support Tool - The Zend Support Tool is a tool for gathering information about user configurations and setup. This tool allows the Zend Support Team to solve problems in a more comprehensive and efficient way.
5. Service Management - Manages all required services, e.g., Zend Core Subsystem and Apache Control.
6. MySQL Management Menu - Allows you to control your MySQL processes and services.
7. System Information and Server IDs - Displays i5/OS System information.

Option 1 - Set the Zend Core Web Administration Console Password

Allows you to change your password for accessing the Zend Core Administration Web GUI.

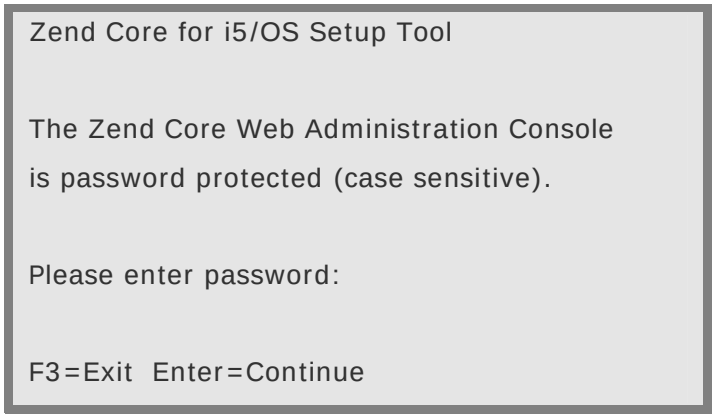


Figure 2 - Zend Core Web Administrative Console Password Screen

Enter a new password and press Enter.
You must restart your web server after changing your password.

Option 2 - Update via Zend Network

Note:

You must be registered for a Zend Support Subscription (Silver, Gold or Platinum) in order to have access to Updates.

By downloading Zend Core for i5/OS, you have received a one year, first-level Silver Support Subscription.

For more on Zend Support Subscriptions, and to register for other programs, see the Zend Core Support page at http://www.zend.com/en/products/core/support.

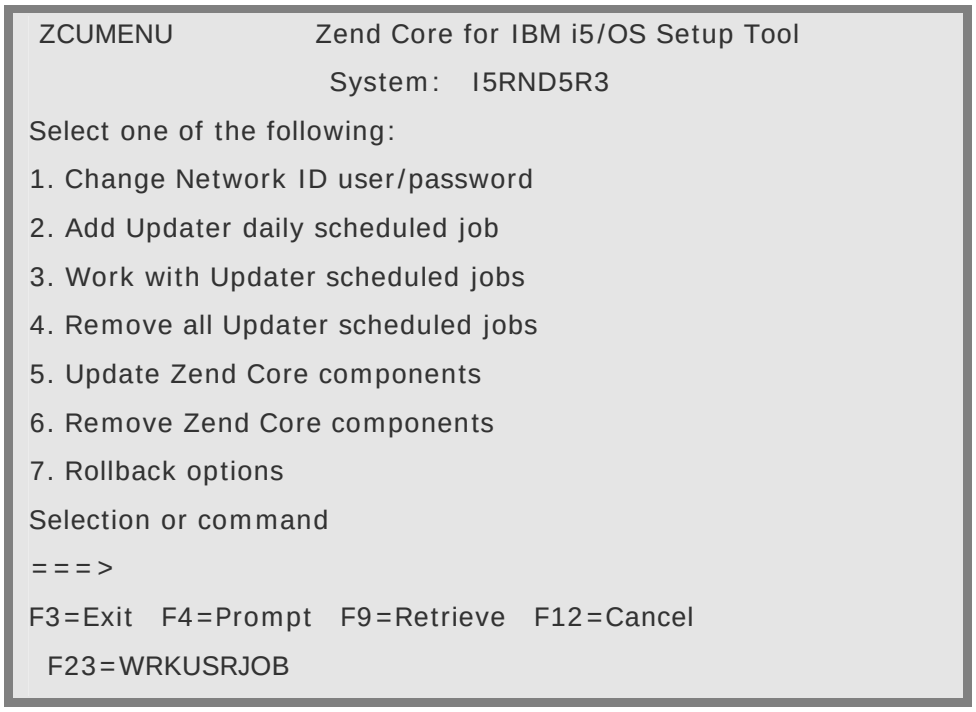


Figure 3 - Zend Core Setup Tool - Update via Zend Network

The Update via Zend Network menu includes the following options:

1. **Change Network ID user/password** - Allows you to specify/edit your Zend Core Support Network User ID and Password. You can also use your Zend account login details.
   You must enter a Zend Core Support Network User ID and Password in order to be able to find and install Updates.

2. **Add Updater daily scheduled jobs:**
    - Enter the time to check for daily updates (or accept the 01:00 AM default) and click Enter.
    - Select one of the following options for the Updater to perform at the specified time:

- List available Updates - Creates a file in your temp directory containing all available Updates.
- Get available Updates - Downloads and installs all available Updates.

3. **Work with Updater scheduled jobs -** Select the required parameter or job and select an action by entering the appropriate number option.

```
                    Work with Job Schedule Entries
              I5RND5R3
                                10/21/07  15:25:39
     Type options, press Enter.
     2=Change  3=Hold  4=Remove  5=Display details  6=Release
     8=Work with last submission  10=Submit immediately
                                  Next
     -----Schedule------          Recovery  Submit
     Opt  Job       Status  Date     Time     Frequency  Action  Date
     ZC_UPD_LST   SCD   *ALL     01:00:00  *WEEKLY   *SBMRLS
      10/22/07
                                   Bottom
     Parameters or command
     ===>
     F3=Exit  F4=Prompt       F5=Refresh  F6=Add   F9=Retrieve
     F11=Display job queue data  F12=Cancel  F17=Top
      F18=Bottom
```

Figure 4 - Work with Updater Scheduled Jobs

The options are:

2. Change
3. Hold
4. Remove
5. Display details
6. Release
8. Work with last submission
9. Submit immediately.

4. **Remove all Updater Scheduled Jobs** - Automatically removes scheduled Updating jobs.
5. **Update Zend Core Components -** Allows you to find and install Updates from the Zend Network.

Note:

You must be registered for a Zend Support Subscription (Silver, Gold or Platinum) in order to have access to Updates.

By downloading Zend Core for i5/OS, you have received a one year, first-level Silver Support Subscription.

For more on Zend Support Subscriptions, and to register for other programs, see the Zend Core Support page at http://www.zend.com/en/products/core/support.

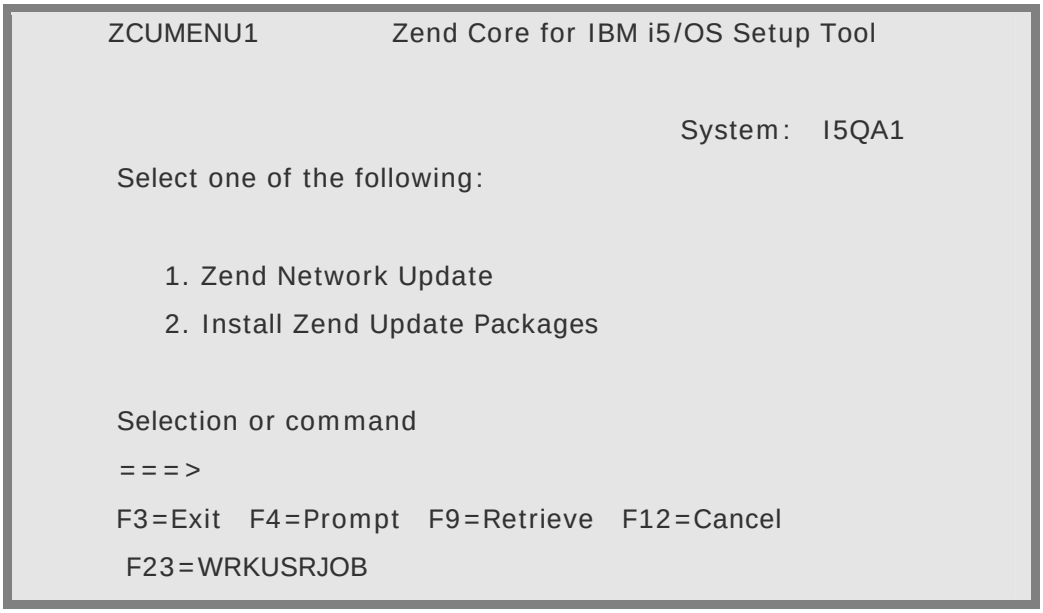


Figure 5 - Update Zend Core Components menu

- Select Option 1 - Zend Network Update - to view and install available Updates.

Note:

You must have configured your Zend Network User ID and Password in Zend Core during the installation process or using the Setup Tool (see the 'Change Network ID user/password' option, above) to be able to download Updates.

- -Or- Select Option 2 - Install Zend Update Packages - if you have previously downloaded Zend Update packages from the Zend Network site.

  Zend Update Packages are files containing a set of the latest Updates. Downloaded Zend Update Packages must be placed in the folder /usr/local/zend/core/setup/dl before they can be installed using the Setup Tool.

6. **Erase Components** - Allows you to delete Zend Core components, including libraries and extensions. If you have chosen to delete a component which other installed components are dependent upon, a prompt will appear asking for confirmation of the components' deletion.
7. **Rollback options**- Allows you to view and carry out Rollbacks, or to delete Rollback information. Rollbacks will delete Updates and revert your Zend Core back to a previous state.

   For more information, see the 'Rollbacks' section under the 'Updating Zend Core' chapter.

Option 3 - Run Support Tool

The Zend Support Tool is a tool for gathering information about your system configuration and setup. This tool allows the Zend Support Team to solve problems in a more comprehensive and efficient way.

To create a file containing the above system information which can be sent to the Zend Support Team, specify the destination directory where the file will be created.

After the file is created it can be sent to Zend Support if the need for support arises.

See Appendix A - Support Tool Information for a complete list of the information collected by the Support Tool.

Note:

For more on Zend Support, and to subscribe to a Support program, see http://www.zend.com/en/products/core/support.

Option 5 - Service Management

This menu allows you to control your Zend subsystem, Apache web server and the PHP toolkit service I5_COMD.

```
ZCAMENU            Zend Core for IBM i5/OS Setup Tool
                        System:   I5RND5R3
Select one of the following:
1. Start Zend Core Subsystem
2. Stop Zend Core Subsystem

4. Start Apache server instances
5. Stop Apache server instances
6. ReStart Apache server instances
7. Additional Apache options
8. Start i5_COMD service
9. End i5_COMD service
Selection or command
===>
F3=Exit   F4=Prompt   F9=Retrieve   F12=Cancel   F23=WRKUSRJOB
```

Figure 6 - Service Management Menu

The Service Management menu includes the following options:

1. Start Zend Core Subsystem - Starts the Zend Core process
2. Stop Zend Core Subsystem - Stops the Zend Core process
4. Start Apache server instances
5. Stop Apache server instances
6. ReStart Apache server instances
7. Additional Apache options - Allows you to start/stop additional instances in the IBM HTTP or PASE Apache Servers.
8. Start i5_COMD service
9. Stop i5_COMD service

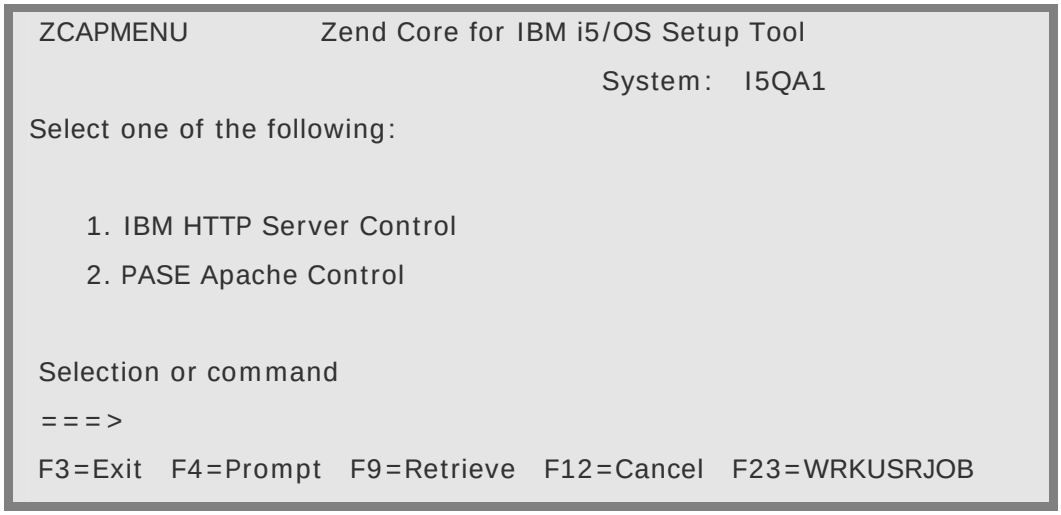


Figure 7 - Additional Apache Options

- Select Option 1 - IBM HTTP Server Control - to control your IBM HTTP Server instances. In the following screen, enter the Instance Name and select the action you would like to perform by entering S (start), E (end) or R (restart) in the 'Action' category.
- Select Option 2 - PASE Apache Control to control your PASE Apache instances. In the following screen, enter the configuration file name's location and name and select the action you would like to perform by entering S (start), E (end) or R (restart) in the 'Action' category.

8. Start i5_COMD service - Allows you to configure your PHP Toolkit Daemon.

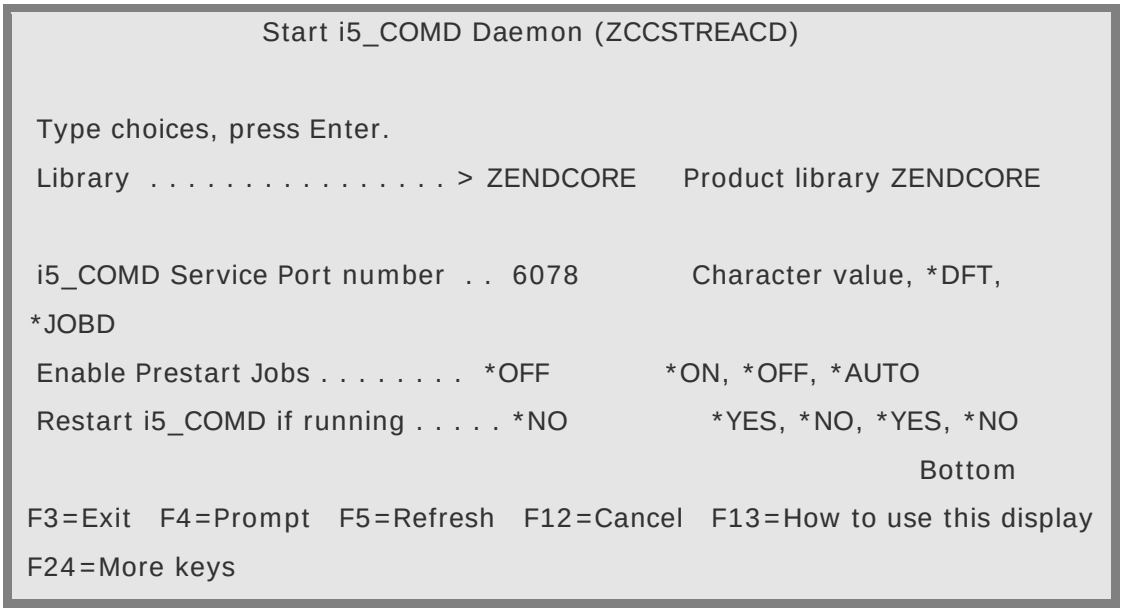


Figure 8 - i5_COMD Daemon Options

Note:

If you change the i5_COMD Service Port number, the daemon will open on a different TCP/IP port number. The new port number (i5comm.port entry) is updated in the /usr/local/Zend/Core/etc/php.ini file.

9. End i5_COMD service - Stops the PHP Toolkit Daemon.

Note:

To apply changes, stop and start the Zend Core subsystem by selecting Options 2 (start) and 1 (stop) on the System Management Menu.

Option 6 - MySQL Management menu

```
ZCMYSQL                Zend MySQL management.
                            System:   I5QA1
Select one of the following:
1. Start MySQL subsystem
2. Stop MySQL subsystem
4. Start MySQL daemon
5. Stop MySQL daemon
Selection or command
===>

F3=Exit   F4=Prompt   F9=Retrieve   F12=Cancel   F23=WRKUSRJOB
Copyright Zend Technologies LTD (2007)
```

Figure 9 - MySQL Management Menu

The MySQL Management menu includes the following options:

1. Start MySQL subsystem - Starts the MySQL process
2. Stop MySQL subsystem - Stops the MySQL process
3. Start MySQL daemon - Starts the MySQL service. The MySQL Daemon allows access to the MySQL database.
4. Stop MySQL daemon - Stops the MySQL service.

Note:

If MySQL is not installed, selecting the MySQL Management menu option will prompt you to install MySQL. See the Zend Core for i5/OS Installation Guide for more on installing MySQL.

Option 7 - System Information and Server IDs

Displays System Information.

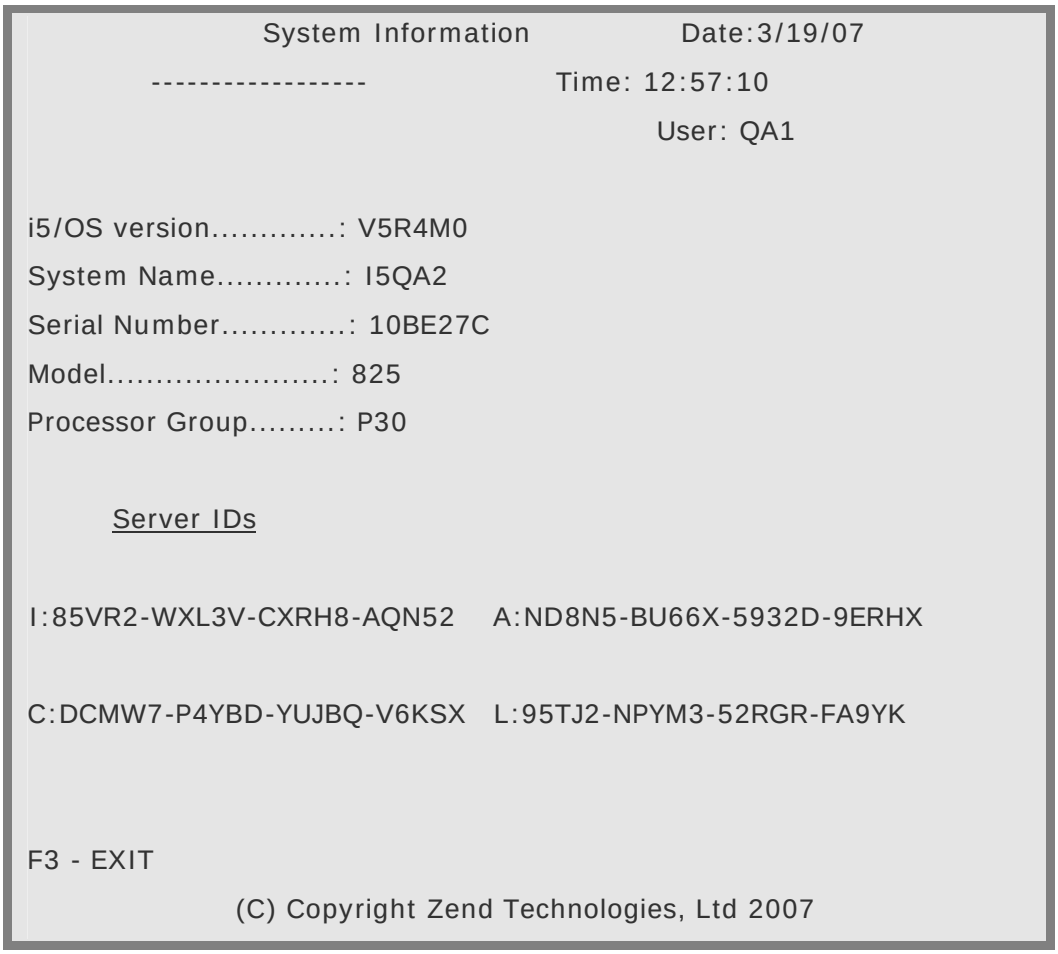


Figure 10 - System Information

The following information is displayed:

- i5/OS version
- System Name
- Serial Number
- Model
- Processor Group
- Server IDs

Getting Started

General Information

Zend Core Environmental Variables

The following items are Zend Core Environmental Variables:

Item	Explanation
SBS	Subsystem
JOBQ	Job Queue
JOBD	Job Description
CLASS	Responsible for process attribute time slots from the CPU.

Zend Core Subsystem

Zend Core auto startup jobs such as PRNGD and APACHE are grouped, and run under the Zend Core subsystem.

Logging In

Once the installation process has completed, you will be ready to login to your Zend Core Administration Web GUI.

To access the Zend Core Administration Web GUI :

1. Enter your i5/OS machine's IP, and port number 89 as follows:
   http://IP_address:89
   The Zend Core for i5/OS Welcome Screen will appear.

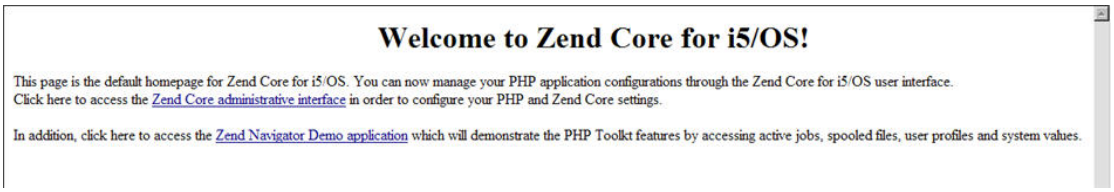


Figure 11 - i5 Welcome Screen

2. Click the 'Zend Core administrative interface' link.
   The Zend Core for i5/OS login screen will appear.

Figure 12 - Zend Core Login Screen

3. To login, type the password you defined in the installation process and click the arrows ⏩.
4. Your Zend Core Administration Web GUI will open.

Note:
If you installed Zend Core for i5/OS in silent mode, the default password will be "zend".

Changing your Zend Core GUI Password

Your Zend Core login password can be changed from the Zend Core Setup Tool in case it is misplaced or needs to be changed.

To change your password:

1. Open the Zend Core Setup Tool by running the following command in your i5/OS emulation screen:

```
GO ZENDCORE/ZCMENU
```

2. Select Option 1 - Set Zend Core Web Administration Console password.

```
          Zend Core for i5/OS Setup Tool

          The Zend Core Web Administration Console
          is password protected (case sensitive).

          Please enter password:

        F3=Exit  Enter=Continue
```

Change Password - Zend Core Setup Tool

3. Follow the instructions and confirm your selection.
4. You must restart your web server for the settings to take effect.
   To restart your server:
   In the Zend Core Setup Tool main menu, select Option 5 - Service Management menu and then select Option 6 - Restart Apache server instance.

Zend Navigator Demo Application

This Zend Navigator Demo Application shows the usage of i5 Toolkit functions.

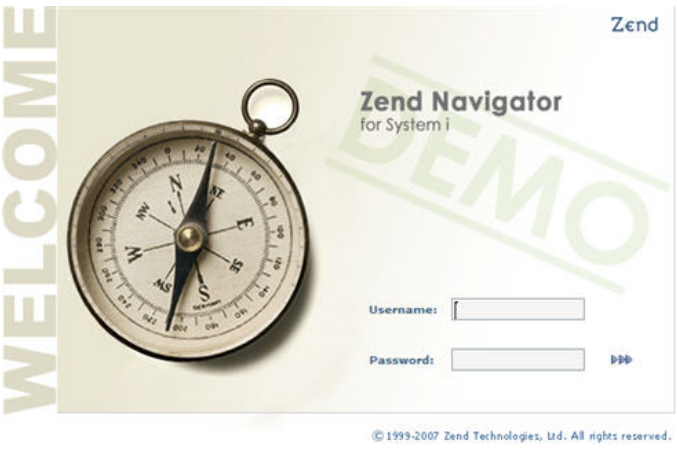


Figure 13 - Zend Navigator login screen

The Zend Navigator demo application can be found in the following directory: /www/zendcore/htdocs/Zend_Navigator_Demo

The demo application has the following options:

- Logon - This option allows you to logon to System i with a valid System i user profile
- Active Jobs - This option allows you to view all active jobs. You can use the "Load Subsystem" filter to view jobs in selected subsystems. Clicking on a job line will display some job details and job log.
- Spooled Files - This option allows you to view current user spooled files. You can use the 'Load User' user filter to display other users' spooled files. Clicking in a spooled file line will display the spooled file content. Some spooled file details and spooled file options such as DELETE, HOLD/RELEASE or display spooled file content, will be displayed in PDF format.
- System Value - This option displays a full list of all System i System values.
- User Profiles - This option allows you to view all user profiles on your server. Clicking on a user profile line will display some user details and user status options, and will allow you to enable or disable selected user profile statuses.
- Database files - This option allows you to view all database files on your server. Selecting a library from the drop-down list and clicking a file will display all data in the file. Click the 'File Description' button in the file display to see more information on the file.

To run the demo application, go to: http://<Your i5_TCP_address>:<port_number>/Zend_Navigator_Demo/login.php

Functional Overview

Zend Core provides a “single point of access” for configuration, documentation, support, monitoring and control of your PHP and Web Server as follows:

- **PHP Configuration** - Configure and view existing PHP configurations and changes in a phpinfo display.
- **Reference Information** - Search reference information included in Zend Core to get immediate answers to questions. Zend Core provides advanced search functionality by searching across all included reference information at once.
- **Server Monitoring and Control** - View the overall condition of the server.
- **Extension Configuration** - Control the extensions loaded in your environment.
- **Benchmarking** - Measure performance standards to make your applications more efficient.

Zend Core and Zend Framework

Zend Framework is a high quality open source framework for developing Web Applications and Web Services with PHP.
The Zend Framework is a collection of common PHP classes which sits above the PHP layer. It packages classes and code, used for common functions such as connecting to databases and creating PDF's, into one easy-to-use application.
Using the Zend Framework negates the need for developers to rewrite already existing code, and so significantly speeds up the development process as well as providing the stability ensured by the use of proven code patterns.
While Zend Framework provides an almost ready to use application, it also grants complete flexibility, allowing developers to adapt the application to their own needs.
The expertise of the qualified PHP developers who have worked on the project have ensured a high-quality, stable tool. Zend Framework is covered by unit tests, automatic self-testing mechanisms which ensure that the Framework is constantly tested and monitored.
In addition, Zend Framework has a clean IP. All contributors to the project have signed a contributor license attesting that their contributions have not been previously copyrighted, thus ensuring peace of mind for developers and allowing free use of all content within the Framework.

For more on Zend Framework, visit the Zend Framework Homepage at http://framework.zend.com.

Installation

The Zend Framework comes bundled with Zend Core for i5/OS and will be automatically installed during installation.

Updating

As Zend Framework is an open source project, new updates are constantly being added.
For the latest Zend Framework news and updates, make sure you are added to Zend Framework's mailing list:
http://framework.zend.com/wiki/display/ZFDEV/Contributing+to+Zend+Framework#ContributingtoZendFramework-Subscribetotheappropriatemailinglists
Alternately, visit the Zend Framework portal in order to see the latest Zend Framework news: http://framework.zend.com

Note

During the Zend Core installation, the Zend Framework library will be placed in a folder entitled "ZendFramework".
By default, this can be found in:/usr/local/Zend/ZendFramework

Loading Zend Framework classes

Once Zend Framework's library has been added to your include path, there are two ways to load Zend Framework's classes in your script:

1. Using the Zend Loader:

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

To use the Zend Loader:

1. Load the Zend Loader utility class once in your script:

```
Require_once 'Zend/Loader.php';
```

2. From now, load each class using the class name:

```
Zend_Loader::loadClass('Zend_Class_Name');
```

For example, in order to load the Zend Http Client:

```
Zend_Loader:: loadClass('Zend_Http_Client);
```

2. Using require / include calls

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

To use 'require class':

1. Enter a 'require' command for the relevant file into your script:

```
Require 'File.php';
```

For example, to require the Zend Http Client Class:

```
require 'Zend/Http/client.php';
```

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

User Interface

The Zend Core Web Administration GUI is a tab-based environment for navigating through the Main Menus. Each of the Main Menus include tabbed sub-menus that change according to the active tab.

The Main Menus and Menu Options include the following:

Main Menu	Menu Options
Control Center	System Overview
	phpinfo
	Benchmark
	Support
	Updates
Configuration	PHP
	Extensions
	Zend Products
	Misc. Directives
	Zend Studio Server
Documentation	PHP Manual

Note:

Each one of the above-mentioned options is described in detail in their own dedicated section in the Zend Core for i5 User Guide.

Control Center

The Control Center is the main system-profiling component for monitoring, testing and configuring server performance and activity. The tab's functionality provides System Administrators with an overall display, essential information, and URL testing capabilities.

The tabs included under the Control Center are:

- **System Overview** - Displays information about the server's environment and activities.
- **phpinfo** - Displays information about the current state of PHP.
- **Benchmark** - A performance standard for measuring Web Server performance and durability.
- **Support** - Instant access to Online Resources.
- **Updates** - View all available Updates.

System Overview

The System Overview tab is a server-monitoring screen that provides valuable information regarding the server's environment and activities. This screen collects information for immediate display that would otherwise require the time consuming task of searching for these details.

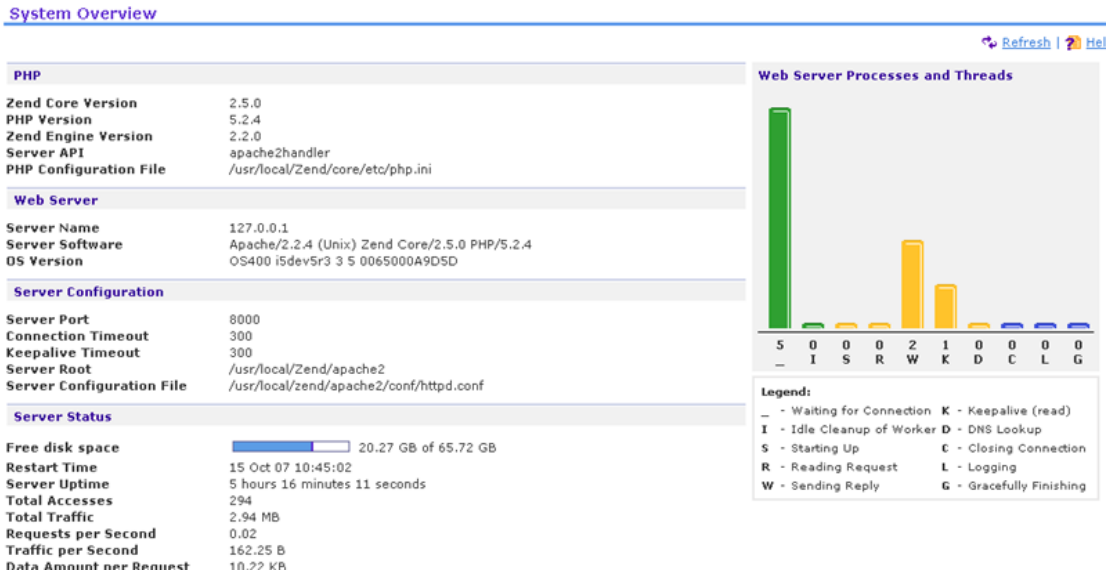


Figure 14 - Control Center Tab – System Overview

The System Overview Screen displays the following information:

PHP

Displays information about the PHP version installed on the server. The information includes which Server API the PHP uses and the location of the PHP configuration file on the server.

Web Server

Displays the Web Server's name and details about the Operating System's environment.

Server Configuration

Displays the Web Server's port number, root directory and connection time-out durations, along with the location of the Server Configuration file.

Server Status

Displays an accumulated list detailing the various activities on the server.

Disk Space

Displays the amount of free disk space available, displayed through bar charts of the partitions and free disk space that give an easy view of the server's disk space status.

Web Server Processes and Threads

This bar graph display shows a snapshot of the various threads that are running on the server.

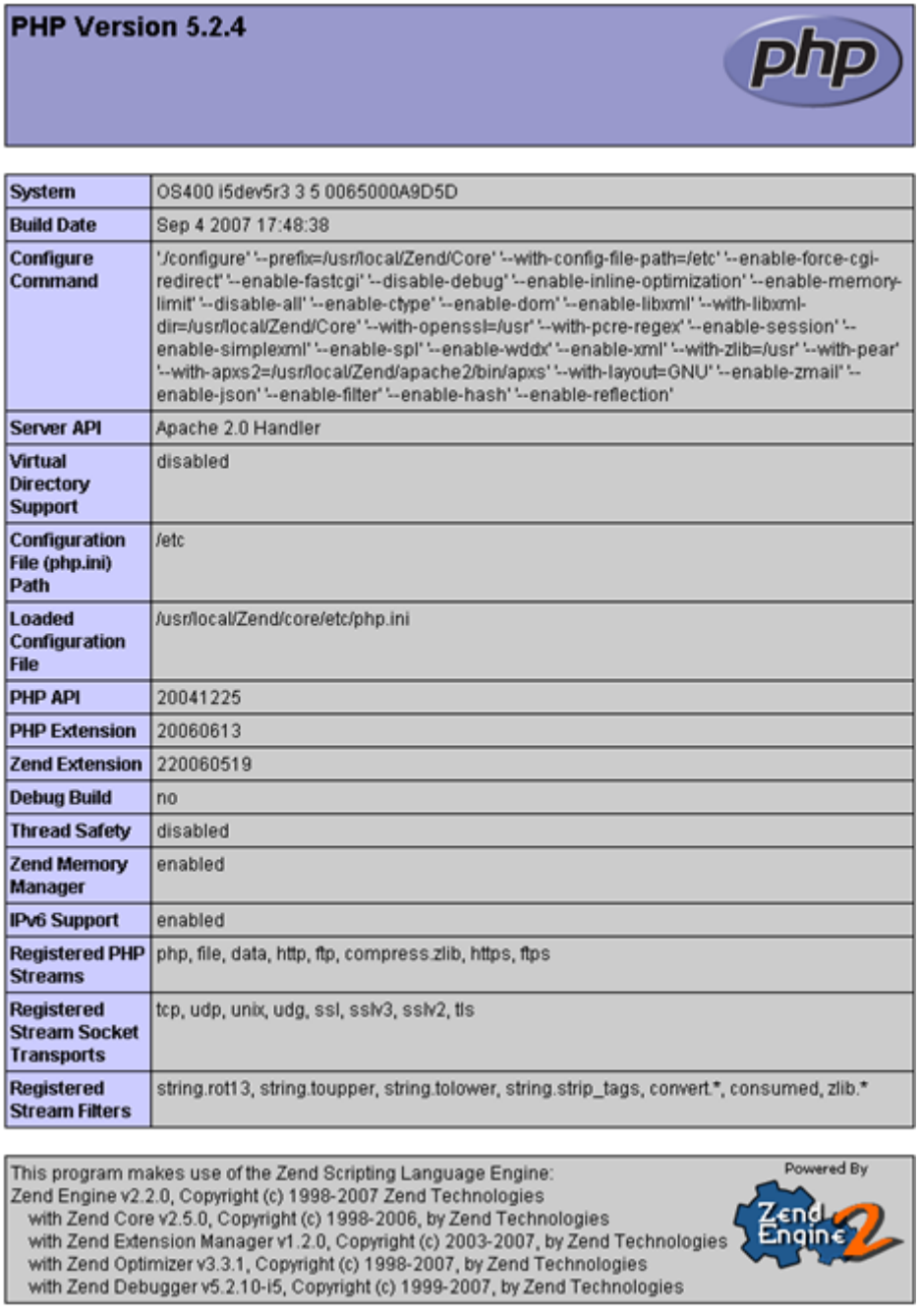
Note:

The System Overview screen information can be refreshed using the Refresh button 

situated in the top-right corner of the screen.

PHPinfo

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



PHP Version 5.2.4

System	OS400 i5dev5r3 3 5 0065000A9D5D
Build Date	Sep 4 2007 17:48:38
Configure Command	'./configure' '--prefix=/usr/local/Zend/Core' '--with-config-file-path=/etc' '--enable-force-cgi-redirect' '--enable-fastcgi' '--disable-debug' '--enable-inline-optimization' '--enable-memory-limit' '--disable-all' '--enable-ctype' '--enable-dom' '--enable-libxml' '--with-libxml-dir=/usr/local/Zend/Core' '--with-openssl=/usr' '--with-pcre-regex' '--enable-session' '--enable-simplexml' '--enable-spl' '--enable-wddx' '--enable-xml' '--with-zlib=/usr' '--with-pear' '--with-apxs2=/usr/local/Zend/apache2/bin/apxs' '--with-layout=GNU' '--enable-zmail' '--enable-json' '--enable-filter' '--enable-hash' '--enable-reflection'
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc
Loaded Configuration File	/usr/local/Zend/core/etc/php.ini
PHP API	20041225
PHP Extension	20060613
Zend Extension	220060519
Debug Build	no
Thread Safety	disabled
Zend Memory Manager	enabled
IPv6 Support	enabled
Registered PHP Streams	php, file, data, http, ftp, compress.zlib, https, ftps
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, sslv3, sslv2, tls
Registered Stream Filters	string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, zlib.*

This program makes use of the Zend Scripting Language Engine:
Zend Engine v2.2.0, Copyright (c) 1998-2007 Zend Technologies
with Zend Core v2.5.0, Copyright (c) 1998-2006, by Zend Technologies
with Zend Extension Manager v1.2.0, Copyright (c) 2003-2007, by Zend Technologies
with Zend Optimizer v3.3.1, Copyright (c) 1998-2007, by Zend Technologies
with Zend Debugger v5.2.10-i5, Copyright (c) 1999-2007, by Zend Technologies

Powered By Zend Engine 2

Figure 15 - Control Center Tab – phpInfo (p.1)

Changing phpInfo

Zend Core's GUI allows easy changing of PHPinfo through the Configuration tab. Any changes made to the extensions and directives in this tab will be automatically updated in your php.ini file and will be reflected in the PHPinfo tab.

Note:
Configuration changes will only take effect once the server has been restarted.

More information about the PHPinfo display can be found in the PHP Manual, accessed by going to the Documentation Tab | PHP and clicking on "PHP Options and Information", Chapter VI., section LV.

Benchmark

Benchmarking is part of the Zend Core environment for building and deploying Web applications. Benchmarks are performance standards for measuring Web Server performance and durability while providing a means for analyzing Web Page performance.

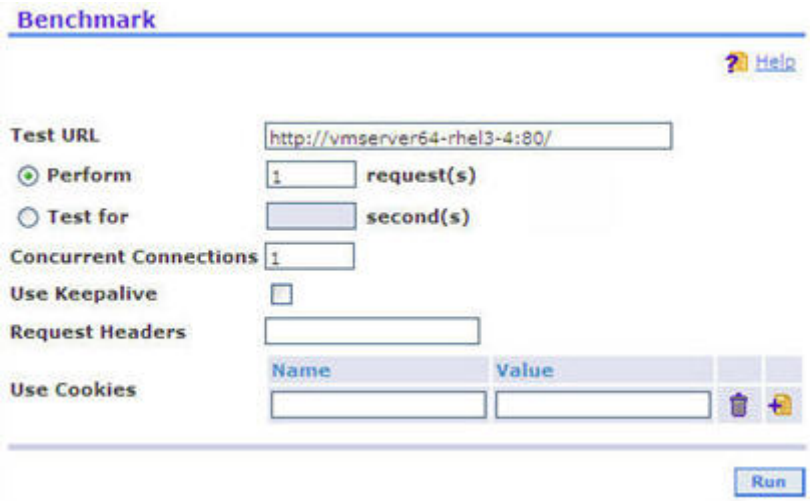


Figure 16 - Control Center Tab – Benchmark

To run a benchmark test on a URL:

1. Specify the complete URL to be tested, including port number.
2. Choose the amount of requests to perform
   -Or- specify the duration (in secs) for which the test will run.
3. Specify the amount of concurrent connections to be simulated in the test.

The limit is 64 concurrent connections, in order to prevent overloading the system.

4. The "Use Keep Alive" option pertains to Web sites that support the HTTP Keep Alive option. Selecting this option keeps the connection open while running the test. This is as opposed to opening and closing connections for every request.
5. Add header lines to the request if necessary.
6. Add a cookie by implementing the following steps:
   i. Add the name of the cookie and its value and press Add Cookie.
   ii. The list will expand for adding additional Cookies to the list.
   iii. Press Delete Cookie to remove a Cookie from the test.
7. To run the Benchmark test, click Run.

Test results are displayed next to the test parameters as follows:

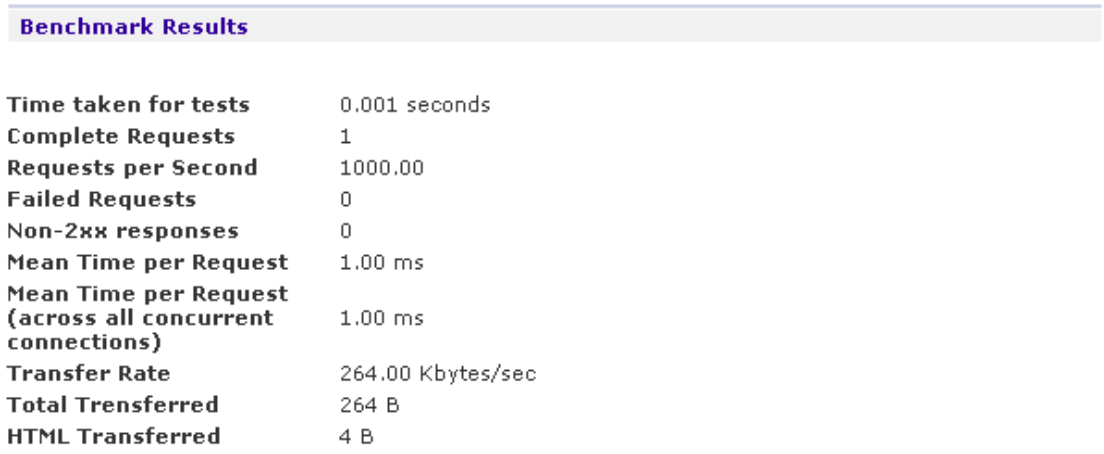


Figure 17 - Benchmark Result Screen

- **Time Taken for Tests** - The duration of the test.
- **Complete Requests** - The number of tests performed.
- **Requests Per Second** - Sum of completed tests divided by the time taken for each request.
- **Failed Requests** - The number of failed tests out of the sum of complete requests.
- **Non-2xx Responses** - The amount of tests that did not get a response containing 2xx from the server (this is a failure indication).
- **Mean Time per Request** - Average time per request.
- **Mean Time per Request** (across all concurrent connections) - Average time per request for all connections.

- **Transfer Rate** - Calculated as the sum of Bytes transferred divided by the time it took to transfer the Bytes.
- **Total Transferred** - The total quantity of Bytes transferred during the test.
- **HTML Transferred** - The amount of HTML code transferred (taken from the Total Transferred).

Support

The Support screen provides instant access to Online Resources. From here, users can get support, provide product feedback and benefit from the PHP community's knowledge and support.

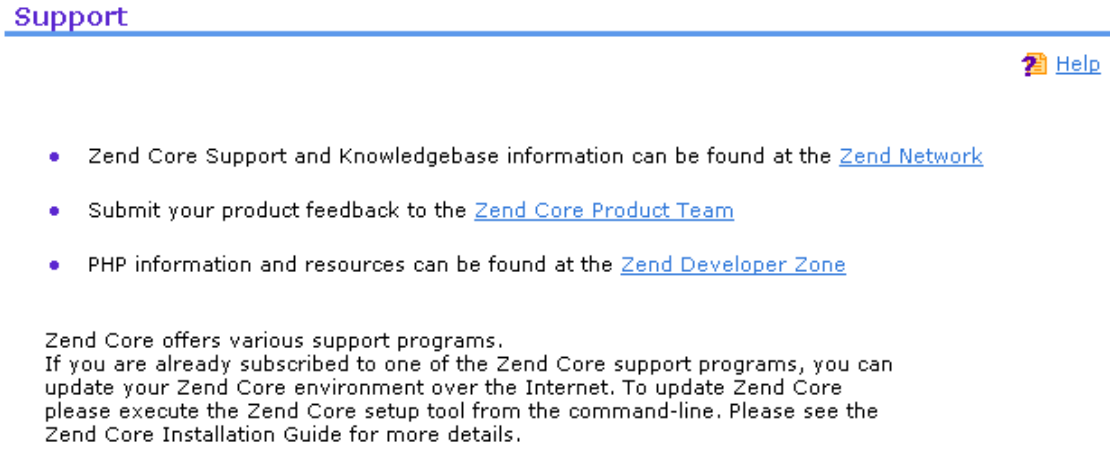


Figure 18 - Support Tab

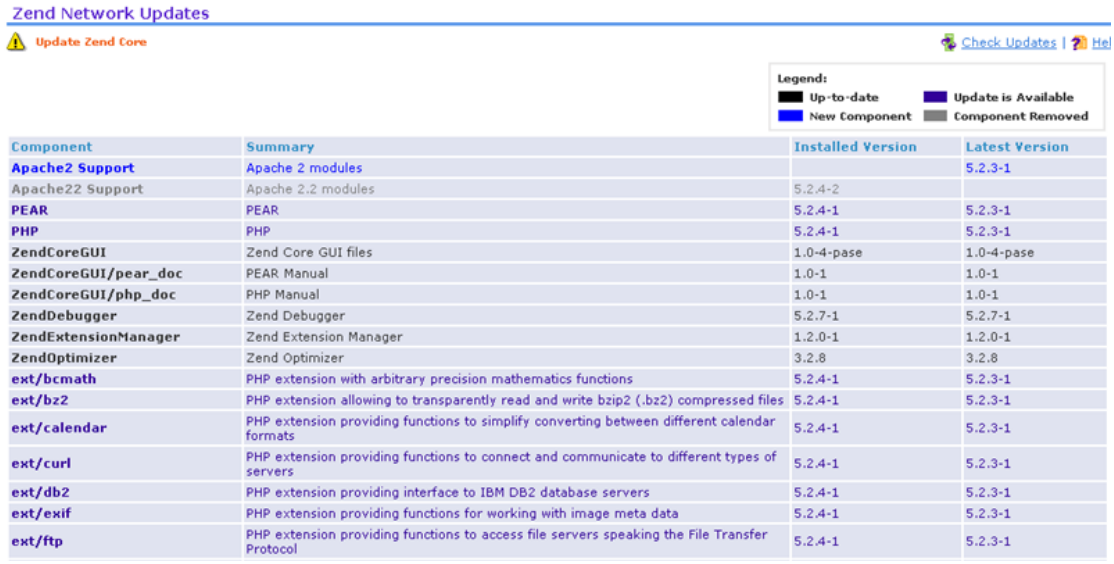
This screen includes links to:

- **Zend Support and Knowledge base** - Provides access to the Zend Core Support Subscription page, through which you can register for a Zend Core Support Subscription in order to have access to the latest security Updates.
  By downloading Zend Core for i5/OS, you have received a one year, first-level Silver Support Subscription.
- **Product feedback** - Allows you to e-mail the Zend team with your comments and suggestions.
- **Zend Developer Zone** - Gives access to the latest PHP information from the Zend Developer Zone.
  This can be accessed from http://www.zend.com/en/developers.php.

Updates

The Updates screen provides a view of available Updates based on your Zend Network registration information.

A color-coded legend indicates the status of each Update. (This will only be available if you have configured your Zend Network login details. See below.)



Component	Summary	Installed Version	Latest Version
Apache2 Support	Apache 2 modules		5.2.3-1
Apache22 Support	Apache 2.2 modules	5.2.4-2	
PEAR	PEAR	5.2.4-1	5.2.3-1
PHP	PHP	5.2.4-1	5.2.3-1
ZendCoreGUI	Zend Core GUI files	1.0-4-pase	1.0-4-pase
ZendCoreGUI/pear_doc	PEAR Manual	1.0-1	1.0-1
ZendCoreGUI/php_doc	PHP Manual	1.0-1	1.0-1
ZendDebugger	Zend Debugger	5.2.7-1	5.2.7-1
ZendExtensionManager	Zend Extension Manager	1.2.0-1	1.2.0-1
ZendOptimizer	Zend Optimizer	3.2.8	3.2.8
ext/bcmath	PHP extension with arbitrary precision mathematics functions	5.2.4-1	5.2.3-1
ext/bz2	PHP extension allowing to transparently read and write bzip2 (.bz2) compressed files	5.2.4-1	5.2.3-1
ext/calendar	PHP extension providing functions to simplify converting between different calendar formats	5.2.4-1	5.2.3-1
ext/curl	PHP extension providing functions to connect and communicate to different types of servers	5.2.4-1	5.2.3-1
ext/db2	PHP extension providing interface to IBM DB2 database servers	5.2.4-1	5.2.3-1
ext/exif	PHP extension providing functions for working with image meta data	5.2.4-1	5.2.3-1
ext/ftp	PHP extension providing functions to access file servers speaking the File Transfer Protocol	5.2.4-1	5.2.3-1

Figure 19 - Control Center Tab – Updates

To refresh the view and check for more Updates from the Zend Network, click the "Check Updates" button 

, located in the top-right corner of the page.

If Updates are available, a message will be displayed at the top of the screen saying 'Update Zend Core.'
Available Updates can be downloaded and installed automatically using the Zend Core Setup.
See the 'Updating Zend Core' chapter for more on how to install Updates.

Note:

You must have purchased a Zend Support Subscription (Silver, Gold or Platinum) in order to have access to Updates.

By downloading Zend Core for i5/OS, you have received a one year, first-level Silver Support Subscription.

If your Zend Core Support Subscription User ID and Password have not been configured in Zend Core, you will receive an error message.

For more on Zend Support Subscriptions, and to register, see the Zend Core Support page at http://www.zend.com/en/products/core/support.

To configure your Zend Core (Network) Support Subscription User ID and Password:

1. Open the Zend Core Setup Tool by running the "go zendcore/zcmenu" command in your i5/OS emulation screen.
2. Select Option 2 - Update via Zend Network menu and then Option 1 - Change Network ID user/password.
3. Enter your Zend Network User ID and Password and press Enter.
4. Restart the web server in order for your changes to take effect.

   To restart your web server:

   In the Zend Core Setup Tool main menu, select Option 5 - Service Management menu and then select Option 6 - Restart Apache server instance.

Configuration

The Zend Core configuration section includes the following tabs:

- PHP - Easily customize your php.ini values.
- Extensions - Control the extensions loaded in your environment.
- Zend Products - Configure the Zend products included with the Zend Core package.
- Misc. Directives - Configure directives that are not part of Zend Core.
- Zend Debugger - Enable PHP code debugging and profiling sessions using the integration between the Zend Debugger and Zend Core.

PHP

The PHP screen is the configuration tool for viewing and customizing the PHP values in the php.ini file.

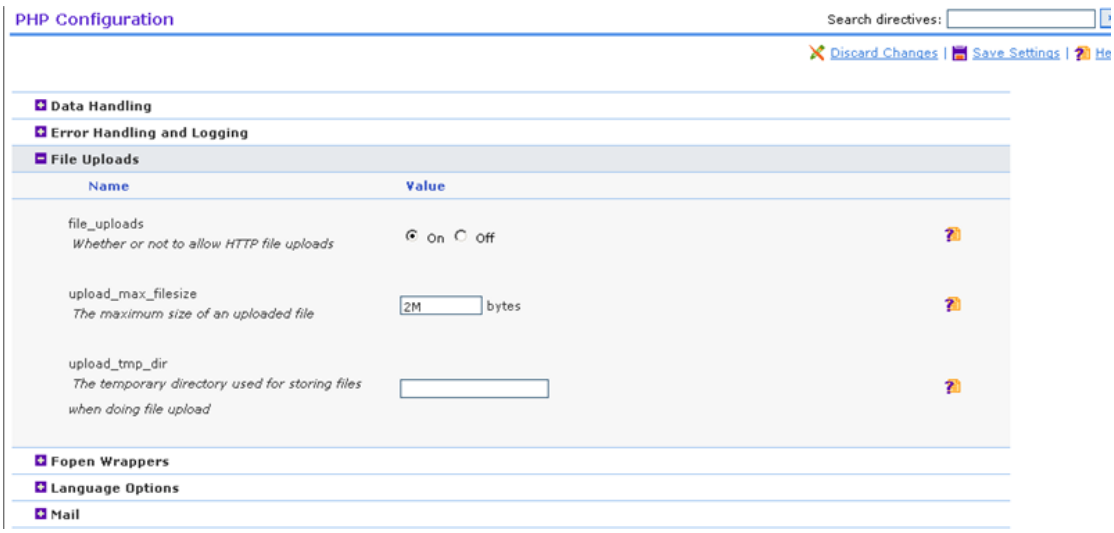


Figure 20 - Configuration Tab PHP Configuration

Configuration options are separated by type in expandable lists. The [+] and [-] signs indicate if there are more options related to that list item.
Clicking on the Plus Icon [+] will expand the list to expose the different options and, where applicable, input fields are added to change an option's value.

When applicable, click the Help icon Help to get more information about the directive.

Note:

The search directives box at the top of the screen allows you to search all the Configuration tabs for a required directive. The result will be displayed in the relevant Configuration tab. If there is more than one result, relevant results will be presented in a drop-down list to the right of the Search directives box. Selecting a directive from the drop-down list will take you to the relevant tab.

To configure a PHP directive:

1. Expand the list or use the search directive box to find the relevant directive.
2. Configure the directive as required.
   You can configure multiple directives before saving.
3. Click the Save Settings  button at the top-right corner of the screen to save all the changes made or use the Discard Changes  button to undo all the changes made since the last save.
4. To apply the changes restart the server.
   To restart your web server:
   In the Zend Core Setup Tool main menu, select Option 5 - Service Management menu and then select Option 6 - Restart Apache server instance.

Changes will be updated in the PHP Configuration screen and will also be made in the php.ini file.

For a full list of available PHP configuration options, see Appendix B - Additional PHP Configuration Information

Extensions

The Zend Core Extensions screen provides a convenient way to view and configure extensions.

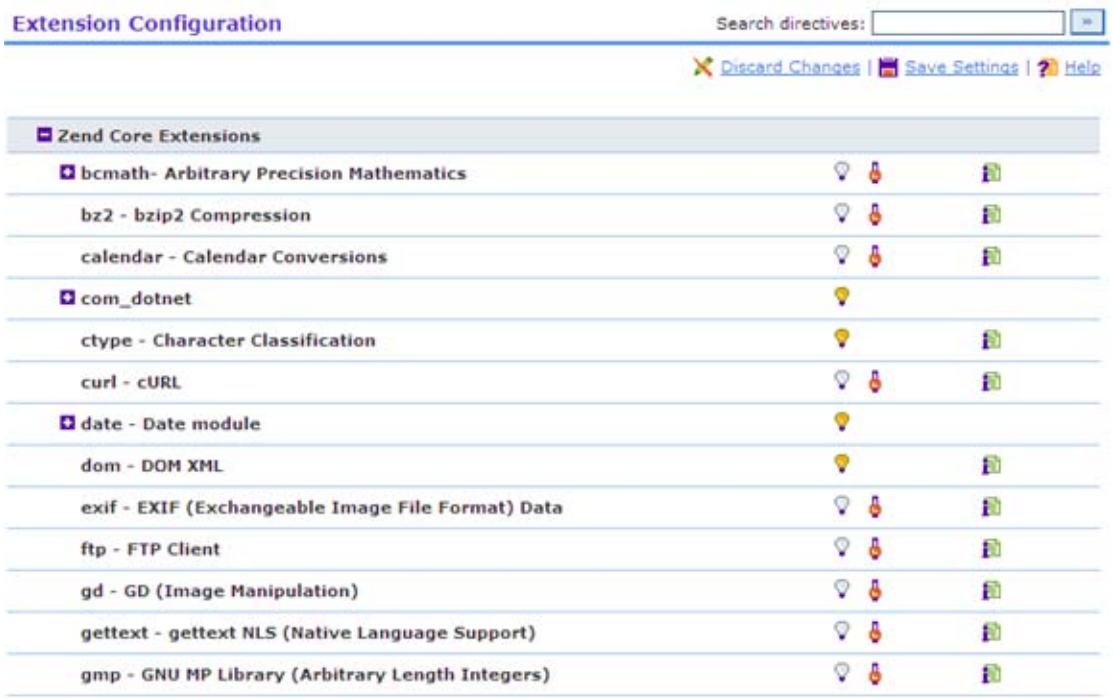


Figure 21 - Configuration Tab Extension Configuration

System Administrators may prefer to control the extensions loaded in their environment to make sure that only necessary extensions are loaded.
A PHP extension is a set of instructions that adds functionality to PHP. Extensions can also be employed to recycle frequently used code. You can place a set of functions into one extension and instruct your projects to utilize the extension. Another use for PHP extensions is to improve efficiency and/or speed. Some processor intensive functions can be better coded as an extension rather than straight PHP code.

Note:
The purpose of the load/unload extension option is to configure php.ini according to the extensions you would like loaded.

The Extensions screen is a configurable list of extensions built in with the Zend Core installation or extensions added to php.ini by the user. It allows you to view the status of all your extensions and enables you to quickly and easily load and unload extensions.

In addition, you can also configure directives associated with certain extensions. Extensions with configurable directives will have a Plus Icon [+] next to them. Click the Plus Icon [+] to expose a list of the different configurable directives associated with a particular extension.

When applicable, click the Reference Icon to the right of an extension to display information about the extension in the PHP manual.

When applicable, click the Help icon Help to view information about a particular directive.

Note:

The search directives box at the top of the screen allows you to search all the Configuration tabs for a required directive. The result will be displayed in the relevant Configuration tab. If there is more than one result, relevant results will be presented in a drop-down list to the right of the Search directives box. Selecting a directive from the drop-down list will take you to the relevant tab.

Extension Status

Extensions can have one of three different statuses:

- Unloaded - The extension is not running on the machine.
- Loaded - The extension is running on the machine.
- Built In - Built-in extensions are extensions that have dependencies, or were complied with PHP. Built in extensions cannot be removed and so do not have an enable/disable icon next to them.

Hovering over the lightbulb icon will display a tooltip indicating whether the status is unloaded, loaded or built in.

Note:

Extensions marked with an '!' indicate that an inconsistency occurred between the server state and the php.ini state. Possible causes are that the php.ini was changed earlier and the server was not restarted, or that the extension failed to load. To test this, try to restart the server.
Extensions and directives marked '*' have different values (or loaded/unloaded states in case of Extensions) in the php.ini file and in the running server instance. To synchronize their state/value, restart the Web Server.

To change an extension's status:

1. Click the Enable or Disable Extension Switch next to the required extension.
   Built-in extensions cannot be disabled and so will not have an Extension Switch displayed.
   A notice will appear to restart the server.
2. Click the Save Settings button Save Settings at the top right-corner of the screen to save the changes or click the Discard Changes Discard Changes button to undo all changes made
3. To apply the changes restart the server.
   To restart your web server:
   In the Zend Core Setup Tool main menu, select Option 5 - Service Management menu and then select Option 6 - Restart Apache server instance.

Changes will be updated in the Extension Configuration screen and will also be made in the php.ini file.

To configure a directive associated with an extension:

1. Expand the list or use the search directive box to find the relevant directive.
2. Configure the directive as required.
   You can configure multiple directives before saving.
3. Click the Save Settings Save Settings button at the top right corner of the screen to save all the changes made or use the Discard Changes Discard Changes button to undo all the changes made since the last save.
4. To apply the changes restart the server.
   To restart your web server:
   In the Zend Core Setup Tool main menu, select Option 5 - Service Management menu and then select Option 6 - Restart Apache server instance.

Changes will be updated in the Extension Configuration screen and will also be made in the php.ini file.

Note:

Directives of both loaded and unloaded extensions can be configured through the Extension configuration screen.

For a full list of extensions and their descriptions, see Appendix C - Zend Core Extensions.

Note:

Some extensions have dependencies on certain libraries.

For a full list of libraries installed with Zend Core, see Appendix D - Libraries.

System i users should only download extensions compiled in AIX (up to version 5.2).

To Add Zend Extensions:

1. Download the extension.
2. Place the extension in your extensions directory.
   To locate the extensions directory, open your php.ini and check the value for the directive extension_dir=.
   By default, your extensions directory will be located in:
   /usr/local/Zend/Core/lib/php/20060613
3. Add the following line to your php.ini:
   zend_extension_manager.<my_extension_name>=
   <full_path_to_extension_location>/<my_extension_name>
4. Ensure that you have replaced <full_path_to_extension_location> with the path to your extension's location and <my_extension_name> with your extension's name.
5. Restart your web server.
   To restart your web server:
   In the Zend Core Setup Tool main menu, select Option 5 - Service Management menu and then select Option 6 - Restart Apache server instance.
6. Ensure that the extension is properly loaded by checking the output of PHPInfo by going to the Control Center | PHPinfo tab in Zend Core.

Note:

If you try to load a PHP extension as a Zend extension you will receive the following error message in your server's error log: "<extension_name> doesn't appear to be a valid Zend extension."
In this case, remove it and add it as a PHP extension following the instructions under "To Add PHP Extensions", below.

To Add PHP Extensions

1. Download the third party extension. Many third party extensions can be found through at http://pecl.php.net.
   Extensions are obtained directly from external web repositories.
2. Place the PHP extension in your extensions directory.
   To locate the extensions directory, open your php.ini and check the value for the directive extension_dir=.
   By default, your extensions directory will be located in:
   /usr/local/Zend/Core/lib/php/20060613
3. Add the following line to your php.ini:
   extension=<my_extension_name>.so
   Ensure that you replace <my_extension_name> with your extension's name.
4. Restart your web server.
   To restart your web server:
   In the Zend Core Setup Tool main menu, select Option 5 - Service Management menu and then select Option 6 - Restart Apache server instance.
5. Ensure that the extension is properly loaded by checking the output of PHPInfo by going to the Control Center | PHPinfo tab in Zend Core.

Compiling extensions

You can also create and compile your own extensions using the phpize command.

Disclaimer:

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

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

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

To compile extensions from source:

1. Download and extract the extension's source.
2. Change into the extension source directory, (by default located in /usr/local/Zend/Core/lib/phpext) and run the following commands:

```
cd <your_extension_directory>
/usr/local/Zend/Core/bin/phpize
```

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

```
./configure --with-php-config=/usr/local/Zend/Core/bin/php-config\
            --enable-shared
```

   Note:
   Some extensions will need additional configuration flags. It is therefore advised to run "./configure --help" and review the possible flags before compiling.

4. Compile and install the extension binaries by running the following commands:

```
make
make install
```

   Make install should install the new .so extension binary in Zend Core's extension directory.
5. Add the following line to your php.ini to load your new extension:

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

   Replace <my_extension_name> with your extension's binary name.
6. Restart your web server.
7. Ensure that the extension is properly loaded by checking the output of PHPInfo. This can be viewed in the Control Center | PHPinfo tab in Zend Core.

The extension will now appear in your Zend Core Web Administration GUI under the Extensions tab and you will be able to use Zend Core Web GUI to load and unload the extension.

Zend Products

The Zend Products tab allows you to view and configure the Zend products included with the Zend Core package.
Through this tab you can view the status (loaded/unloaded) of your Zend Core products, and configure certain directives associated with them.

Zend products are listed by type in expandable lists. Clicking on the Plus Icon [+], where applicable, will expand the list to expose the configurable directives associated with a Zend Product. Relevant input fields are added to change a directive's value.

Note:
The search directives box at the top of the screen allows you to search all the Configuration tabs for a required directive. The result will be displayed in the relevant Configuration tab. If there is more than one result, relevant results will be presented in a drop-down list to the right of the Search directives box. Selecting a directive from the drop-down list will take you to the relevant tab.

Through this screen, users can configure the following Zend products:

- Zend Core
- Zend Debugger
- Zend Extension Manager
- Zend Optimizer
- Additional Zend Product Directives

Note:
More extensions may be listed if additional Zend Products are installed on the machine.

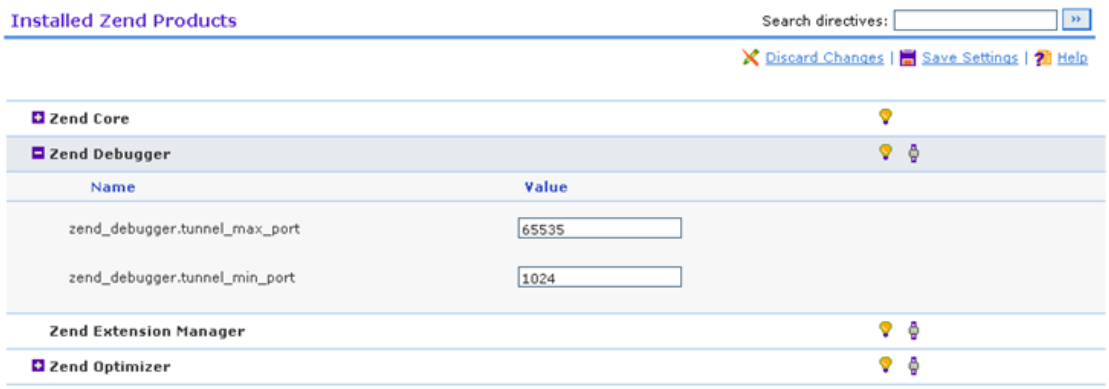


Figure 22 - Configuration Tab - Installed Zend Products List

Zend Core

zend_core.allow_restart	Enables restarting the server from the Zend Core restart button. This option is disabled under on Zend Core for i5/OS.
zend_core.default_gui_language	Determines the user interface language (supported languages will vary according to product version).

Zend Debugger

The Zend Debugger is the client extension for debugging PHP with Zend Studio IDE. This extension provides the initial framework needed for initiating debug sessions.

Using the Zend Debugger with Zend Studio IDE provides advanced debugging features, including Conditional Breakpoints, Stack Trace View, Advanced Watches, Variables and Output Buffer.

The Studio Client does not have to be installed on the Web Server and can be used for debugging with a remote Client over SSL. Remote connections are secure, ensuring maximum protection for remote debugging with offsite locations or across the Internet.

For more on debugging using Zend Studio IDE, see the Zend Studio User Guide. For more information on Zend Studio and to download the product or the User Guide, go to:

http://www.zend.com/products/zend_studio/i5OS

Note:
The settings in the Zend Debugger tab are only applicable when the Debugger extension is loaded.
See the 'Zend Debugger' topic for more information.

The following zend.ini directives define a port range for Tunneling. You can modify these settings to ensure persistent connections while using Tunneling over firewalls for debugging event information in Zend Platform or debugging scripts edited in Zend Studio IDE:

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

While Tunneling, the Debugger will try to locate a free port in the range defined in the max and min Zend Debugger Tunnel Port directives above. Another consideration when defining a port range is to ensure that 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 IDEs that will be using the Debugger Port.

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.

In parallel, the System Administrator must ensure that proper firewall policies are set to allow communication via the selected ports.
The tunnel server, and not the debugger, uses these tunnel settings. The debugger will still use random ports for debugging.

Note:

The following error message might appear in your web server's error log:
"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.

For more on configuring debugging settings using Zend Core, see information under the Zend Debugger tab topic.

Zend Extension Manager

The Zend Extension Manager is in charge of loading the Zend modules according to their appropriate versions.

Zend Optimizer

Zend Optimizer is a free application that runs the PHP scripts encoded by Zend Guard for enhancing PHP application running speed.
In addition, Zend Optimizer goes over the intermediate code generated by the standard Zend run-time compiler and optimizes it for faster execution.
Read more about the Zend Optimizer at
http://www.zend.com/en/products/guard/optimizer.

Zend Optimizer Directives:

zend_optimizer.disable_licensing	You can disable the Zend Optimizer license request if you do not need to use any licensing features.
zend_optimizer.enable_loader	Adding the zend_optimizer.enable_loader = 0 directive will slightly improve the Optimizer's performance as it disables the transparent auto-loading mechanism that is built into the Zend Optimizer. Only disable this directive if you do not plan to use the Zend Optimizer to load encoded files.
zend_optimizer.licence_path	A license file is required to load encoded PHP scripts that were encoded with a require-license option. If you turn off this option, encoded scripts on your server that require a license may not load.
zend_optimizer.optimization_level	Enable optimizations bitmask.

Zend Product Status

Zend Products can have one of three statuses according to different requirements and the environment running PHP.

The statuses are as follows:

- Unloaded - The product is not running on the machine.
- Loaded - The product is running on the machine.
- Built In - Built-in products are products that have dependencies, or were complied with PHP. Built in extensions cannot be removed and so do not have an enable/disable icon next to them.

Note:

Zend Products cannot be disabled through the Zend Products tab. To disable Zend Products, go to your php.ini file and comment out the required extension.

To configure a directive associated with a Zend Product:

1. Expand the list or use the search directive box to find the relevant directive.
2. Configure the directive as required.
   You can configure multiple directives before saving.
3. Click the Save Settings  button at the top right corner of the screen to save all the changes made or use the Discard Changes  button to undo all the changes made since the last save.
4. To apply the changes restart the server.
   To restart your web server:
   In the Zend Core Setup Tool main menu, select Option 5 - Service Management menu and then select Option 6 - Restart Apache server instance.

Changes will be updated in the Zend Products screen and will also be made in the php.ini file.

Additional information about Zend Products can be found at http://www.zend.com/products, or by going to the Additional Zend Products and Services chapter.

Misc. Directives

Directives that are not part of Zend Core are listed in the Misc. Directives screen. This screen allows you to easily view and configure additional commonly used directives.

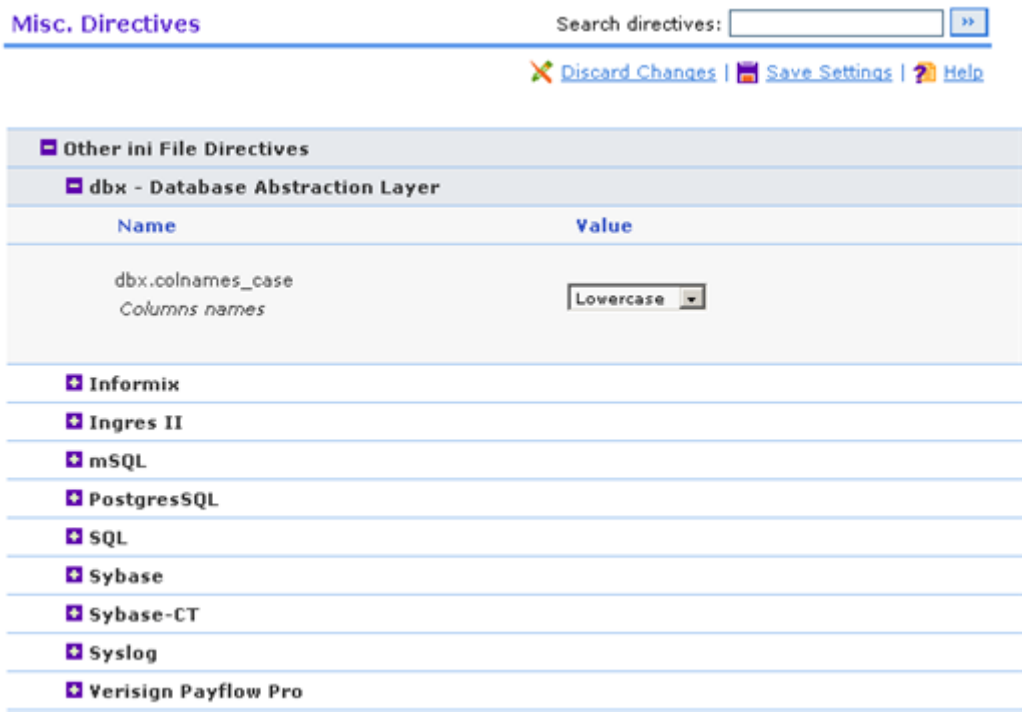


Figure 23 - Configuration Tab - Misc.Directives

The available directives are separated by type in expandable lists. Clicking on the Plus Icon [+] will expand the list to expose the different options and, where applicable, input fields are added to change a directive's value.

When applicable, click the Help icon Help to get more information about the directive.

Note:

The search directives box at the top of the screen allows you to search all the Configuration tabs for a required directive. The result will be displayed in the relevant Configuration tab. If there is more than one result, relevant results will be presented in a drop-down list to the right of the Search directives box. Selecting a directive from the drop-down list will take you to the relevant tab.

For a full list of Misc. Directives configuration information, see Appendix E - Misc. Directives Configuration Information.

To configure Misc. directives:

1. Expand the list or use the search directive box to find the relevant directive.
2. Configure the directive as required.
   You can configure multiple directives before saving.
3. Click the Save Settings  button at the top right corner of the screen to save all the changes made or use the Discard Changes  button to undo all the changes made since the last save.
4. To apply the changes restart the server.
   To restart your web server:
   In the Zend Core Setup Tool main menu, select Option 5 - Service Management menu and then select Option 6 - Restart Apache server instance.

Changes will be updated in the Misc. Directives screen and will also be made in the php.ini file.

Zend Debugger

Zend Core comes complete with the option to enable remote PHP debugging and profiling of Web applications through using the Zend Debugger (Zend Studio Server) component.
This component enables developers using the Zend IDE to connect to a remote server in order to analyze (debug and profile) and fix code.

The Zend Debugger tab allows you to configure which hosts should be allowed to initiate debugging and profiling sessions.

Note:
For the Zend Studio IDE to be able to initiate debugging and profiling sessions, the IP address of the machine where the Zend Studio IDE is installed must be in the Allowed Hosts list.

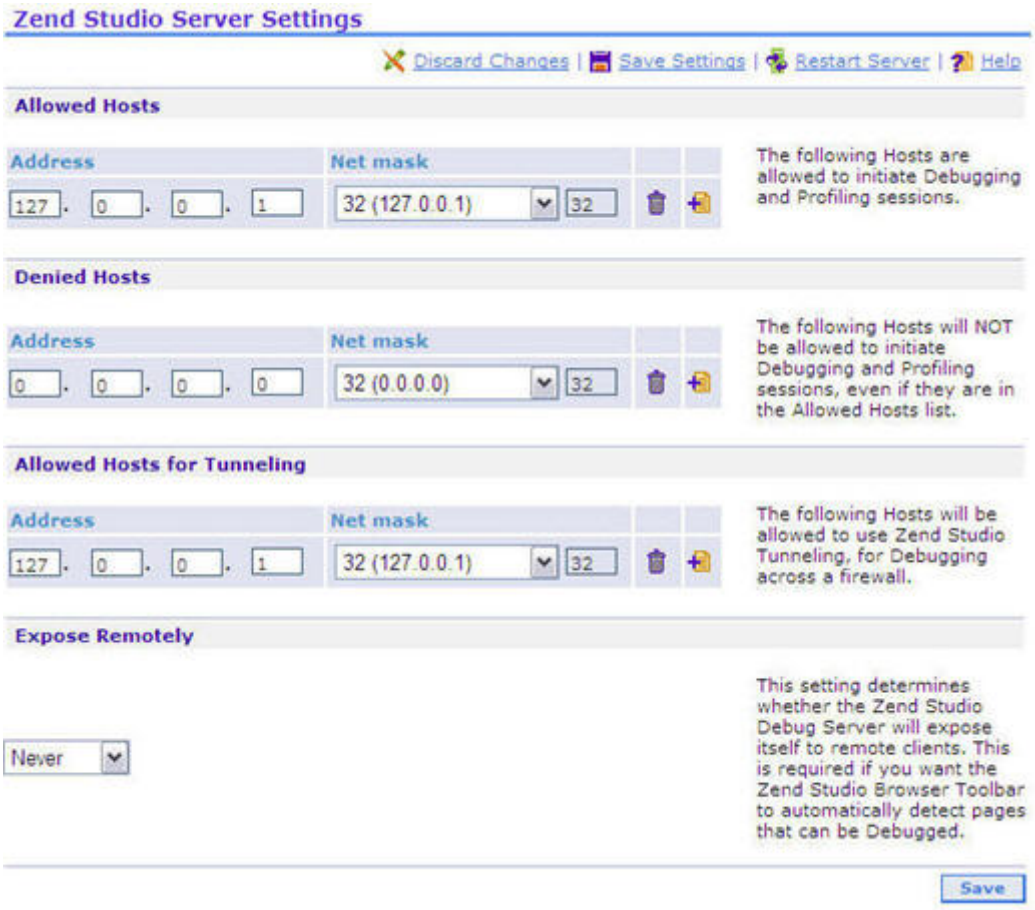


Figure 24 - Configuration Tab - Zend Debuger

Settings

The Zend Debugger tab displays the settings for the Debugger that resides on the server.

There are four categories of settings:

1. **Allowed Hosts** - Lists the hosts allowed to initiate Debugging and Profiling sessions.
2. **Denied Hosts** - Lists the hosts that are not allowed to initiate Debugging and Profiling sessions, even if they are on the Allowed Hosts list.
3. **Allowed Hosts for Tunneling** - Lists the hosts allowed to use the Zend Studio Tunnel for debugging across a firewall.
4. **Expose Remotely** - 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.

To add/remove an Allowed, Denied or Tunneling Host:

1. Click "Add"  .
   A new unconfigured line will be added to the Host list.
2. Enter the required address and Net Mask.
3. To remove a Host, click the "Remove Host button"  next to the required host.
4. Press Save  .
5. To apply the changes restart the server.
   To restart your web server:
   In the Zend Core Setup Tool main menu, select Option 5 - Service Management menu and then select Option 6 - Restart Apache server instance.

To configure your "Expose Remotely" settings:

1. Choose the required option from the drop-down list:
   - Always - Will expose all hosts
   - Selective - Only exposes the hosts in the allowed host list
   - Never - Will not expose any host
2. Press Save  .
   To apply the changes restart the server.
   To restart your web server:
   In the Zend Core Setup Tool main menu, select Option 5 - Service Management menu and then select Option 6 - Restart Apache server instance.

Note:

The settings in the Zend Debugger Tab are only applicable when the Debugger extension is loaded. See the Zend Products tab for more on the Debugger extension.

Documentation

The Documentation tab is the main source for reference information.
Using the Search sub-menu considerably reduces the time it takes to obtain information as all necessary information is locally available, with no need to search the internet.

The tabs included under the Documentation tab are:

- PHP - The PHP Manual
- PEAR - The PEAR Manual
- Search - Allows you to search the PHP/PEAR Manuals

PHP

The PHP screen of the Documentation Tab allows easy access to the PHP Manual for instant access to the most comprehensive PHP reference information.
The PHP Manual consists primarily of function references, but also contains language references, explanations of some of PHP's major features, and other supplemental information.

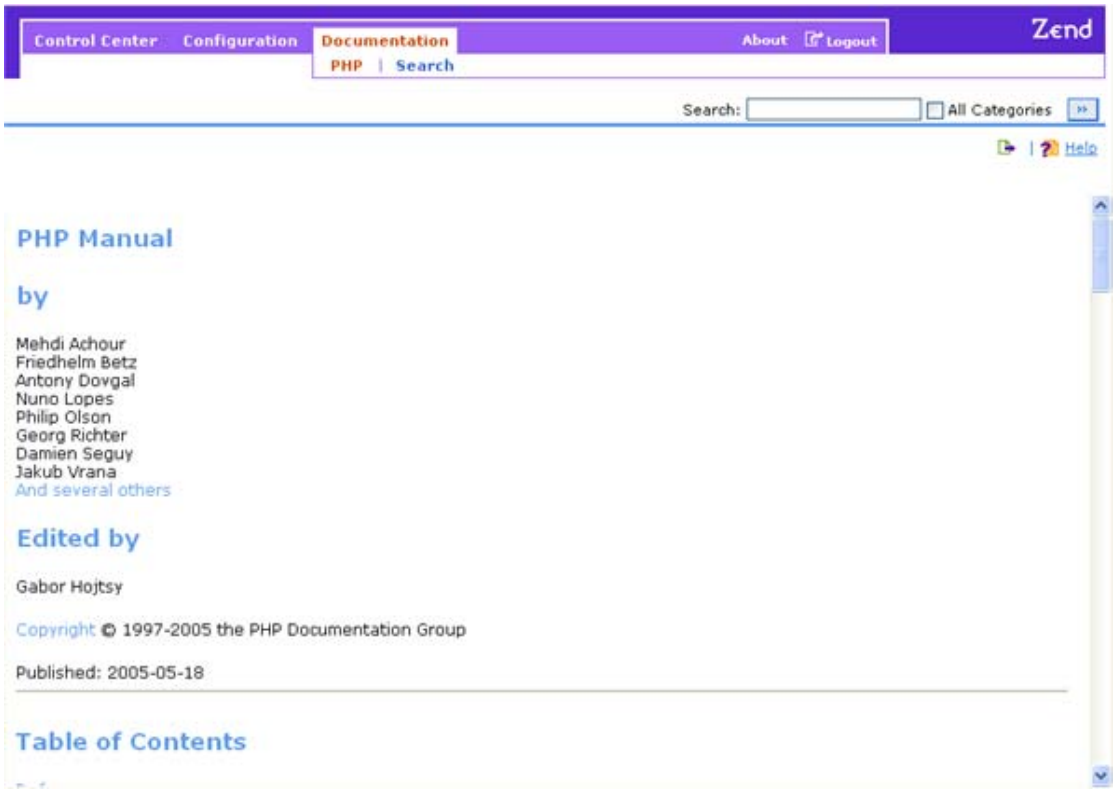


Figure 25 - PHP Manual Tab

Use the Table of Contents to browse to the required section of the PHP Manual.

Use the browse buttons at the top of the page to go to the next / previous page, to go up a level or to return to the PHP Manual Homepage .

The Search box Search: at the top of the page allows you to search the PHP Manual for specific information.
For more advanced search options, see the Search tab.

Note:
For the most up to date php information, refer to the online PHP manual found at http://php.net.

PEAR

The PEAR screen of the Documentation Tab allows easy access to the PEAR Manual for instant access to the most comprehensive PEAR reference information.

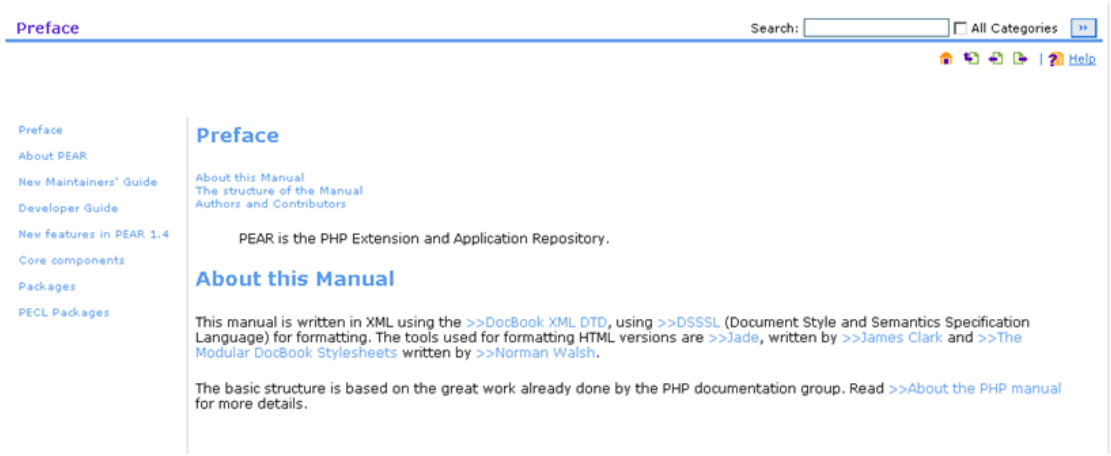


Figure 26 - PEAR Manual Tab

Use the Table of Contents on the right to browse to the required section of the PHP Manual.
Use the browse buttons at the top of the page to go to the next / previous page, to go up a level or to return to the PHP Manual Homepage .

The Search box Search: at the top of the page allows you to search the PHP Manual for specific information.
For more advanced search options, see the Search tab.

Search

The Search Documentation tab allows you to search the PHP and PEAR Manuals to get the most relevant information.

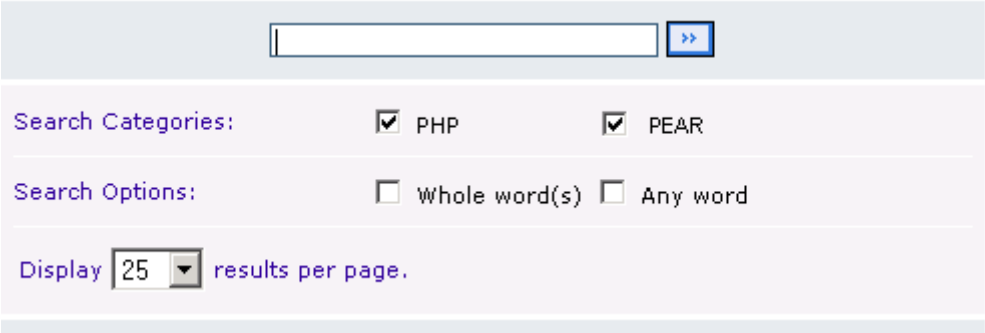


Figure 27 - Search Tab

Enter a string and click the arrow icon  to search for it in either the PHP or PEAR Manuals.

Note:

Unmark the PHP/PEAR checkbox to limit your search to a specific manual.

In addition to the PHP and PEAR Manuals Search functionality, the PHP, Extensions, Zend Products and Misc. Directives pages under the Configuration tab also have a 'Search Directives' box which allows you to search all the Configuration tabs for a required directive. The result will be displayed in the relevant Configuration tab. If there is more than one result, relevant results will be presented in a drop-down list to the right of the Search directives box. Selecting a directive from the drop-down list will take you to the relevant tab.

i5 PHP API Toolkit

i5 PHP API Toolkit

The purpose of the PHP ToolKit is to allow Zend Core for i5/OS to interact with native i5/OS services.

The PHP APIs enable PHP programs to access System objects such as RPG/COBOL/Java programs, CL commands, Data Queue, Spooled file, etc. These APIs expose the PHP Object Oriented programming interface.

From an architectural standpoint, PHP functionality is implemented as a PHP extension that is enabled during the Zend Core for i5/OS installation.

The extension implements the client side of the interface.

A server, implementing the native i5/OS interface, is installed on the i5/OS machine as a native i5/OS service.

i5 Toolkit Classes (sample)

i5 Toolkit classes can be found in the directory /www/zendcore/i5Toolkit_library. This directory contains the i5 Toolkit classes file "Toolkit_classes.php", and the "demo_for_toolkit_classes.php" sample program, which utilizes the Toolkit classes.

The i5 Toolkit class library contains the following classes:

- i5_Connection - Connection class
- i5_Description - Data type definition class
- i5_Program - Program call class
- i5_DataQueue - Data Queue class
- i5_DataQueueKey - Keyed Data Queue class
- i5_SpoolList - Spooled file list class
- i5_Userspace_Create - User Space create class
- i5_Userspace_Delete - User Space delete class
- i5_UserspaceManage - User Space read/write class
- i5_DataAreas - Data Area class
- i5_JobLogs - Job log class
- i5_ActiveJobs - Active Jobs class
- i5_ObjectListing - Object list class
- i5_NativeFileAccess - Database class

INSTALLATION

No special installation is required, just place the i5 Toolkit class file "Toolkit_classes.php" in the same directory as your PHP program.

Zend Studio IDE templates

Zend Studio IDE for i5 comes complete with code templates containing i5 PHP API Toolkit functions.
For a full list of these functions, see Appendix F - I5 Toolkit Templates.

i5 PHP Connector API

All API calls start with prefix "i5".

Connection Management

i5_connect

resource i5_connect(string server, string user, string password[, array options]).

- **Description**: Connects to the AS/400 server.
- **Return Values**: AS/400 connection resource or false on failure.
- **Arguments**:
  - server - Name of the server to connect to. This can be either a symbolic name or an IP
  - user - Username to use for connecting.
    Note: Username QSECOFR cannot be used in this function.
  - password - Password for the username
  - options - Connection options.
- **Example**:

```
$conn = i5_connect("1.2.3.4", "MYUSER", "MYPWD");
if (!$conn) {
  die(i5_errormsg());
}
```

Connection Options:

- I5_OPTIONS_JOBNAME – job name (machine name by default)
- I5_OPTIONS_SQLNAMING – Enables using dotted (.) or slashed (/) notation in SQL requests
- I5_OPTIONS_DECIMALPOINT – Enables using dot or comma as decimal separator
- I5_OPTIONS_CODEPAGEFILE – Enables using specific code page (CCSID)
- I5_OPTIONS_ALIAS – Enables naming a connection. If the name is used in another i5_connect, then the another i5_connect will use the same connection.
- I5_OPTIONS_INITLIBL – Specified libraries are added to the beginning of the initial library list.

i5_close

bool i5_close([resource connection]).

- **Description**: Closes connection to AS/400 server.
- **Return Values**: Boolean success value.
- **Arguments**:
  - connection - Result of i5_connect

i5_adopt_authority

bool i5_adopt_authority(string username, string password, [resource connection]).

- **Description**: Changes authority of the connection to a specific user. All actions will be executed as this user from now on.
- **Return Values**: Boolean success value.
- **Arguments:**
  - username - Name of the user to change to
  - password - Password for the user
  - connection - Connection - result of i5_connect

i5_error

bool i5_error([resource connection]).

- **Description:** Retrieves error information for last action that was executed.
- **Return Values:** If there was no error, returns false. Otherwise, returns an array with the following elements:

  0 - error number, as in **i5_errno().**
  1 - error category.
  2 - error message, as in **i5_errmsg().**
  3 - detailed description of the error.

- **Arguments:**
  - connection - Connection - result of i5_connect

i5_errormsg

string i5_errormsg([resource connection]).

- **Description:** Gets error message for last executed action.
- **Return Values:** Error message string.
- **Arguments:**

- connection - Connection - result of **i5_connect**.

CL Calls

i5_command

bool i5_command(string command[, array inputs, array outputs, resource connection]).

- **Description:** Calls CL command.
- **Return Values:** Boolean success value.
- **Arguments:**
  - inputs - Array of name => value parts, name describing the call input parameters.
    Names should match i5 cl command parameter names.
    If the array is empty or not provided, no input parameters are given. If the value is integer, integer is passed, if the value is string, quoted string is passed. If the value is an array, the list of contained values is

passed.
Note: The output parameter is required if the input parameter is specified.

- outputs - Array which describes output parameters of the command. If not provided, no output parameters are defined.
  Key of the array defined i5 cl command parameter name
  "rc" is a predefined name containing the result of the command.
  Value can be string. If so - it defines a php variable name to accept the parameter or array; it should have 2 elements:
- A php variable name to accept the parameter.
- Description of the parameter

  Note: The input parameter is required if the output parameter is specified.
- connection - Connection - result of i5_connect.

- **Example**:

```
i5_command("rtvjoba", array(), array("curlib" => "curl",.
"user"=>"user",.
"usrlibl" => "userlib",.
"syslibl" => array("syslib", "char(165)"), .
).
);.
print "User : $user<br>" ;.
print "User library : $userlib<br>" ;.
print "System libs list : $syslib<br>" ;.
print "Current library : $curl<br>" ;.
```

Program Calls

i5_program_prepare

resource i5_program_prepare(string name[, array description][, resource connection]).

- **Description:** Opens a program and prepares it to be run.
- **Return Values:** Resource if open succeeded, false if open failed.
- **Arguments:**
  - name - Program name. If a service procedure call is made done, the procedure name is given in parentheses, e.g.,
    Lib/Service_Program(PROC)

- description - PHP-format program description. This should be provided if the program is not described on server.
  See PHP Data Description, page 58.
- connection - Result of i5_connect

i5_program_prepare_PCML

resource i5_program_prepare_PCML (array description[, resource connection]).

- **Description:** Opens a program PCML file and prepares it to be run.
- **Return Values**: Resource if open succeeded, false if open failed.
- **Arguments:**
  - description - PCML file's program and parameters information
  - connection - Result of i5_connect

The program information file (in PCML format) can be created by compiling the RPG program.

- **Example**:

  CRTBNDRPG PGM(EACDEMO/TESTSTRUC)
  SRCFILE(EACDEMO/QRPGLESRC) SRCMBR(TESTSTRUC) PGMINFO(*PCML)
  INFOSTMF('/www/zendcore/htdocs/teststruc.pcml')

  The PCML file will contain the program parameters info. There are two ways you can assign the program parameters to i5-program_prepare_PCML description:

  i. Copy the content of PCML file to you PHP script and assign the i5_program_prepare_PCML description array to the PCML content. See PCML Example 1
  ii. Assign i5_program_prepare description array to the PCML file located in the same PHP program directory. See PCML Example 2.

- **PCML Example 1:**

```
$description = "<pcml version=\"4.0\">
   <!-- RPG module: TESTSTRUC  -->
   <!-- created: 2006-10-12-11.46.56 -->
   <!-- source: EACDEMO/QRPGLESRC(TESTSTRUC) -->
   <!-- 5 -->
   <struct name=\"S2\">
      <data name=\"ZOND2\" type=\"zoned\" length=\"10\"
precision=\"5\" usage=\"inherit\" />
      <data name=\"PACK2\" type=\"packed\" length=\"19\"
precision=\"5\" usage=\"inherit\" />
      <data name=\"PACK3\" type=\"packed\" length=\"19\"
precision=\"5\" usage=\"inherit\" />
      <data name=\"ALPH2\" type=\"char\" length=\"20\"
usage=\"inherit\" />
   </struct>
   <!-- 1 -->
   <struct name=\"S1\">
      <data name=\"ZOND\" type=\"zoned\" length=\"10\"
precision=\"5\" usage=\"inherit\" />
      <data name=\"PACK1\" type=\"packed\" length=\"19\"
precision=\"5\" usage=\"inherit\" />
      <data name=\"ALPH1\" type=\"char\" length=\"10\"
usage=\"inherit\" />
   </struct>
   <program name=\"TESTSTRUC\"
path=\"/QSYS.LIB/EACDEMO.LIB/TESTSTRUC.PGM\">
      <data name=\"CODE\" type=\"char\" length=\"10\"
usage=\"output\" />
      <data name=\"S1\" type=\"struct\" struct=\"S1\"
usage=\"inputoutput\" />
      <data name=\"S2\" type=\"struct\" struct=\"S2\"
usage=\"inputoutput\" />
      <data name=\"PACK\" type=\"packed\" length=\"1\"
precision=\"1\"
      usage=\"output\" />
      <data name=\"CH10\" type=\"char\" length=\"19\"
usage=\"output\" />
      <data name=\"CH11\" type=\"char\" length=\"20\"
usage=\"output\" />
```

```
      <data name=\"CH12\" type=\"char\" length=\"29\"
usage=\"output\" />
      <data name=\"CH13\" type=\"char\" length=\"33\"
usage=\"output\" />
   </program>
</pcml>
   ";
```

- **PCML Example 2:**

```
($description =
file_get_contents("/www/zendcore/htdocs/teststruc.pcml"))
or trigger_error("Error while opening PCML file", E_USER_ERROR);
```

i5_program_call

bool i5_program_call(resource program, array params[, array retvals]).

- **Description:** Calls the program and optionally accepts results.
- **Return Values:** Boolean success value.
- **Arguments:**
  - program - Program resource opened by i5_program_prepare.
  - params - Parameters according to description.
    Can be given as flat array, then parameters are assigned in order, or as key => value pairs then the values are assigned to the parameter named by the key
  - retvals - Array of key => value pairs where keys describe output parameter name and values name PHP variable that would receive the parameter

Fetch should still work even if the return parameters are defined and assigned.

- **Example**:

```
$prog = i5_program_prepare("DEMOPGM");.
if(i5_program_call($prog, array(1,2,"abc"))) {.
$result = i5_fetch_assoc($prog);.
print "result is $result['retval']<br>";.
} else {.
print "Program call failed.<br>";.
}.
```

Note:
Use i5_COMMAND in order to invoke a program without parameters. For example, i5_command("call LIB_NAME/PROGRAM_NAME").

i5_program_close

void i5_program_close(resource program).

- **Description:** Frees program resource handle.
- **Return Values:** Boolean success value.
- **Arguments:**
  - program - Program resource opened by program_open.

Data Retrieval

i5_fetch_array

array i5_fetch_array(resource result [, int option]).
array i5_fetch_assoc(resource result [, int option]).
object i5_fetch_object(resource result [, int option]).
array i5_fetch_row(resource result [, int option]).

- **Description:** Fetches a row of data from the resource.
- **Return Values**: According to the specific fetch function used, it returns either an array or an object containing the data:
  - array - by index and name.
  - assoc - by name.
  - row - by index.
  - object - by name as object properties.
- **Arguments:**
  - result - Resource resulting from operation returning data
  - option - Flag specifying which record to fetch.
    Current record - I5_READ_SEEK
    Next record - I5_READ_NEXT
    Previous record - I5_READ_PREV
    First record - I5_READ_FIRST
    Last record - I5_READ_LAST
    Default is **I5_READ_NEXT**

i5_info

array i5_info (resource result [, int/string field]).

- **Description:** Gets information about the file/record.

- **Return Values:** An array with information about record. If there is no way to return whole information; false is returned when the field parameter is omitted.
- **Arguments:**
  - result - Resource describing file or other record set
  - field - Integer or string identifying the field. If this parameter is omitted, whole file information is given (when possible).

i5_field_len

int i5_field_len (resource result, int/string field).

- **Description:** Gets field length.
- **Return Values:** field's length.
- **Arguments:**
  - result - Resource describing file or other record set
  - field - Integer or string identifying the field position or name.

i5_field_name

int i5_field_name (resource result, int field).

- **Description:** Get field name.
- **Return Values:** field's length.
- **Arguments:**
  - result - Resource describing file or other record set
  - field - Integer identifying the field position.

i5_field_scale

int i5_field_scale (resource result, int/string field).

- **Description:** Gets field scale - number of digits for numeric fields.
- **Return Values:** The number of digits of the field. If the field is not numeric, returns –1.
- **Arguments:**
  - result - Resource describing file or other record set
  - field - Integer or string identifying the field position or name.

i5_field_type

string i5_field_type (resource result , int/string field).

- **Description:** Gets field type.

- **Return Values:** Field's type string.
- **Arguments:**
  - result - Resource describing file or other record set .
  - field - Integer or string identifying the field position or name.

i5_list_fields

array i5_list_fields (resource result).

- **Description:** Gets list of fields for resource.
- **Return Values**: Array containing field names, in order.
- **Arguments:**
  - result - Resource describing file or other record set.

i5_num_fields

int i5_num_fields (resource result).

- **Description:** Get the numbers of fields for resource.
- **Return Values:** Number of fields.
- **Arguments:**
  - result - Resource describing file or other record set.

i5_result

mixed i5_result (resource result, int/string field]).

- **Description:** Gets one field of the result.
- **Return Values**: Field's contents in current record.
- **Arguments:**
  - result - Resource describing file or other record set.
  - field - Integer or string identifying the field position or name.

Native File Access

i5_open

resource i5_open (string fileName [, int mode][,resource connection]).

- **Description:** Opens native i5 file.
- **Return Values:** Resource, if “open” is successful, false otherwise.
- **Arguments:**
  - name - File name, may include library

- mode - File mode to use:
  I5_OPEN_READ - default
  I5_OPEN_READWRITE
  I5_OPEN_COMMIT
  I5_OPEN_SHRRD
  I5_OPEN_SHRUPD
  I5_OPEN_SHRNUPD
  I5_OPEN_EXCLRD
  I5_OPEN_EXCL
- connection - Connection - result of i5_connect

Note:
OPEN_READ or I5_OPEN_READWRITE modes are required to be combine with other modes. For example, $ret = i5_open ("LIB/FILE", I5_OPEN_READWRITE | I5_OPEN_EXCL);

i5_addnew

bool i5_addnew (resource file [, int mode]).

- **Description:** Creates new record in the file. Use setvalue() to set values in new record, then update() to write it to file. i5_new_record() is an atomic function doing all the work.
- **Return Values:** Resource if open succeeded, false if “open” failed.
- **Arguments:**
  - file - Opened i5 file.
  - mode - I5_ADDNEW_CLEAR: clears all record fields (default).
    I5_ADDNEW_NOCLEAR: does not clear all record fields

i5_edit

bool i5_edit (resource file [, int mode]).

- **Description:** Sets editing mode for the record. In order for a value to be changed, it should be set in edit mode. This locks the record so that other users cannot edit it simultaneously.
- **Return Values:** Boolean success value. Returns false if the record is already being edited by other used.
- **Arguments:**
  - file - i5 file resource.
  - mode - Editing mode:
    I5_EDIT_ONE leaves edit mode after i5_update() and also after reading or i5_delete().

I5_EDIT_ALWAYS remains in edit mode until i5_cancel_edit() is called.
I5_EDIT_AUTO is called automatically therefore there is no need to call i5_update() after setting values.

i5_delete

bool i5_delete (resource file).

- **Description:** Remove current record.
- **Return Values:** Boolean success value. Return is false if the record is already being edited by other used.
- **Arguments:**
  - file - i5 file resource.

i5_cancel_edit

bool i5_cancel_edit (resource result).

i5_setvalue

bool i5_setvalue (resource file, int/string field, mixed value).
bool i5_setvalue (resource file, array values).

- **Description:** Changes the value of the current record. The record should be in edit mode after i5_edit() or created by i5_addnew().
- **Return Values:** Boolean success value.
- **Arguments:**
  - file - i5 file resource.
  - field - Field identifier by name or position.
  - value - Value for the field.
  - values - Set of key=>value parts describing fields to change and their new values.

i5_update

bool i5_update (resource file).

- **Description:** Commits changes done to the file record after i5_edit() or i5_addnew() into the file.
- **Return Values:** Boolean success value.
- **Arguments:**
  - file - i5 file resource.

i5_range_from

bool i5_range_from (resource file,bool included,array values).

- **Description:** Sets an upper range bound for the file. Once the bound is set, the first line for all seeks becomes the line defined by the range.
- **Return Values:** Boolean success value.
- **Arguments:**
  - file - i5 file resource.
  - included - True if the field with this key should be included in the range, false otherwise.
  - values - Values for the key fields - array of key=>value pairs.

i5_range_to

bool i5_range_to (resource result,bool included, array values).

- **Description:** Sets a lower range bound for the file. Once the bound is set, the last entry for all seeks becomes the entry defined by the range.
- **Return Values:** Boolean success value.
- **Arguments:**
  - file - i5 file resource
  - included - True if the field with this key should be included in the range, false otherwise.
  - values - Values for the key fields - array of key=>value pairs.

i5_range_clear

bool i5_range_clear (resource file).

- **Description:** Removes range. Reverses the action of range_from() and range_to().
- **Return Values:** Boolean success value.
- **Arguments:**
  - file - i5 file resource

i5_data_seek

bool i5_data_seek (resource result, int record_number).

- **Description:** Seeks to a specific record of the result.
- **Return Values:** Boolean success value.
- **Arguments:**

- file - i5 file resource.
- Record_number - Number of the record to seek to, starting from 0.

i5_seek

bool i5_seek (resource file, int/string operator, array keyValue).

- **Description:** Goes to a specific record in query/file.
- **Return Values:** Boolean success value.
- **Arguments:**
  - file - i5 file resource
  - operator - Comparison operator. Position is set to first record satisfying the operator. Available operators:

    I5_EQ "="

    I5_GT ">"

    I5_LT "<"

    I5_GE ">="

    I5_LE "<="
  - keyValue - values of the keys to compare

i5_bookmark

int i5_bookmark (resource file).

- **Description:** Return Values the ID of the current record.
- **Return Values:** The ID of the current record that can be used with i5_data_seek() to position on this record again.
- **Arguments:**
  - file - i5 file resource.

i5_free_file

bool i5_free_file (resource file).

- **Description:** Closes file handle and frees file resources.
- **Return Values:** Boolean success value.
- **Arguments:**
  - file - i5 file resource.

Additional functions to the existing API.

i5_new_record

bool i5_new_record (resource file, array data).

- **Description:** Creates a new record in the file and inserts data into it.

- **Return Values:** Boolean success value.
- **Arguments:**
  - file - Opened i5 file resource.
  - data - Array of data fields conforming to file description.

Can be either a flat array or key-value pairs, e.g., i5_setvalue arguments.

i5_update_record

bool i5_update_record (resource file, array data).

- **Description:** Updates the current row with given data.
- **Return Values:** Boolean success value.
- **Arguments:**
  - file - Opened i5 file resource.
  - data - Array of data fields conforming to file description.

Can be either flat array or key-value pairs, like i5_setvalue arguments.

- **Example**:

  $file = i5_open("API/TESTFILE", I5_OPEN_READWRITE);.
  $rec = i5_fetch_row($file, I5_READ_FIRST);.
  i5_update_record($file, array("CODE" => "C-02", "NOM" => "DUPONT", "TYPE" => 3));.
  i5_new_record($file, array('C-105', 'DUPOND', 'Jean', 'Avenue du Qubec', 'Les Ulis', 3, 'FR'));.

i5_delete_record

bool i5_delete_record(resource file).

- **Description:** Removes current record.
- **Return Values:** Boolean success value. False value is returned if the record is already being edited by other used.
- **Arguments:**
  - File - Opened i5 file resource.
- **Example**:

  $file = i5_open("API/TESTFILE", I5_OPEN_READWRITE);.
  i5_new_record($file, array('C-105', 'DUPOND', 'Jean', 'Avenue du Qubec', 'Les Ulis', 3, 'FR'));.
  $rec = i5_fetch_row($file, I5_READ_FIRST);.
  i5_update_record($file, array("CODE" => "C-02", "NOM" => "DUPONT", "TYPE" => 3));.

```
i5_delete_record($file);
```

i5_get_keys

array i5_get_keys(resource file).

- **Description:** Gets information about key fields in the file.
- **Return Values:** An array of integers specifying positions for key fields in the file. Can then use i5_info to discover descriptions of these fields.
- **Arguments:**
  - file - Opened i5 file resource.

SQL File Access

i5_query

resource i5_query (string query [, resource connection])

- **Description:** Executes an SQL statement directly
- **Return Values:** For SELECT request returns resource if statement was executed successfully and FALSE in case of error. For INSERT, UPDATE and DELETE requests returns TRUE if statement was executed successfully and FALSE in case of error.

Note:
i5_query function is suitable for SQL requests without parameters. If you plan to issue the same SQL statement with different parameters, consider using i5_prepare() and i5_execute().

- **Arguments**:
  - Query **-** SQL request string such as SELECT, INSERT, DELETE, UPDATES and etc
  - connection - result of i5_connect

- **Example:**

```
/* Straight request execution */
$query = i5_query("SELECT * FROM EACDEMO/SP_CUST");
if(!$query)){
echo "Error code: " . i5_errno($query) . "<br>";
echo "Error message: " . i5_errormsg($query) . "<br>";
    }
else {
/* Read records and display */
echo "<table>";
while ($values == i5_fetch_row($query, I5_READ_NEXT )) {
echo "<tr>";
echo "<td>" .$values[0]. "</td>";
echo "<td>" .$values[1]. "</td>";
echo "<td>" .$values[2]. "</td>";
echo "<td>" .$values[3]. "</td>";
echo "<td>" .$values[4]. "</td>";
echo "<td>" .$values[5]. "</td>";
echo "</tr>";
}
echo "</table>";
```

i5_prepare

resource i5_prepare (string query [, resource connection])

- **Description:** Prepares an SQL statement to be executed
  Query parameter may include one or several SQL variables if question marks (?) are set at the right places. There are three main advantages using prepared requests in your script: Performance: While preparing a request, database server creates a return optimized path in order to collect the requested data's. Later on, when the i5_prepare prepared request is sent, it will use the path avoiding processor overload with each request sent. Safety: While preparing a request, it is possible to set markers for entry values. Processing the prepared request with entry values, Easycom checks each entry value to make sure that their type match with the column or the description parameters. Advanced Functionality: Markers not only allow introducing entry values in stored procedure, but also allow collecting OUTPUT and INPUT/OUTPUT recording procedure parameters using i5_bind_param function.

- **Return Values**: Returns a statement resource if the SQL statement was successfully parsed and prepared by the database server. FALSE if the database server returned an error.
- **Arguments:**
  - query - SQL request to prepare
  - connection - result of i5_connect

i5_bind_result

bool i5_bind_result (resource result/query, mixed &var1 [,mixed &var2 ...])
-Or-
bool i5_bind_result (resource result/query, mixed &var, string namefield)

- **Description:** Binds a PHP variable to an SQL statement parameter in a statement resource returned by i5_prepare().
- Returns TRUE on success or FALSE on failure.
- **Arguments**:
  - query/stmt **-** 5_prepare prepared request ID
  - var1 , &var2 **-** variables to associate referenced list
  - namfield **-** request field or associated file name

i5_execute

bool i5_execute (resource stmt [,params])

- **Description:** Executes a prepared SQL statement

  i5_execute executes an SQL request prepared with i5_prepare. If the SQL statement returns a result set, for example, a SELECT statement or a CALL to a stored procedure that returns one or more result sets, you can retrieve a row as an array from the stmt resource using i5_fetch_array, i5_fetch_assoc or i5_fetch_row. If the request creates several results sets, i5_next_result function moves pointer to the next available set.

  i5_execute is much more efficient than i5_query if the same request has to be run several times with only few parameter changes.. Refer to i5_prepare for a brief discussion of the advantages of using i5_prepare and i5_execute rather than i5_query.

  A request may contain markers, identified with "?" sign. These markers

can be linked to PHP variables (seer i5_bind_param), the results may be linked to PHP variables using i5_bind_result function.

- **Return Values**: Returns Boolean and updated stmt resource in case of success FALSE if it fails
- **Arguments**:
  - stmt - A prepared statement returned from i5_prepare
  - params **-** Input parameters matching any parameter markers contained in the prepared statement.
- **Example:**

```
$town = "Paris";
/* Prepare a request */
$req = i5_prepare("SELECT area FROM cities WHERE Name=?");
if ($req) {
    /* Associate SQL variables */
    i5__bind_param($req, $town);
    /* Execute the request */
    i5_execute($req);
    /* Associate the results variables */
    i5_bind_result($req, $region);
    /* Read records */
    i5_fetch_row($req);
    printf("%s is in area %s\n", $town, $region);
```

i5_getblob

string i5_getblob(resource result, int position)

-Or-

string i5_getblob(resource result, string namefield)

- **Description**: Reads binary data from a BLOB field type.
  This function applies to SELECT type (i5_queryi5_query or i5_executei5_execute) requests containing one or more BLOB type fields.

Note:

Reading and writing a blob requires a transaction.

- **Return Values**: String with BLOB binary chain or FALSE on failure
- **Arguments**:
  - result - File ID
  - position - BLOB field index

- namfield - BLOB field name

- **Example:**

```
$sql = "SELECT BLOB_COLUMN FROM BLOB_TABLE;";
$res = i5_query($sql);
$line = i5_fetch_row($res);

/* $line[0] contains blob ID */
$blob_data = i5_getblob($line[0], $res);
/* the blob can be displayed or processed */
```

i5_setblob

bool i5_setblob (resource stmt, int position, string blob)

- **Description**: Writes a binary data in a BLOB field type.
  This function only applies to parameterized requests resources and is used the same way as i5_setparam function.

Note:
Writing a blob requires a transaction.

- **Return Values:** TRUE on success or FALSE on failure
- **Arguments:**
  - result - Parameterized file ID
  - position - Parameter index
  - blob - Binary chain content

- **Example**

```
/* Writing jpeg file content in blob */
$slq = "INSERT INTO CONTACTS (NAME, PRENOM, PHOTO) VALUES
(?,?,?)";
$req_prepa = i5_prepare($sql);
if ($req_prepa) {
    $name = "DUPONT";
    $préname = "HENRY";
    $file_image = fopen("hdupont.jpg", 'r');
    $contents = fread($file_image, filesize($file_image));
    $ret0 = i5_setparam($file_as, 0, $name);
    $ret1 = i5_setparam($file_as, 1, $prenom);
    $ret2 = i5_setblob($file_as, 2, $contents);
    $ret = i5_execute($file_as);
    if ($ret) {echo "Blob writing successful.\n";}
    }
```

i5_setparam

bool i5_setparam (resource stmt, int position, mixed value)

- **Description:** Allocates parameter to parameterized request.
  This function is an alternative to i5_bind_param function (automatically linked). It allows explicit value allocation to a parameter.

Note:
Request must be prepared with i5_prepare function.

- **Return Values:** TRUE on success or FALSE on failure
- **Arguments:**
  - stmt - i5_prepare prepared request ID
  - position - parameter index (marker) in the request
  - value - parameter allocated value

- **Example 1:**

```
$insert = 'INSERT INTO animals (id, race, name, weight) VALUES
(?, ?, ?, ?)';
$req = i5_prepare($insert);
$animals = array(0, 'cat', 'Mistinguette', 3.2);
if ($req) {
    $result = i5_execute($req, $animals);
    if ($result) {
        print "Mistinguette adding successful.";
    }
    i5_setparam($req, 3, "Minouche");
    i5_setparam($req, 4, 3.8);
    $result = i5_execute($req, $animals);
    if ($result) {
        print "Hercule adding successful.";
    }
```

- **Example 2** - Calling stored procedures with IN parameter

  The stored procedure in the following example accepts one parameter:

1. Create table
2. an input (IN) parameter that accepts the name of the first animal as input
3. an input-output (INOUT) parameter that accepts the name of the second animal as input and returns the string TRUE if an animal in the database matches that name
4. an output (OUT) parameter that returns the sum of the weight of the two identified animals

In addition, the stored procedure returns a result set consisting of the animals listed in alphabetic order starting at the animal corresponding to the input value of the first parameter and ending at the animal corresponding to the input value of the second parameter.

```
<?php
//
//CREATE TABLE SQL_LIB/TEST2 (A DATE NOT NULL WITH DEFAULT)
//
//
//create procedure SQL_LIB/test_a1(in parm1 date)
//language SQL
//begin


//
//set transaction isolation level UR;
//insert into SQL_LIB/TEST2 values(parm1) ;
//commit;
//end;
$user     = 'USER';
$password = 'PASS';
$conn_resource = i5_connect('127.0.0.1',$user,$password );
echo "Begin <br>";
if (!$conn_resource) {
echo  i5_errormsg();
exit();

}
$sql = "CALL SQL_LIB/TEST_A1(?)";
$stmt= i5_prepare($sql);
$val = '2007-05-22';
$ret = i5_paramdesc($stmt, I5_TYPE_CHAR, 0, 10, 0, I5_INOUT);
$ret = i5_setparam($stmt, 0, $val);
$result = i5_execute($stmt );
if($result === false){
echo "Execute Error:". i5_errno()."  Msg:".i5_errormsg()."<br>";

//echo $err;
}
else {
"<br>executed";
}
echo "<br>end";
?>
```

i5_free_query

bool i5_free_query (resource query)

- **Description:** Frees SQL request result
  Removes a query type resource (i5_query or i5_execute) from memory
  This function needs only to be called if your script requires too much memory, when a request returns very large results or if a large requests number are processed and may overload the web server memory. It is recommended to use this function to free memory resource used by SQL request. All memory resources are freed when the SQL request is ended.
- **Return Values:** TRUE on success or FALSE on failure
- **Arguments:**
  - query - query resource

Transactions

i5_transaction

bool i5_transaction (int mode [, resource connection])

- **Description:** Starts transaction.
- **Return Values:** Returns TRUE if transaction has started, FALSE in case of error
- **Arguments**:
  - **mode -** Transaction modes:
    - I5_ISOLEVEL_CHG - READ UNCOMMITED, READ WRITE (UR)
      - Modified records remain locked.
      - Modifications are showed
    - I5_ISOLEVEL_CS - READ COMMITED (CS)
      - Read records are locked.
      - Modified records remain locked.
      - Changes are not showed
    - I5_ISOLEVEL_ALL - REPEATABLE READ (RS)
      - Read records remain locked.
      - Modified records remain locked.
      - Modifications are not showed.
    - I5_ISOLEVEL_NONE - No transactions
      - Each record is commited immediately
  - **connection -** result of i5_connect

- **Example:**

```
<?php
$conn = i5_connect("MY_i5", "USER", "PASSWORD");
if ($conn) {
    $res = i5_query("SELECT count(*) FROM animals");
    $rec = i5_fetch_array($res );
    echo $rec[0] . "\n";

       /* Start a transaction */
    i5_transaction(I5_ISOLEVEL_NONE, $conn);

    /* Add records to ANIMALS table */
    i5_query("INSERT INTO animals VALUE 'Cat', 'Mistigri'");

    $res = i5_query("SELECT count(*) FROM animals");
    $rec = i5_fetch_array($res);
    echo $res[0] . "\n";
```

i5_commit

bool i5_commit([string comment] [resource connection])

- **Description:** Commits an in-progress transaction.
- **Return Values**: TRUE if transaction is valid, FALSE in case of error.
- **Arguments:**
  - comment - a transaction comment that will be added to the journal
  - Connection **-** result of i5_connect

- **Example:**

```
$conn = i5_connect("MY_i5", "USER", "PASSWORD");
if ($conn) {
    $res = i5_query("SELECT count(*) FROM animals");
    $rec = i5_fetch_array($res );
    echo $rec[0] . "\n";

    /* Start a transaction */
    i5_transaction(I5_ISOLEVEL_NONE);

    /* Insert records to ANIMALS table*/
    i5_query("INSERT INTO Animals VALUES ('CAT', 'Misstic', 'F',
3.2)");
    i5_query("INSERT INTO Language VALUES ('DOG', 'Hercule',
'M', 4.4)");

    $res = i5_query("SELECT count(*) FROM animals");
    $rec = i5_fetch_array($res);
    echo $rec[0] . "\n";

    /* Commit the changes */
    i5_commit($conn);

    $res = i5_query("SELECT count(*) FROM animals");
    $rec = i5_fetch_array($res);
    echo $rec[0] . "\n";
    i5_close($conn);
    }
```

i5_rollback

bool i5_rollback ([resource connection])

- **Description:** Rolls back a transaction.
- **Return Values:** TRUE on success or FALSE on failure
- **Arguments**:
- **connection** - result of i5_connect

- **Example:**

```
<?php
$conn = i5_connect("MY_i5", "USER", "PASSWORD");
if ($conn) {
    $res = i5_query("SELECT count(*) FROM animals");
    $rec = i5_fetch_array($res );
    echo $res[0] . "\n";

    /* Start a transaction*/
    i5_transaction($conn);

    /* Delete all records from the ANIMALS table */
    i5_query("DELETE * FROM animals");

    $res = i5_query("SELECT count(*) FROM animals");
    $rec = i5_fetch_array($res);
    echo $res[0] . "\n";
    /* Cancel the DELETE operation */
    i5_rollback($conn);

    $res = i5_query("SELECT count(*) FROM animals");
    $rec = i5_fetch_array($res);
    echo $rec[0] . "\n";
    i5_close($conn);
   }
?>
```

Data Queues

i5_dtaq_prepare

resource i5_dtaq_prepare(string name, array description [,int key][,resource connection])

- **Description:** Opens a data queue with optional description.
- **Return Values:** Resource if OK, false if failed.
- **Arguments:**
  - name - The queue name

- description - Data description in format defined by program¬_prepare. For more, see Easycom PHP Data Description at the end of this document.
- key - key size - for keyed DataQ
- connection - Connection - result of i5_connect

i5_dtaq_receive

mixed i5_dtaq_receive(resource queue[, string/int operator, string key][, int timeout])

- **Description:** Reads data from the data queue.
- **Return Values:** False if could not read because of error or timeout, the data read from the queue otherwise.
- **Arguments:**
  - queue - resource received from dtaq_open
  - operator:
    "EQ"
    "GT"
    "LT"
    "GE"
    "LE"
  - key- key value to look for
  - timeout - timeout value in seconds

i5_dtaq_send

bool i5_dtaq_send(resource queue, string key, mixed data)

- **Description:** Puts data to the data queue.
- **Return Values:** False if could not be written because of error, true otherwise.
- **Arguments:**
  - queue - resource received from dtaq_open
  - key - key value to look for
  - data - data to put into the queue

The data should conform to the description format, and can be either in flat array or key->value pair array.

i5_dtaq_close

bool i5_dtaq_close(resource queue)

- **Description:** Free program resource handle.
- **Return Values:** Bool success value.
- **Arguments:**
  - queue - resource received from dtaq_open
- **Example 1**:

```
<?php
$description = array("Name"=>"DATA", "Type"=>I5_TYPE_CHAR,
"Length"=>50);
$dtaqHdl_KEY = i5_dtaq_prepare("EACDEMO/DTAQ_KEY", $description,
5);
$ret = i5_dtaq_send($dtaqHdl_KEY, "mykey", "the dataqueue test
data");
var_dump($ret);
if(!$ret) var_dump(i5_error());
$ret = i5_dtaq_receive($dtaqHdl_KEY, "EQ", "mykey");
var_dump($ret);
?>
```

- **Example 2:**

```
<?php
$descriptionC = array("DSName"=>"PS", "DSParm"=>array(
array("Name"=>"PS1", "Type"=>I5_TYPE_CHAR, "Length"=>"10"),
array("Name"=>"PS2", "Type"=>I5_TYPE_PACKED, "Length"=>"10.4"),
array("Name"=>"PS3", "Type"=>I5_TYPE_CHAR, "Length"=>"10")
)
);
$dtaqHdl_KEY = i5_dtaq_prepare("EACDEMO/DTAQ_KEY",
$descriptionC, 10);
$parameter = array("PS1"=>"test1", "PS2"=>13.1415,
"PS3"=>"test2");
$key = "abcd";
$ret = i5_dtaq_send($dtaqHdl_KEY, $key, $parameter);
var_dump($ret);
$ret = i5_dtaq_receive($dtaqHdl_KEY, "EQ", $key);
var_dump($ret);
?>
```

System Values

i5_get_system_value

string i5_get_system_value(string name[, resource connection]).

- **Description:** Retrieves system value
- **Return Values:** System value, false if not found.
- **Arguments:**
  - name - Name of the system value.
  - connection - Connection - result of i5_connect.
- **Example**:

  print "Date is: ".i5_get_system_value("QDATE");.

User Spaces

i5_userspace_create

bool i5_userspace_create(properties[, resource connection]).

- **Description:** Creates a new user space object.
- **Return Values:** Boolean success value
- **Arguments:**
  - properties -

    I5_INITSIZE – The initial size of the user space being created. This value must be from 1 byte to 16, 776, 704 bytes.
    I5_DESCRIPTION – user space description
    I5_INIT_VALUE – The initial value of all bytes in the user space.
    I5_EXTEND_ATTRIBUT – extended attribute. The extended attribute must be a valid *NAME. For example, an object type of *FILE has an extended attribute of PF (physical file), LF (logical file), DSPF (display file), SAVF (save file), and so on.
    I5_AUTHORITY – The authority you give users who do not have specific private or group authority to the user space
    I5_LIBNAME – Library name where the user space is located
    I5_NAME – User space name (10 char max)
  - connection - Result of i5_connect

i5_userspace_prepare

resource i5_userspace_prepare(string name, array description [, resource connection]).

- **Description:** Opens a user space and prepares it to be run.
- **Return Values:** Resource if open succeeded, false if open failed.
- **Arguments:**
  - name - User space name in library/object format
  - description - Data description in format defined by program_prepare. See PHP Data Description, page 58.
  - connection - Result of i5_connect

i5_userspace_get

resource i5_userspace_get(resource user space, array params)

- **Description:** Retrieve user space data.
- **Return Values:** Boolean success value.
- **Arguments:**
  - user space - User Space resource opened by i5_userspace_prepare
  - params - Parameters according to description. If given as flat array, then parameters are assigned in order

i5_userspace_put

bool i5_userspace_put(resource user space, params)

- **Description:** Add user space data
- **Return Values:** Boolean success value.
- **Arguments:**
  - user - space User Space resource opened by i5_userspace_prepare
  - params - Parameters according to description. If given as flat array, then parameters are assigned in order

Job Log List

i5_jobLog_list

resource i5_jobLog_list([array elements, resource connection])

- **Description:** Opens job log.

- **Return Values:** The resource for fetching job log list if OK and false if failed.
- **Arguments:**
  - elements - JobName, JobUser, JobNumber, MaxMessage, Direction (default is current job)
  - connection - Result of i5_connect

Use i5_jobLog_list_read function to retrieve the job entries from this handle.

i5_jobLog_list_read

array i5_jobLog_list_read(resource list)

- **Description:** Get an array for a job log entry.
- **Return Values:** Array with the message element if OK, false if failed.
- **Arguments:**
  - list - Resource returned by i5_jobLog_list function

i5_jobLog_list_close

bool i5_jobLog_list_close (resource handle)

- **Description:** Close handle received from i5_jobLog_list().
- **Return Values:** Boolean success value
- **Arguments:**
  - handle - Job list handle as returned by i5_jobLog_list()

Active Job List

i5_job_list

resource i5_job_list([array elements, resource connection])

- **Description:** Open active job list.
- **Return Values:** The resource for fetching job list if OK and false if failed.
- **Arguments:**
  - elements - JobName, JobUser, JobNumber, JobType, Direction (default is current job)
  - connection - Result of i5_connect

Use i5_job_list_read function to retrieve the job entries from this handle.

i5_job_list_read

array i5_job_list_read(resource list)

- **Description:** Get an array for an active job entry.
- **Return Values:** Array with the job entry element if OK, false if failed.
- **Arguments:**
  - List - Resource returned by i5_job_list function

i5_job_list_close

bool i5_job_list_close (resource handle)

- **Description:** Close handle received from i5_job_list().
- **Return Values:** Boolean success value
- **Arguments:**
  - handle - Job list handle as returned by 15_job_list()

Data Areas

i5_data_area_create

bool i5_data_area_create(string name, int size[, resource connection]).

- **Description:** Creates data area of given size
- **Return Values:** Boolean success value.
- **Arguments:**
  - name - Name of the data area.
  - size - Size in bytes of the data area.
  - connection - result of i5_connect .

i5_data_area_read

string data_area_read(string name[, int offset, int length][, resource connection]).

- **Description:** Reads data from the area
- **Return Values:** String data if read successful, false if read failed (including when offset is wrong).
- **Arguments:**
  - name - Name of the data area.
  - offset - Offset for the data.

- length - Length of the data to read, -1 means whole area.
- connection - Connection - result of i5_connect.

If no offset is specified, all the area is read.

i5_data_area_write

bool data_area_write(string name, string value[, int offset, int length][, resource connection]).

- **Description:** Writes data to the area
- **Return Values:** Boolean success value.
- **Arguments:**
  - name - Name of the data area.
  - value - Value to write.
  - Offset - Offset for the data.
  - length - Length of the data to read.
  - connection - result of i5_connect .

If no offset is specified, all the area is written. If value is shorter than length it is padded to the length. If it's longer it is truncated.

i5_data_area_delete

bool data_area_delete(string name[, resource connection]).

- **Description:** Delete the data area
- **Return Values:** Boolean success value.
- **Arguments:**
  - name - Name of the data area.
  - connection - Connection - result of i5_connect.

Spooled File

i5_spool_list

resource i5_spool_list([array description][, resource connection])

- **Description:** Create an spool file lists, of certain output queue or for all queues.
- **Return Values:** resource if OK, false if failed
- **Arguments:**
  - description - The data by which the sppol files will be filtered, array with following keys:

- username - username that created the job
- outq - qualified name for the output queue containing the spool file
- userdata - the user-supplied key data for the spool file. All keys are optional and can be provided together
- connection - result of i5_connect.

i5_spool_list_read

array i5_spool_list_read(resource spool_list)

- **Description:** Gets spool file data from the queue.
- **Return Values:** next spool file data array in the list, or false if queue is empty.
- The data will be formated using SPLF0300 format. See following link for more details: http://publib.boulder.ibm.com/infocenter/iseries/v5r4/index.jsp?topic=/apis/QUSLSPL.htm
- **Arguments:**
  - Spool_list resource received from i5_spool_list

i5_spool_list_close

void i5_spool_list_close(resource spool_list)

- **Description:** Free spool list resource
- **Return Values:** Boolean success value
- **Arguments:**
  - Queue - resource received from i5_spool_list

i5_spool_get_data

string i5_spool_get_data(string spool_name, string jobname, integer job_number, string username, integer spool_id [,string filename])

- **Description:** Get the data from the spool file.
- **Return Values:** String if no file name passed as parameter, false if function failes
- **Arguments:**
  - spool_name - The spool file name
  - job_name - The name of the job that created the file
  - job_number - The number of the job that created the file
  - username - The username of the job that created the file

- spool_id - ID of the spool file in the queue (as returned by outq_read)
- filename - IFS filename to store the data. If not provided, the data is returned as string

Object Listing

i5_objects_list

resource i5_objects_list(string library, [string name, string type, resource connection])

- **Description:** Open an object list.
- **Return Values:** Resource for fetch if everything is OK, false on error.
- **Arguments:**
  - library - Library name (can be also *CURLIB or I5_CURLIB)
  - name - Name or wildcard of objects to read, default is “all”.
  - type - Object type to fetch (*ALL or I5_ALL_OBJECTS for all)
  - connection - Connection - result of i5_connect

i5_objects_list_read

array i5_objects_list_read (resource list)

- **Description:** Get an array for an object list entries.
- **Return Values:** Array with the object element if OK; false if failed.
- **Arguments:**
  - List - Resource returned by i5_objects_list

i5_ objects_list _close

bool i5_ objects_list_close (resource handle)

- **Description:** Close handle received from i5_ objects_list ().
- **Return Values:** Boolean success value
- **Arguments:**
  - handle - Object list handle as returned by i5_ objects_list ()

PHP Data Description

Data structures are defined via PHP as follows:

Main data is the array of values, having the following fields:

- **Name** - name of the field
- **Type** - type of the field, can be:
  - I5_TYPE_SHORT
  - I5_TYPE_LONG
  - I5_TYPE_DOUBLE
  - I5_TYPE_BIN
  - I5_TYPE_DATE
  - I5_TYPE_TIME
  - I5_TYPE_TIMESTP
  - I5_TYPE_DBCS
  - I5_TYPE_LONG8
  - I5_TYPE_NUMERICCHAR
  - I5_TYPE_BLOB
  - I5_TYPE_CLOB
  - I5_TYPE_UNICODE
  - I5_TYPE_VARCHAR
  - I5_TYPE_VARBIN
- **Length**
  - For CHAR, BYTE - integer describing length. Length can be number or name of the variable holding the length in the data structure.
  - For PACKED, ZONED - string "NUMBER.NUMBER" defining length and precision
  - For STRUCT - array containing data definition of the structure
  - For INT, FLOAT - ignored
- **IO**
  - I5_IN
  - I5_OUT
  - default is input, these values can be OR'ed together to get input-output value

- **Count** (optional) - repetition count if the field is an array
- **CountRef** (optional) - reference to the repetition count if the field is an array

Data structure is defined via PHP as follows:

- **DSName** - name of the parameter
- **DSParm** (optional) - array of the parameter of the Data structure. Each parameter is defined by a simple data definition.

Example:

```
<?php
$description = Array(
array("Name"=>"P1", "IO"=>I5_INOUT,
"Type"=>I5_TYPE_CHAR, "Length"=>"10", "count"=>5),
array("Name"=>"P2C", "IO"=>I5_INOUT, "Type"=>I5_TYPE_LONG),
array("Name"=>"P2", "IO"=>I5_INOUT, "Type"=>I5_TYPE_CHAR,
"Length"=>"1", "countRef"=>"P2C"),
array("DSName"=>"PS", "count"=>2, "DSParm"=>array(
array("Name"=>"PS1", "IO"=>I5_IN|I5_OUT, "Type"=>I5_TYPE_CHAR,
"Length"=>"10"),
array("Name"=>"PS2", "IO"=>I5_IN|I5_OUT, "Type"=>I5_TYPE_CHAR,
"Length"=>"10"),
array("Name"=>"PS3", "IO"=>I5_IN|I5_OUT, "Type"=>I5_TYPE_CHAR,
"Length"=>"10"),
                    )
                )
            );
$prg = i5_program_prepare("MYLIB/PERSONPGM", $description);
?>
```

Command Constants

I5_CURLIB (Default Value = "*CURLIB")

I5_ALL_OBJECTS (Default value = "*ALL")

I5_ALL_NAMES (Default value = "*")

I5_LIST_MINIMAL

I5_LIST_DETAILED

I5_LIST_FULL

Active Job (i5_job_list) array elements constants

i5_JOB_ACT_JOB_STS
I5_JOB_BRKMSG
I5_JOB_CNTRYID
I5_JOB_POOL_ID
I5_JOB_PROCESS_UNIT_TIME_DB
I5_JOB_DATETIME_SCHED
I5_JOB_DATSEP
I5_JOB_DFTWAIT
I5_JOB_DFTCCSID
I5_JOB_ENDSEV
I5_JOB_FUNC_NAME
I5_JOB_GRPPRFNAME
I5_JOB_ACCOUNT_CODE
I5_JOB_QUEUE_NAME
I5_JOB_JOBMSGQFL
I5_JOB_USRID_SETTING
I5_JOB_TYPE_ENHANCED
I5_JOB_LOGCLPGM
I5_JOB_MODE_NAME
I5_JOB_MAX_THREADS
I5_JOB_MSGRPL
I5_JOB_ MCH_LCKW
I5_JOB_OUTQ_NAME
I5_JOB_PRTDEVNAME
I5_JOB_PROG_RETCODE
I5_JOB_RESPONSE_TIME
I5_JOB_STRSEQ
I5_JOB_SBMJOB
I5_JOB_SYSPOOLID
I5_JOB_SGNSTS
I5_JOB_TIMSEP
I5_JOB_TMPSTGK
I5_JOB_TIME_NONDB_LCKW
I5_JOB_ALW_MULTI_THREADS
I5_JOB_CANCEL_KEY
I5_JOB_USRPRF
I5_JOB_CHAR_ID_CTRL
I5_JOB_DATETIME_ACTIVE
I5_JOB_DATETIME_JOBQ
I5_JOB_DBCS_CAP
I5_JOB_DEVRCYACN
I5_JOB_DECFMT
I5_JOB_ENDSTS
I5_JOB_FUNC_TYPE
I5_JOB_GRPPRFNAME_SUP
I5_JOB_DATE
I5_JOB_QUEUE_PTY
I5_JOB_JOBMSGQ_SIZE
I5_JOB_END_REASON
I5_JOB_LANGID
I5_JOB_LOGSEV
I5_JOB_MAX_PROC_UNIT_TIME
I5_JOB_MAX_TMP_STG_M
I5_JOB_INTERACTIVE_TRS
I5_JOB_NONDB_LCKW
I5_JOB_OUTQ_PTY
I5_JOB_PURGE
I5_JOB_PENDING_SGNSET
I5_JOB_RUNPTY
I5_JOB_STS_MSGHDL
I5_JOB_SBMMSGQ
I5_JOB_SPCLENV
I5_JOB_SVRTYPE
I5_JOB_TIMESLICE
I5_JOB_TIME_DB_LCKW
I5_JOB_THREADCNT
I5_JOB_ACT_ENDJOB_STS
I5_JOB_CCSID
I5_JOB_COMPLETION_STS
I5_JOB_PROCESS_UNIT_TIME
I5_JOB_DATETIME_IN
I5_JOB_DATFMT
I5_JOB_DDM_HANDLE
I5_JOB_DEVNAME
I5_JOB_DATETIME_END
I5_JOB_EXITKEY
I5_JOB_SIGNED_JOB
I5_JOB_INQMSGRPLY
I5_JOB_DESC_NAME
I5_JOB_SWITCHES
I5_JOB_USRID
I5_JOB_LOG_PENDING
I5_JOB_LOGLVL
I5_JOB_LOGTEXT
I5_JOB_MAX_TMP_STG_K
I5_JOB_MEM_POOL_NAME
I5_JOB_DB_LCKWAIT
I5_JOB_AUX_IOREQ
I5_JOB_PRTTEXT
I5_JOB_PRD_RETCODE
I5_JOB_PROCESS_ID
I5_JOB_ROUTING_DATA
I5_JOB_STS_JOBQ
I5_JOB_SBSD
I5_JOB_SGNBLK_MASK
I5_JOB_SPLFILE_ACTION
I5_JOB_TIMESLICE_END
I5_JOB_TIME_MCH_LCKW

Job Log Constants (i5_jobLog_list) array elements constants

I5_LOBJ_MESSAGE_SEVERITY
I5_LOBJ_MESSAGE_IDENTIFIER
I5_LOBJ_MESSAGE_TYPE
I5_LOBJ_MESSAGE_FILENAME
I5_LOBJ_MESSAGE_FILELIBRARY
I5_LOBJ_DATASENT
I5_LOBJ_TIMESENT
I5_LOBJ_TIMESENT_MICRO
I5_LOBJ_ALERTOPT
I5_LOBJ_RPLDATA1
I5_LOBJ_MSG
I5_LOBJ_MSGDTA
I5_LOBJ_MSGHLP
I5_LOBJ_MSGHLPDTA
I5_LOBJ_MSGHLPDTAFMT
I5_LOBJ_DFTRPLY
I5_LOBJ_SNDNAME
I5_LOBJ_SNDTYPE
I5_LOBJ_SNDPGM
I5_LOBJ_SNDMOD
I5_LOBJ_SNDPROC
I5_LOBJ_RCVTYPE
I5_LOBJ_RCVPROG
I5_LOBJ_RCVMOD
I5_LOBJ_RCVPROC
I5_LOBJ_MSGFILE
I5_LOBJ_PROBLEMID
I5_LOBJ_RPLYSTS
I5_LOBJ_RQSSTS
I5_LOBJ_RQSLVL
I5_LOBJ_TXTCCSID
I5_LOBJ_DATACCSID

Errors

I5_TYPE_CHAR
I5_ERR_OK
I5_ERR_ERROR
I5_ERR_TOOMUCHOPENFILE
I5_ERR_MEMALLOC

Data Retrieval Errors

I5_ERR_INVALIDPTR
I5_ERR_NOLINKDEFINED
I5_ERR_FIELDNOTFOUND
I5_ERR_NOCURRENTRECORD
I5_ERR_INVALIDFIELDNBR
I5_ERR_NULLNOTALLOWED
I5_ERR_INVALIDKEYLEN
I5_ERR_BADSESSION
I5_ERR_INVALIDKEYNBR
I5_ERR_WRONGLOGIN
I5_ERR_NOTENABLETOUPDATE
I5_ERR_NOTENOUGHRIGHTS
I5_ERR_INVALIDOPENMODE
I5_ERR_FIELDNULL
I5_ERR_RECORDNOTFOUND
I5_ERR_INVALIDTYPE
I5_ERR_RECORDLOCKED
I5_ERR_INVALIDINFO
I5_ERR_BEOF
I5_ERR_NOTTYPEPROPERTY
I5_ERR_FILELIMITS
I5_ERR_RECORDCHANGED
I5_ERR_NOTCONNECTED
I5_ERR_ALLREADYINTRAN
I5_ERR_INVALIDSEQ
I5_ERR_NOTINTRAN
I5_ERR_NORANGESET

Function Errors

I5_ERR_PHP_HDLDFT
I5_ERR_PHP_HDLCONN
I5_ERR_PHP_HDLBAD
I5_ERR_PHP_OPTIONSTYPE
I5_ERR_PHP_OPTIONSNUMBER
I5_ERR_PHP_RESOURCE_BAD
I5_ERR_PHP_TYPEPARAM
I5_ERR_PHP_NBPARAM_BAD
I5_ERR_PHP_TYPEGET
I5_ERR_PHP_OPERATOR_BAD
I5_ERR_PHP_BOOKMARK
I5_ERR_PHP_NOT_BOOKMARK
I5_ERR_PHP_CALL_BINDPARAM
I5_ERR_PHP_GETPARAM
I5_ERR_PHP_BINDPARAM
I5_ERR_PHP_PARAM_DESC
I5_ERR_PHP_BLOBSIZE
I5_ERR_PHP_VARIABLE
I5_ERR_PHP_INTERNAL
I5_ERR_PHP_EXECUTE
I5_ERR_PHP_NO_COMMAND
I5_ERR_PHP_EMPTY_ARRAY
I5_ERR_PHP_NO_KEYNAME
I5_ERR_PHP_NO_PARMNAME
I5_ERR_PHP_NO_ZVALUE
I5_ERR_PHP_COMMAND_ERROR
I5_ERR_PHP_DATAREA_READ
I5_ERR_PHP_GET_SYSVAL
I5_ERR_PHP_ELEMENT_MISSING
I5_ERR_PHP_BAD_DEF
I5_ERR_PHP_BAD_KEYNAME
I5_ERR_PHP_NO_DS_VALUE
I5_ERR_PARAMNOTFOUND
I5_ERR_ENDOFOCC
I5_ERR_PHP_BAD_DS_INPUT
I5_ERR_DESC_UNEXP
I5_ERR_DQDESC_UNSUPP
I5_ERR_DESC_WRONG_DATAOP
I5_ERR_INCORRECTVALUE
I5_ERR_PHP_BAD_PROG_NAME
I5_ERR_PHP_AS400_MESSAGE
I5_ERR_PHP_NOT_DTAQ_KEY
I5_ERR_PHP_DTAQ_BADKEY
I5_ERR_PHP_DESC_EMPTY
I5_ERR_PHP_BAD_LEN_PROP
I5_ERR_PHP_LIST_PROP
I5_ERR_PHP_SPOOL_FILE_FOPEN
I5_ERR_PHP_API_LENGTH

Easycom PHP Data Description

Short Data Format

Data structures are defined via PHP as follows:
Main data is an array of key-value pairs, where key is the parameter name and value is the array of:

- type - one of Data types
- type modifier
  - for CHAR, BYTE - integer describing length. Length can be number or name of the variable holding the length in the data structure.
  - for PACKED, ZONED - string "NUMBER.NUMBER" defining length and precision
  - for STRUCT - array containing data definition of the structure
  - for INT, FLOAT - ignored
- direction (optional) - one of I/O values
- count (optional)
  - if integer - repetition count if the field is an array
  - if string - reference to the repetition count field

Example:

```
<?php
$person = array(
    "name" => array(I5_TYPE_CHAR, 50),
    "age" => array(I5_TYPE_INT, 0),
    "ID" => array(I5_TYPE_BYTE, 10)
);
$data = array(
"person" => array(I5_TYPE_STRUCT, $person),
"last_accesses" => array(I5_TYPE_INT, 0, I5_OUT, 3),
"account_balance" => array(I5_TYPE_PACKED, "10.3", I5_OUT)
);
$prg = i5_program_prepare("MYLIB/PERSONPGM", $data);
?>
```

In any place data a description is required; the name of the file with external data structure description can be used instead.

Long Data Format

Data structure is defined via PHP as follows:
Main data is the array of values, having following fields:

- Name - name of the field
- Type - type of the field, can be one of Data types
- Length
  - for CHAR, BYTE - integer describing length. Length can be number or name of the variable holding the length in the data structure.
  - for PACKED, ZONED - string "NUMBER.NUMBER" defining length and precision
  - for STRUCT - array containing data definition of the structure
  - for INT, FLOAT - ignored
- IO - can be one of I/O values

Data structure is defined via PHP as follows:

- DSName - name of the parameter
- DSParm (optional) - array of the parameter of the Data structure. Each parameter is defined by a data definition in the same format as described here.
- Count (optional) - repetition count if the field is an array
- CountRef (optional) - reference to the repetition count if the field is an array

Example:

```
<?php
$description = Array(
                          array("Name"=>"P1", "IO"=>I5_INOUT,
"Type"=>I5_TYPE_CHAR, "Length"=>"10", "count"=>5),
                          array("Name"=>"P2C", "IO"=>I5_INOUT,
"Type"=>I5_TYPE_LONG),
                          array("Name"=>"P2", "IO"=>I5_INOUT,
"Type"=>I5_TYPE_CHAR, "Length"=>"1", "countRef"=>"P2C"),
                          array("DSName"=>"PS", "count"=>2,
"DSParm"=>array(
                                    array("Name"=>"PS1",
"IO"=>I5_IN|I5_OUT, "Type"=>I5_TYPE_CHAR, "Length"=>"10"),
                                    array("Name"=>"PS2",
"IO"=>I5_IN|I5_OUT, "Type"=>I5_TYPE_CHAR, "Length"=>"10"),
                                    array("Name"=>"PS3",
"IO"=>I5_IN|I5_OUT, "Type"=>I5_TYPE_CHAR, "Length"=>"10"),
                                    )
                          )
                 );
$prg = eac_program_prepare("MYLIB/PERSONPGM", $description);
?>
```

Data Types

I5_TYPE_CHAR	I5_TYPE_INT
I5_TYPE_PACKED	I5_TYPE_ZONED
I5_TYPE_FLOAT	I5_TYPE_BYTE
I5_TYPE_STRUCT (long format uses DSName to define structure)	

I/O Values

- I5_IN
- I5_OUT

These values can be OR'ed together to get input-output value:

- I5_TYPE_RETVAL - return value for procedure call
- I5_TYPE_BYVAL - by-value parameter for procedure call
- default is input

Error Types

I5_ERR_OK	I5_ERR_ERROR
I5_ERR_TOOMUCHOPENFILE	I5_ERR_MEMALLOC
I5_ERR_INVALIDPTR	I5_ERR_FIELDNOTFOUND
I5_ERR_INVALIDFIELDNBR	I5_ERR_INVALIDKEYLEN
I5_ERR_INVALIDKEYNBR	I5_ERR_NOTENABLETOUPDATE
I5_ERR_INVALIDOPENMODE	I5_ERR_RECORDNOTFOUND
I5_ERR_RECORDLOCKED	I5_ERR_BEOF
I5_ERR_FILELIMITS	I5_ERR_NOTCONNECTED
I5_ERR_INVALIDSEQ	I5_ERR_NORANGESET
I5_ERR_NOLINKDEFINED	I5_ERR_NOCURRENTRECORD
I5_ERR_NULLNOTALLOWED	I5_ERR_BADSESSION
I5_ERR_WRONGLOGIN	I5_ERR_NOTENOUGHRIGHTS
I5_ERR_FIELDNULL	I5_ERR_INVALIDTYPE
I5_ERR_INVALIDINFO	I5_ERR_NOTTYPEPROPERTY
I5_ERR_RECORDCHANGED	I5_ERR_ALLREADYINTRAN
I5_ERR_NOTINTRAN	I5_ERR_PHP_HDLDFT
I5_ERR_PHP_HDLCONN	I5_ERR_PHP_HDLBAD
I5_ERR_PHP_OPTIONSTYPE	I5_ERR_PHP_OPTIONSNUMBER
I5_ERR_PHP_RESOURCE_BAD	I5_ERR_PHP_TYPEPARAM
I5_ERR_PHP_NBPARAM_BAD	I5_ERR_PHP_TYPEGET
I5_ERR_PHP_OPERATOR_BAD	I5_ERR_PHP_BOOKMARK
I5_ERR_PHP_NOT_BOOKMARK	I5_ERR_PHP_CALL_BINDPARAM
I5_ERR_PHP_GETPARAM	I5_ERR_PHP_BINDPARAM
I5_ERR_PHP_PARAM_DESC	I5_ERR_PHP_BLOBSIZE
I5_ERR_PHP_VARIABLE	I5_ERR_PHP_INTERNAL
I5_ERR_PHP_EXECUTE	I5_ERR_PHP_NO_COMMAND
I5_ERR_PHP_EMPTY_ARRAY	I5_ERR_PHP_NO_KEYNAME
I5_ERR_PHP_NO_PARMNAME	I5_ERR_PHP_NO_ZVALUE
I5_ERR_PHP_COMMAND_ERROR	I5_ERR_PHP_DATAREA_READ
I5_ERR_PHP_GET_SYSVAL	I5_ERR_PHP_ELEMENT_MISSING
I5_ERR_PHP_BAD_DEF	I5_ERR_PHP_BAD_KEYNAME
I5_ERR_PHP_NO_DS_VALUE	I5_ERR_PARAMNOTFOUND
I5_ERR_ENDOFOCC	I5_ERR_PHP_BAD_DS_INPUT
I5_ERR_DESC_UNEXP	I5_ERR_DQDESC_UNSUPP
I5_ERR_DESC_WRONG_DATAOP	I5_ERR_INCORRECTVALUE
I5_ERR_PHP_BAD_PROG_NAME	I5_ERR_PHP_AS400_MESSAGE

I5_ERR_PHP_NOT_DTAQ_KEY	I5_ERR_PHP_DTAQ_BADKEY
I5_ERR_PHP_DESC_EMPTY	I5_ERR_PHP_BAD_LEN_PROP
I5_ERR_PHP_LIST_PROP	I5_ERR_PHP_SPOOL_FILE_FOPEN
I5_ERR_PHP_API_LENGTH	

Program Samples

i5 Program Call

The i5 program call process contains the following PHP functions:

- i5_connect
- i5_program_prepare
- i5_program_call
- i5_close

The sample PHP script below invokes an i5 program:

```
<?php
$conn = i5_connect($i5_server_ip, $i5_uname, $i5_pass);
if ($conn === false)
{
   print ("FAIL : Failed to connect to server : $i5_server_ip, with
user name : $i5_uname and password : $i5_pass <br>\n");
   $errorTab = i5_error();
   var_dump($errorTab);
   die();
}
/* Prepare File for execution */
$desc = array (
array ("name"=>"code", "io"=>I5_INOUT, "type" => I5_TYPE_CHAR,
"length"=> "10"),
array ("name"=>"name", "io"=>I5_INOUT, "type" => I5_TYPE_CHAR,
"length"=> "10"),
);
$prog = i5_program_prepare("EACDEMO/TESTSTP2", $desc);
if ($prog === FALSE)
{
   $errorTab = i5_error();
   echo "Program prepare failed <br>\n";
   var_dump($errorTab);
   die();
```

```
}
/* Execute Program */
$params = array ("code"=>" ","name"=>" ");
$retvals = array("code"=>"code","name"=>"name");
$ret = i5_program_call($prog, $params, $retvals) ;
echo "The return values are: <br>", "Name: ", $name, "<br> Code: ",
$code, "<br>";
if ($ret === FALSE)
{
   $errorTab = i5_error();
   echo "FAIL : i5_program_call failure code <br>";
   var_dump($errorTab);
   die();
}
$close_val = i5_program_close ($prog);
if ($close_val === false )
{
   print ("FAIL : i5_program_close returned fales, closing an open
prog.<br>\n");
   $errorTab = i5_error();
   var_dump($errorTab);
}
i5_close($conn) || print ("FAIL : Failed to disconnect from server
:$i5_server_ip");
?>
```

Service Program

```
<?php
// This program calls a service program (*SRVPGM), which was created
using the following i5/OS commands:
// CRTRPGMOD SRCFILE(EACDEMO/QRPGLESRC) SRCMBR(TESTMOD)
// CRTSRVPGM SRVPGM(EACDEMO/TESTSTRUC2) MODULE(EACDEMO/TESTMOD)
EXPORT(*ALL)
$Hdlcon = i5_connect($connect, $user, $pass, array(I5_OPTIONS_JOBNAME
=> "PHPAIX"));
if (is_bool($Hdlcon) && $Hdlcon == FALSE)
die(i5_errormsg());
echo "Connected!<BR>";
$desc = array (
array ("name"=>"code", "io"=>I5_INOUT, "type" => I5_TYPE_CHAR,
"length"=> "10"),
array ("name"=>"name", "io"=>I5_INOUT, "type" => I5_TYPE_CHAR,
"length"=> "10"),
);
$ret = $prog = i5_program_prepare("EACDEMO/TESTP2SRV(TESTSTMOD)",
$desc);
if (!$ret){
getError(I5_ERR_OK, -1);
} else {
echo "1. Prepare - It works! <br>";
}
$hdlPgm = $ret;
$parameter = array("code"=>" ", "name"=>" ");
$parmOut = array("code"=>"code", "name"=>"name");

$ret = i5_program_call($hdlPgm, $parameter, $parmOut);
if (!$ret){
getError(I5_ERR_OK, -1);
} else {
echo "2. Call - It works! <br>";
}
echo "code : $code<BR> name : $name<BR>";
?>
```

Data Retrieval

```
<?php
$Hdlcon = i5_connect($connect, $user, $pass);
if (!$Hdlcon) {
die(i5_errormsg());
}
$HdlFile = i5_open("eacdemo/sp_cust", I5_OPEN_READWRITE, $Hdlcon);
if (!is_bool($HdlFile))
{
echo "It works <br>\n";
}
$fealds = i5_list_fields($HdlFile);
$fetch_array = i5_fetch_array($HdlFile,I5_READ_FIRST);
$fetch_assoc = i5_fetch_assoc($HdlFile,I5_READ_NEXT);
$fetch_object = i5_fetch_object($HdlFile,I5_READ_PREV);
$fetch_row = i5_fetch_row($HdlFile,I5_READ_LAST);
print_r($fetch_array); echo"<br>\n <br>\n";
print_r($fetch_assoc); echo"<br>\n <br>\n";
print_r($fetch_object); echo"<br>\n <br>\n";
print_r($fetch_row); echo"<br>\n <br>\n";
$info = i5_info($HdlFile,1);
print_r($info); echo"<br>\n <br>\n";
$field_length = i5_field_len($HdlFile,1);
$field_name = i5_field_name($HdlFile,1);
$field_type = i5_field_type($HdlFile,1);
$field_scale = i5_field_scale($HdlFile,1);
echo "Field Name: {$field_name} <br>\n Field Lenght: {$field_length}
<br>\n Field Type: {$field_type} <br>\n Field Scale:
{$field_scale}<br>\n";
$list_fields = i5_list_fields($HdlFile);
print_r($list_fields);
$num_fields = i5_num_fields($HdlFile);
echo "<br>\n {$num_fields}";
$result = i5_result($HdlFile,2);
echo "<br>\n {$result}";
i5_close($Hdlcon);
?>
```

Native File Access sample

```
<?php
$conn = i5_connect($connect, $user, $pass);
if ($conn === false) die(i5_errormsg());
$file = i5_open("EACDEMO/SP_CUST", I5_OPEN_READWRITE, $conn);
if ($file === false) die(i5_errormsg());
i5_addnew($file);
i5_setvalue($file,array('11111', 'Kauai', 'Erica', 'Norm', '4-976
Hwy', 'Suite 103', 'Kapaa', 'HI', '94766', 'US', '808-555', '808-
555'));
i5_update($file);
i5_edit($file);
i5_delete($file);
i5_cancel_edit($file);
$tabf = array(1500);
i5_range_from($file, FALSE, $tabf);
$tabf = array(1600);
i5_range_to($file, FALSE, $tabf);
$fetch = i5_fetch_row($file, I5_READ_FIRST);
echo $fetch[0], " ",$fetch[1], " ", $fetch[2], "<br>";
$fetch = i5_fetch_row($file, I5_READ_NEXT);
echo $fetch[0], " ",$fetch[1], " ", $fetch[2], "<br>";
i5_range_clear($file);
$fetch = i5_fetch_row($file, I5_READ_FIRST);
echo $fetch[0], " ",$fetch[1], " ", $fetch[2], "<br><br>";
i5_data_seek($file, 2);
$rowTab = i5_fetch_row($file);
echo $rowTab[0], " ",$rowTab[1], " ", $rowTab[2], "<br>";
$tab=array(1510);
$seek = i5_seek($file, "=", $tab);
$rowTab = i5_fetch_row($file);
echo $rowTab[0], " ",$rowTab[1], " ", $rowTab[2], "<br>";
$id = i5_bookmark($file);
echo $id, "<br>";
i5_new_record($file, array('1229', 'Kauai Dive Shoppe ', 'Irica',
'Norman', '4-976 Sugarloaf Hwy', 'Suite 103', 'Kapaa Kauai', 'HI',
'94766-1234', 'US', '808-555-0269', '808-555-0278'));
i5_fetch_row($file,I5_READ_FIRST);
i5_update_record($file,array("FIRSTNAME"=>"Lina","LASTNAME"=>"Karasko
"));
```

```
i5_delete_record($file);
$keys = i5_get_keys($file);
echo $keys;
i5_free_file($file);
?>
```

Data Queues

Data Queue Without Key

```
<?php
$conn = i5_connect($i5_server_ip, $i5_uname, $i5_pass);
if (!$conn) {
  die(i5_errormsg());
}
$description = array("Name"=>"DATA", "Type"=>I5_TYPE_CHAR,
"Length"=>"50");
$queue = i5_dtaq_prepare("eacdemo/DTAQ_FIFO", $description);
$ret = i5_dtaq_send($queue,"","the dataqueue test data");
var_dump($ret); echo "<br>\n";
if(!$ret)  var_dump(i5_error());
$ret = i5_dtaq_receive($queue);
var_dump($ret);
i5_dtaq_close($queue);
i5_close($conn);
?>
```

Data Queue With key

```
<?php
$conn = i5_connect($i5_server_ip, $i5_uname, $i5_pass);
if (!$conn) {
  die(i5_errormsg());
}
$descriptionC = array("DSName"=>"PS", "DSParm"=>array(
array("Name"=>"PS1", "Type"=>I5_TYPE_CHAR, "Length"=>"10"),
array("Name"=>"PS2", "Type"=>I5_TYPE_PACKED, "Length"=>"10.4"),
array("Name"=>"PS3", "Type"=>I5_TYPE_CHAR, "Length"=>"10")
)
);
$dtaqHdl_KEY = i5_dtaq_prepare("EACDEMO/DTAQ_KEY", $descriptionC,10);
var_dump($dtaqHdl_KEY); echo "<br>\n";
$parameter = array("PS1"=>"test1", "PS2"=>13.1415, "PS3"=>"test2");
$key = "abcd";
$ret = i5_dtaq_send($dtaqHdl_KEY, $key, $parameter);
var_dump($ret); echo "<br>\n";
$ret = i5_dtaq_receive($dtaqHdl_KEY, "EQ", $key);
var_dump($ret);
i5_dtaq_close($dtaqHdl_KEY);
i5_close($conn);
?>
```

System Values

```
<?php
$conn = i5_connect($i5_server_ip, $i5_uname, $i5_pass);
print "Date is: ".i5_get_system_value("QDATE");
i5_close($conn);
?>
```

User Spaces

```
<?php
$conn = i5_connect($connect, $user, $pass);
if (!$Hdlcon) {
die(i5_errormsg());
}
$property = array(
I5_INITSIZE=>10,
I5_DESCRIPTION=>"Created by PHP",
I5_INIT_VALUE=>"A",
I5_EXTEND_ATTRIBUT=>"File",
I5_AUTHORITY=>"*ALL",
I5_LIBNAME=>"EACDEMO",
I5_NAME=>"USERSPACE"
);
$ret = i5_userspace_create($property);
if ($ret) echo "1. It works! <br>\n";
$description = Array(
array("Name"=>"filler0", "IO"=>I5_INOUT, "Type"=>I5_TYPE_CHAR,
"Length"=>"64"),
array("Name"=>"generic", "IO"=>I5_INOUT, "Type"=>I5_TYPE_LONG),
array("Name"=>"filler", "IO"=>I5_INOUT, "Type"=>I5_TYPE_CHAR,
"Length"=>"36"),
array("Name"=>"outputsize", "IO"=>I5_INOUT, "Type"=>I5_TYPE_LONG),
array("Name"=>"offsetInput", "IO"=>I5_INOUT, "Type"=>I5_TYPE_LONG)
);

$parameter = Array(
"filler0"=>"AAAA",
"generic"=>10,
"filler"=>"BBB",
"outputsize"=>100,
"offsetInput"=> 0
);

$parmOut = array("filler0"=>"filler0", "filler"=>"filler",
"generic"=>"generic", "outputsize"=>"outputsize",
"offsetInput"=>"offsetInput");
```

```
$UspcHdlBad = i5_userspace_prepare("EACDEMO/USERSPACE",
$description);
if ($UspcHdlBad) echo "2. It works! <br>\n";
$ret = i5_userspace_put($UspcHdlBad, $parameter);
if ($ret) echo "3. It works! <br>\n";
$ret = i5_userspace_get($UspcHdl, $parmOut);
if (!$ret) echo "4. It works! <br>\n";
if ($ret) echo "5. It works!";
var_dump($ret);*/
$ret = i5_command("DLTUSRSPC USRSPC(EACDEMO/USERSPACE)");
if ($ret) echo "6. It works!";
i5_close($conn);
?>
```

Active Job List

```
<?php
$Hdlcon = i5_connect($connect, $user, $pass, array(I5_OPTIONS_JOBNAME
=> "PHPAIX"));
if (is_bool($Hdlcon) && $Hdlcon == FALSE){
die(i5_errormsg());}

echo "i5_job_list: ";
$ret = i5_job_list();
if (!$ret)
die(i5_errormsg());
else
echo "It works!<br>";

$listHdl = $ret;
echo 'i5_job_list_read: ';
$ret = i5_job_list_read($listHdl);
if (!$ret)
die(i5_errormsg());
 else
echo "It works! <br>";

echo 'i5_job_list_close: ';
$ret = i5_job_list_close($listHdl);
if (!$ret) {
die(i5_errormsg());
```

```
} else {
echo "It works! <br>";
}

$listHdl = i5_job_list(array(I5_JOBNAME=>"*ALL", I5_JOBTYPE=>"S"));
if (is_bool($listHdl))
die(i5_errormsg());
else
echo "List <BR>";


$a= 0;
$ret = true;

while($ret && $a < 3){
echo "<p>Message $a<BR>";
$ret = i5_job_list_read($listHdl);
$a ++;
if (is_bool($ret))
die(i5_errormsg());
else
{
print_r($ret);echo "<p>";
echo "Job queue Name : " . $ret[I5_JOB_QUEUE_NAME] . ", Response Id :
" . $ret[I5_JOB_PROCESS_ID] . "<BR>";
echo "I5_JOB_JOBMSGQFL : " . $ret[I5_JOB_JOBMSGQFL] . "<BR>";
}
}
$ret = i5_job_list_close($listHdl);
?>
```

Job Log List

```
<?php
$Hdlcon = i5_connect($connect, $user, $pass, array(I5_OPTIONS_JOBNAME
=> "PHPAIX"));
if (is_bool($Hdlcon) && $Hdlcon == FALSE){
die(i5_errormsg());}

$listHdl = i5_jobLog_list();
if (is_bool($listHdl))
die(i5_errormsg());
 else
echo "List<BR>";

$a= 0;
$ret = true;

while($ret && $a < 2){
echo "Message $a<BR>";
$ret = i5_jobLog_list_read($listHdl);
$a ++;
if (is_bool($ret))
die(i5_errormsg());
else
{
print_r($ret);echo "<p>";
echo "Message : " . $ret[I5_LOBJ_MSG]. ",<BR> data : ".
$ret[I5_LOBJ_MSGDTA] . "<BR>";
}
}
$ret = i5_jobLog_list_close($listHdl);
if (is_bool($ret))
die(i5_errormsg());
 else {
print_r($ret);echo "<p>";
echo "Message : " . $ret[I5_LOBJ_MSG]. ",<BR> data : ".
$ret[I5_LOBJ_MSGDTA] . "<BR>";
?>
```

Data Areas

```
<?php
$Hdlcon = i5_connect($connect, $user, $pass);
if (!$Hdlcon) {
die(i5_errormsg());
}
$ret = i5_data_area_create("eacdemo/MYDTA", "50");
if ($ret) echo "1.It works! <br>";
$ret = i5_data_area_write("eacdemo/MYDTA", "coucou");
if ($ret) echo "3.It works! <br>";
$ret = i5_data_area_read("eacdemo/MYDTA", 2, 4);
if ($ret) echo "4.It works!: ", $ret, "<br>";
$ret = i5_data_area_read("eacdemo/MYDTA");
if ($ret) echo "5.It works!: ", $ret, "<br>";
$ret = i5_data_area_write("eacdemo/MYDTA", "lina", 5, 45);
if ($ret) echo "6.It works! <br>";
$ret = i5_data_area_read("eacdemo/MYDTA", 1, 5);
if ($ret) echo "7.It works!: ", $ret, "<br>";
$ret = i5_data_area_read("eacdemo/MYDTA");
if ($ret) echo "8.It works!: ", $ret, "<br>";
$ret = i5_data_area_delete("eacdemo/MYDTA");
if ($ret) echo "9.It works! <br>";
?>
```

Spooled Files

```
<?php
$conn = i5_connect($connect, $user, $pass);
if (!$conn) die("<br>Connect fail");
echo "<br>================<br>";
echo "<br>connected.";
$spool = i5_spool_list(array("username"=>"lina"),$conn);
if ($spool)
{
$count = 0;
while (($a = i5_spool_list_read($spool)) && ($count <= 2))
{
        echo "<br>================<br>";
        var_dump($a);
        echo "<br>data {$count}:<br>";
        $data = i5_spool_get_data($a['SPLFNAME'], $a['JOBNAME'],
                              $a['USERNAME'], $a['JOBNBR'],
                              $a['SPLFNBR']);
       if (is_bool($data)) var_dump(i5_error());
       var_dump($data);
       $count++;
   }

   i5_spool_list_close($spool);
}
else echo "No spool today.";
i5_close($conn);
XII. Object Listing
$conn = i5_connect($connect, $user, $pass);
if (!$conn) die("<br>Connect fail");
echo "<br>================<br>";
echo "<br>connected. <br>";
$objects = i5_objects_list("EACDEMO", "*ALL", "*PGM");
if ($objects) echo "1.It works! <br>";
$HdlObj = $objects;
$objects = i5_objects_list_read($HdlObj);
if (!is_bool($objects))
{
   echo "2.It works! <br>";
```

```
   print_r($objects); echo "<br>";
}
$continue = true;
$count = 0;
while($continue)
{
$objects = i5_objects_list_read($HdlObj);
if (is_bool($objects) && $objects == FALSE )
$continue = false;
else
{
   echo "3.It works! <br>";
   print_r($objects); echo "<br>";
}
$count ++;
if ($count == 2) break;
}
$objects = i5_objects_list_close($HdlObj);
if (is_bool($objects) && $objects == FALSE)
   $continue = false;
else
   echo "4.It works! <br>";
?>
```

PCML Program Call – PCML Description Used in the PHP program

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Program call example using PCML</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-
1">
<link href="style.css" rel="stylesheet" type="text/css">
<link rel="shortcut icon" type="image/x-icon" href="favicon.ico">
</head>
<body>
<?php
include("conn.php");
/* connection step  */
$Hdlcon = i5_connect($connect,$user, $pass,
array(I5_OPTIONS_JOBNAME=>"I5JOB"));
// This is the PCML taken from the following i5/OS command:
// CRTRPGMOD SRCFILE(EACDEMO/QRPGLESRC) SRCMBR(TESTSTRUC)
 PGMINFO(*PCML) INFOSTMF('/tmp/teststruc.pcml')
// we only defined some parameters as "output" to have minimal
network load
// we also changed the program name tag to specify the "path" value
for the program
// we also replace " with \" to have a correct PHP syntax. We also
could load it from the IFS.
$description = "<pcml version=\"4.0\">

   <!-- RPG module: TESTSTRUC  -->

   <!-- created: 2006-10-12-11.46.56 -->

   <!-- source: EACDEMO/QRPGLESRC(TESTSTRUC) -->

   <!-- 5 -->


   <struct name=\"S2\">
```

```
      <data name=\"ZOND2\" type=\"zoned\" length=\"10\"
precision=\"5\" usage=\"inherit\" />
      <data name=\"PACK2\" type=\"packed\" length=\"19\"
precision=\"5\" usage=\"inherit\" />
      <data name=\"PACK3\" type=\"packed\" length=\"19\"
precision=\"5\" usage=\"inherit\" />
      <data name=\"ALPH2\" type=\"char\" length=\"20\"
usage=\"inherit\" />
   </struct>


   <!-- 1 -->


   <struct name=\"S1\">

      <data name=\"ZOND\" type=\"zoned\" length=\"10\"
precision=\"5\" usage=\"inherit\" />
      <data name=\"PACK1\" type=\"packed\" length=\"19\"
precision=\"5\" usage=\"inherit\" />
      <data name=\"ALPH1\" type=\"char\" length=\"10\"
usage=\"inherit\" />
   </struct>


   <program name=\"TESTSTRUC\"
path=\"/QSYS.LIB/EACDEMO.LIB/TESTSTRUC.PGM\">
      <data name=\"CODE\" type=\"char\" length=\"10\"
usage=\"output\" />
      <data name=\"S1\" type=\"struct\" struct=\"S1\"
usage=\"inputoutput\" />
      <data name=\"S2\" type=\"struct\" struct=\"S2\"
usage=\"inputoutput\" />
      <data name=\"PACK\" type=\"packed\" length=\"1\"
precision=\"1\" usage=\"output\" />
      <data name=\"CH10\" type=\"char\" length=\"19\"
usage=\"output\" />
      <data name=\"CH11\" type=\"char\" length=\"20\"
usage=\"output\" />
      <data name=\"CH12\" type=\"char\" length=\"29\"
usage=\"output\" />
```

```
      <data name=\"CH13\" type=\"char\" length=\"33\"
usage=\"output\" />
   </program>


</pcml>


   ";
// define some input values
$pack3value=7789777.44;
$alph2value=4;
// now, prepare the program (only pcml parsing at this stage)
($hdlPgm = i5_program_prepare_PCML($description))
or trigger_error("Error while parsing PCML: " . i5_errormsg(),
E_USER_ERROR);
// let's define some input values
$in_parameters = Array(
"S1"=>Array("ZOND"=>54.77, "PACK1"=>16.2, "ALPH1"=>"MyValue"),
"S2"=>Array("ZOND2"=>44.66, "PACK2"=>24444.99945,
"PACK3"=>$pack3value, "ALPH2"=>$alph2value)
);
// now we need to define where to place output values; it will create
new local variables
$out_parameters = array(
"S1"=>"S1_Value", "S2"=>"S2_Value",
"CH10"=>"CH10_Value", "CH11"=>"CH11_Value", "CH12"=>"CH12_Value",
"CH13"=>"CH13_Value",
"CODE"=>"Code_Value", "PACK"=>"Pack"
);
// the call is made here
i5_program_call($hdlPgm, $in_parameters, $out_parameters)
or trigger_error("Error while executing program: " . i5_errormsg(),
E_USER_ERROR);
// all variables are now filled with program results.
echo "<br>S1:"; var_dump($S1_Value);
echo "<br>S2:"; var_dump($S2_Value);
echo "<br>CH10:"; var_dump($CH10_Value);
echo "<br>CH11:"; var_dump($CH11_Value);
echo "<br>CH12:"; var_dump($CH12_Value);
echo "<br>CH13:"; var_dump($CH13_Value);
```

```
echo "<br>Code:"; var_dump($Code_Value);
echo "<br>Pack:"; var_dump($Pack);

?>
</body>
</html>
```

PCML Program Call 2 – PCML File External to PHP Program

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Program call example using PCML</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-
1">
<link href="style.css" rel="stylesheet" type="text/css">
<link rel="shortcut icon" type="image/x-icon" href="favicon.ico">
</head>
<body>
<?php
include("conn.php");
/* connection step  */
$Hdlcon = i5_connect($connect,$user, $pass,
array(I5_OPTIONS_JOBNAME=>"I5JOB"));
// This is the PCML taken from the following i5/OS command:
// CRTRPGMOD SRCFILE(EACDEMO/QRPGLESRC) SRCMBR(TESTSTRUC)
 PGMINFO(*PCML) INFOSTMF('/tmp/teststruc.pcml')
// we only defined some parameters as "output" to have minimal
network load
// we also changed the program name tag to specify the "path" value
for the program
// we also replace " with \" to have a correct PHP syntax. We also
could load it from the IFS.
// the PCML file is located in the same location as the PHP program
($description =
file_get_contents("/www/zendcore/htdocs/teststruc.pcml"))
or trigger_error("Error while opening PCML file", E_USER_ERROR);
// define some input values
$pack3value=7789777.44;
$alph2value=4;
```

```
// now, prepare the program (only pcml parsing at this stage)
($hdlPgm = i5_program_prepare_PCML($description))
 or trigger_error("Error while parsing PCML: " . i5_errormsg(),
E_USER_ERROR);
// let's define some input values
$in_parameters = Array(
"S1"=>Array("ZOND"=>54.77, "PACK1"=>16.2, "ALPH1"=>"MyValue"),
"S2"=>Array("ZOND2"=>44.66, "PACK2"=>24444.99945,
"PACK3"=>$pack3value, "ALPH2"=>$alph2value)
);
// now we need to define where to place output values; it will create
new local variables
$out_parameters = array(
"S1"=>"S1_Value", "S2"=>"S2_Value",
"CH10"=>"CH10_Value", "CH11"=>"CH11_Value", "CH12"=>"CH12_Value",
"CH13"=>"CH13_Value",
"CODE"=>"Code_Value", "PACK"=>"Pack"
);
// the call is made here
i5_program_call($hdlPgm, $in_parameters, $out_parameters)
or trigger_error("Error while executing program: " . i5_errormsg(),
E_USER_ERROR);
// all variables are now filled with program results.
echo "<br>S1:"; var_dump($S1_Value);
echo "<br>S2:"; var_dump($S2_Value);
echo "<br>CH10:"; var_dump($CH10_Value);
echo "<br>CH11:"; var_dump($CH11_Value);
echo "<br>CH12:"; var_dump($CH12_Value);
echo "<br>CH13:"; var_dump($CH13_Value);
echo "<br>Code:"; var_dump($Code_Value);
echo "<br>Pack:"; var_dump($Pack);

?>
</body>
</html>
```

List of an RPG Program, "TESTSTRUC", Called by the PCML sample programs

```
D S1              DS
     D  ZOND                   1     10S 5
     D  pack1                  11    20P 5
     D  alph1                  21    30
     D S2              DS
     D  ZOND2                  1     10S 5
     D  pack2                  11    20P 5
     D  pack3                  21    30P 5
     D  alph2                  31    50
     D                SDS
     D  $PGM                    1      10
     D  $LIB                   81     90
     D  $JOB                   244   253
     D  $USER                  254   263
     D  $JOBNM                 264   269
     D INFDS           DS
     D  KEY                    369   369
     D  PAGRRN                 378   379B 0
     D*
     D F03             C                   CONST(X'33')
     C*----------------------------------------------------------------
-----
  ******************
     C     *ENTRY        PLIST
     C                   PARM                    CODE            10
     C                   PARM                    S1
     C                   PARM                    S2
     C                   PARM                    PACK             1
1
     C                   PARM                    ch10            19
     C                   PARM                    ch11            20
     C                   PARM                    ch12            29
     C                   PARM                    ch13            33
     **********************
     C                   MOVEL     'HELLO'       CODE
     C                   Z-SUB     0.5           PACK
     C                   eval      ch10 = *all'A'
     C                   eval      ch11 = *all'B'
```

```
     C                   eval      ch12 = *all'ZX'
     C                   eval      ch13 = *all'5'
     C                   eval      ZOND = 0 - zond
     C                   eval      pack1 = 0 - pack1
     C                   eval      alph1 ='alph1'
     C                   – service side eval      ZOND2 = 0 - zond2
     C                   eval      pack2 = 0-pack2
     C                   eval      pack3 =0 - pack3
     C                   eval      alph2 ='alph2'
     C*
     C                   SETON
                                       LR
     C                   RETURN
```

Web Services

```
<?php
/*
This service invokes a RPG program with two parameters
*/
class i5_program_service {

private $conn = false;
function __construct() {

        $this->conn = i5_connect('127.0.0.1', 'user', 'password'/*,
$connection_parameters*/);
        if (!is_resource($this->conn)) {
     throw new SoapFault('i5_program_service', 'Connection to i5
server failed, use i5_errormsg() to get the failure reason');
    }

}

public function service_for_i5_program($var_0, $var_1) {
        $description = Array (
        array ('Name' => 'code', 'IO' => I5_INOUT, 'Type' =>
I5_TYPE_CHAR, 'Length' => '10'),
        array ('Name' => 'name', 'IO' => I5_INOUT, 'Type' =>
I5_TYPE_CHAR, 'Length' => '10'));
```

```
        $prog = i5_program_prepare('eacdemo/teststp2', $description,
$this->conn);
        if (is_resource($prog)) {

         /* Execute Program */
            $params = array (
            'code' => $var_0, 'name' => $var_1);

            $retvals = array(
            'code' => 'ret_val_1', 'name' => 'ret_val_2');
            $ret = i5_program_call($prog, $params, $retvals) ;
            if ($ret === true) {
                 $ret = array($ret_val_1, $ret_val_2);
                 return $ret;
            }

            else {
                throw new SoapFault('i5_program_service', 'Failed to
call the program, use i5_errormsg() to get the failure reason');
            }

            if (!i5_program_close ($prog) ) {
                 throw new SoapFault('i5_program_service', 'Failed to
free program resource handle, use i5_errormsg() to get the failure
reason');
            }
        }

        else {
            throw new SoapFault('i5_program_service', 'Program
prepare failed, use i5_errormsg() to get the failure reason');
        }
}
    function __destruct() {

    if (!i5_close($this->conn)) {
            // Failed to disconnect from i5 server, use i5_errormsg()
to get the failure reason
        }
    }
}
```

```
ini_set('soap.wsdl_cache_enabled', '0');
$server = new SoapServer('zend.wsdl');
$server->setClass('i5_program_service');
$server->handle();
?>
```

Web Services – Client side

```
<?php
/*
This client calls the above service using ‘zend.wsdl’ XML file
generated by the WSDL Wizard in the Zend Studio.
/*
ini_set('soap.wsdl_cache_enabled', '0');
$my_client =  new SoapClient('zend.wsdl');
try{
var_dump($my_client->service_for_i5_program('111', ' '));
} catch (SoapFault $exception) {
    echo $exception;
  }

?
```

Technical Support

Online Help

Online Help is available from every page within the Zend Core Web Administration GUI by clicking the Help icon  located at the top-right corner of the screen.
Comments and suggestions are always welcome. To make suggestions on the Help, contact Zend's documentation department directly by sending an e-mail to documentation@zend.com. In addition, at the bottom of each Online Help page is a link for sending e-mails directly to the Zend Documentation Team.

Support Services

Zend's Support Center (http://www.zend.com/support) provides professional support and resources for Zend's solutions:

- Helpdesk - Submit a Helpdesk Ticket for assistance from the Zend Support Team, and get in touch with expert Zend engineers, ready to provide professional support with the highest priority and personal attention.
- Resources - Access a wealth of support information including Knowledgebase Articles and Frequently Asked Questions.

Consulting Services

Zend offers an array of Consulting Service Solutions to support the development and deployment of your business-critical PHP applications.
To find out more, please visit http://www.zend.com/en/services/consulting.

Education Services

Zend offers a variety of PHP Education Solutions including Certification and Training Courses, delivered online and through approved Training Partners.

Additional Zend Products and Services

Zend Technologies provide a complete lifecycle solution to develop, deploy and manage your business-critical PHP applications. Zend delivers the premier Web application platform, services and solutions for PHP applications running on Linux, UNIX, Windows and Macintosh systems.

Rapid Development & Deployment

Zend delivers an award-winning solution for organizations developing and deploying business-critical PHP applications. Starting with Zend Studio IDE, developers have the only PHP IDE (Integrated Development Environment) that encompasses all the development components necessary for the full PHP application lifecycle. Zend Platform then provides intelligence for the run-time aspects of the PHP application under development. A powerful integration between Zend Studio IDE and Zend Platform allows development and test teams to thoroughly check that code is free of run-time errors and potential performance bottlenecks before deployment.

In deployment, copyright protection and performance are key. Zend Guard secures your application from infringement by encoding and obfuscating the PHP code so you can maintain your intellectual property investment. Zend Platform provides maximum performance by accelerating your running PHP applications with four modules: Content Caching, File Compression, Code Acceleration and Download Serving. Requiring no code intervention, the modules typically increase execution speed by x3 – x100.

Zend's development and deployment solutions bring together your Development, QA, Production and IT teams. These will let you gain complete insight into your PHP applications with interactive alerting and XML-based messaging for integration with other tracking and support tools.

Reduced Testing Cycles

Zend provides the industry's only integrated IDE and run-time testing solution that reduces testing cycles by quickly identifying problems early in the lifecycle. Zend Studio and Zend Platform provide an integrated solution that can be deployed in development and testing environments to quickly identify run-time errors/anomalies and performance problems in PHP applications and database queries. Detailed reporting pinpoints the cause and provides the environmental data that contributed to each problem, eliminating the need to reproduce each error. Detailed reports can be shared between QA and development to speed up

the fixing process. In addition, alerts can be generated in both email and XML formats for integration with bug tracking systems.
Zend's testing cycle solution speeds up development and QA through early identification of run-time problems and with a level of detailed reporting required to fix bugs quickly and efficiently.

Central Monitoring and Management

Zend Platform provides the only PHP monitoring and management solution that actively reports run-time errors/anomalies on a deployed application. At a glance, Zend Platform provides an instant health status of your PHP servers and applications along with detailed reporting (with filtering) on a wide variety of application issues. Zend Platform can be configured to proactively alert you about the most critical of problems through email or XML-based messaging. Zend Platform also provides centralized management of your PHP configuration, ensuring that all PHP settings are configured correctly and consistently across groups of servers.

Improved Mean-Time-Between-Unscheduled-Interruption (MTBUI)

Zend's integrated solutions improve your application's MTBUI in several ways. Zend Platform actively monitors each production server, proactively alerting you to problems immediately. Detailed reporting pinpoints each problem with comprehensive information so that IT and development teams can immediately get to work on fixing the problem rather than trying to recreate it. Integration with Zend Studio provides instant access to source code, debugging and profiling information so that your teams can focus on eliminating the problem rather than spending time trying to gather information about the cause of the problem.

Proven Scalability

Zend's award-winning performance solution provides peace-of-mind about the scalability of your PHP application. Zend Platform's performance modules automatically improve the performance of each PHP application, enhancing the overall user experience and allowing you to serve more users without investing in new hardware.

Seamless Interoperability

Zend's solutions provide interoperability with other existing legacy or backend applications. Zend Platform provides seamless PHP/Java interoperability without the overhead of continual JVM (Java Virtual Machine) instantiation. The PHP/Java Bridge provides direct calls to Java code using natural coding syntax. The run-

time engine instantiates a persistent JVM once, providing minimal overhead and the performance necessary for composite PHP/Java applications.

Active Community / Support

Zend actively participates in the PHP community and provides Training, Consulting and Support. Zend's PHP Training Courses range from beginner and intermediate to advanced levels. The courses are live and take place within an online class environment, where students communicate and interact with the instructors in real-time. All of our instructors are Zend Certified PHP Experts, experienced in providing a professional and in-depth training experience. Zend also offers an array of Consulting Services to support the successful development and deployment of your business-critical PHP application projects.

Appendixes

Appendix A - Support Tool Information

The following information will be collected by the Zend Support Tool:

- access_log
- df.out
- error_log
- Files
- httpd.conf
- httpd.conf.orig
- httpd.pid
- httpd-autoindex.conf
- httpd-dav.conf
- httpd-default.conf
- httpd-info.con
- httpd-languages.conf
- httpd-manual.conf
- httpd-mpm.conf
- http-multilang-errordoc.conf
- http-ssl.conf
- http-userdir.conf
- http-vhosts.conf
- ls-lR.out
- magic
- mime types
- php.ini
- php_error_log
- ps.out
- registry.xml
- uname.out

Appendix B - PHP Configuration Information

The following PHP Configuration options are included in the Core version 2.5 installation packages:

Categories:

Data Handling	always_populate_raw_post_data	Always populate the $HTTP_RAW_POST_DATA variable.
	arg_separator.input	List of separator(s) used by php to parse input URLs into variables. Every character in this directive is considered as a separator.
	arg_separator.output	The separator used in php generated URLs to separate arguments.
	auto_append_file	Optionally defines the name of a file that is automatically parsed after the main php script file. **Note**: The file is included as if it were called by the php function include(), therefore the directive include_path is used and must be appropriately set. **Note**: The auto-append is not implemented if the php script file is terminated by the php function exit().
	auto_globals_jit	When enabled, the SERVER and ENV variables are created when they're first used (Just In Time) instead of when the script starts. If these variables are not used within a script, having this directive on will result in a performance

		gain. The php directives register_globals, register_long_arrays, and register_argc_argv must be disabled for this directive to be effective.
	auto_prepend_file	Optionally defines the name of a file that is automatically parsed before the main php script file. **Note**: The file is included as if it were called by the php function include(), therefore the directive include_path is used and must be appropriately set.
	default_charset	php outputs a default character set in the Content-type: header. To disable this output, set this directive to be an empty string ("").
	default_mimetype	MIME = Multipurpose Internet Mail Extensions. This directive specifies the protocol to be used for defining file attachments for the World Wide Web.
	magic_quotes_gpc	Enables/disables the Magic Quotes state for GPC (Get, Post, Cookie) operations. When this directive is enabled, all single quotes ('), double quotes ("), backslashes (\), and NULs are automatically (preceded by) a backslash. If magic_quotes_sybase is also enabled, a single quote is escaped with a single quote instead of with a backslash.
	magic_quotes_runtime	If this directive is enabled, most functions that return data from an external source, including databases (such as SQL), exec(), and text files, will have both single quotes (') and double quotes (") escaped with (preceded by) a backslash (\). If

		magic_quotes_sybase is also enabled, a single quote is escaped with a single quote instead of with a backslash.
	magic_quotes_sybase	Enables/disables the use of Sybase-style Magic Quotes: a single quote (') is escaped with (preceded by) a single quote instead of with a backslash (\). **Note**: This directive is enabled only if magic_quotes_gpc or magic_quotes_runtime is also enabled.
	post_max_size	Maximum size of POST data that php will accept.
	register_argc_argv	Specifies whether or not to instruct php to declare the variables argv and argc, which are used for holding the GET information. If you do not use these variables, disable this directive for increased performance.
	register_globals	Specifies whether or not to register the EGPCS variables (Environment, Get, Post, Cookie, and Server built-in) as global variables. If you do not want to clutter the global scope of your scripts with user data, turn off this directive. You will still be able to access the EGPCS variables by turning on the directive track_vars and then using the $HTTP_*_VARS[] arrays.
	register_long_arrays	Tells php whether or not to register the deprecated long $HTTP_*_VARS type predefined variables. When On (default), long predefined php variables like $HTTP_GET_VARS will be defined. If you're not using them, it's recommended to turn them off for performance reasons. Instead, use the superglobal arrays, like $_GET.

	variables_order	Specify the registration order for the GET, POST, Cookie, Environment and Server built-in variables (G, P, C, E, and S, respectively - often referred to as EGPCS and sometimes as GPC). Registration is done from left to right, and newer values override the older ones.
Error Handling and Logging	display_errors	Specifies whether or not php prints errors as part of the HTML script output. **Warning**: For production Web sites, it is strongly recommend to turn this feature Off and use error logging instead (see log_errors). Enabling display_errors on a production Web site can reveal security information to end users, such as file paths on your Web server, your database schema, and other sensitive information.
	display_startup_errors	Even when display_error is on, errors that occur during php's startup sequence are not displayed. It is strongly recommended to keep this option off, except for debugging purposes.
	docref_ext	Extensions for document reference. See docref_root. The value of docref_ext must begin with a dot '.'
	docref_root	The new error format contains a reference to a page describing the error or function causing the error. In case of manual pages you can download the manual in your language and set this ini directive to the URL of your local copy. If your local copy of the manual can be reached by '/manual/' you can simply

		use docref_root=/manual/. Additionally you have to set docref_ext to match the fileextensions of your copy docref_ext=.html.
	error_append_string	Specifies the string php outputs after an error message.
	error_log	Defines the file in which php errors should be logged. If the special value **syslog** is used, the errors are sent to the system logger. On UNIX this is syslog(3), on Windows it means the Event Log.
	error_prepend_string	Specifies the string php outputs before an error message.
	error_reporting	Specifies the types of php errors to be reported. This directive is a bit field whose value is composed by ORing the values for the individual error types. **Warning:** If you use the error-control operator prefix @ when calling a php expression (which turns off error reporting for that particular expression), then it is strongly recommend that you have the track_errors feature enabled. That way, if an error occurs during the evaluation of that expression, you can find the error message in the global variable $php_errormsg.
	html_errors	Turn off HTML tags in error messages. The new format for HTML errors produces clickable messages that direct the user to a page describing the error or function in causing the error. These references are affected by docref_root and docref_ext.
	ignore_repeated_errors	Do not log repeated messages. Repeated errors must occur in the same

		file on the same line until ignore_repeated_source is set to true.
	ignore_repeated_source	Ignore source of message when ignoring repeated messages. When this setting is On you will not log errors with repeated messages from different files or sourcelines.
	log_errors	Specifies whether or not php logs errors to a log file (server-specific log, stderr, or error_log). **Warning**: For production Web sites, it is strongly recommend using error logging instead of displaying errors; see display_errors.
	log_errors_max_len	Set the maximum length of log_errors in bytes.
	report_memleaks	If this parameter is set to Off, then memory leaks will not be shown (on stdout or in the log). This has only effect in a debug compile, and if error_reporting includes E_WARNING in the allowed list.
	report_zend_debug	Prints out descriptive bug messages in case of error in PHP development.
	track_errors	Specifies whether or not php stores the last error/warning message in $php_errormsg. **Warning**: If you use the error-control operator prefix @ when calling a php expression (which turns off error reporting for that particular expression), then it is strongly recommend that you have the track_errors feature enabled. That way, if an error occurrs during the evaluation of that expression, you can find the error message in the global variable $php_errormsg.

File Uploads	file_uploads	Specifies whether to allow HTTP file uploads.
	upload_max_filesize	Specifies the maximum file size in bytes that can be uploaded. Default is 2MB. **Note:** The MAX_FILE_SIZE item of the php file upload feature cannot specify a file size that is greater than the size set in this directive.
	upload_tmp_dir	Defines the temporary directory to use for storing files when doing HTTP file upload. If no directory is specified, the system default directory is used. **Note**: This directory must be writable by the user currently running php.
Fopen Wrappers	allow_url_fopen	This option enables the URL-aware fopen wrappers that enable accessing URL object like files. Default wrappers are provided for the access of remote files using the ftp or http protocol, some extensions like zlib may register additional wrappers. On Windows versions prior to php 4.3.0, the following functions do not support remote file accessing: include(), include_once(), require(), require_once() and the imagecreatefromXXX functions in the XLII, Image Functions extension.
	allow_url_include	This option allows the use of URL-aware fopen wrappers with the following functions: include(), include_once(), require(), require_once. This setting requires allow_url_fopen to be on.
Language Options	allow_call_time_ pass_reference	Enables/disables passing arguments by reference at function-call time. **Note:** This method of passing

		arguments by reference is not recommended, and future versions of php/Zend are likely not to support it.
	asp_tags	Enables/disables the use of ASP-like <% %> tags, in addition to the usual <?php ?> tags. Enabling this directive also enables the variable-value printing shorthand of the form <%=$value %>. For more information, see Escaping from HTML.
	engine	Turns php parsing on or off. This directive is really only useful in the Apache module version of php. It is used by sites that would like to turn php parsing on and off on a per-directory or per-virtual server basis. By putting engine off in the appropriate places in the httpd.conf file, php can be enabled or disabled.
	expose_php	Specifies whether php can expose the fact that it is installed on the server, for example, by adding its signature to the Web-server header. Exposing php is not a security threat in any way, but it does make it possible to determine that your server uses php. **Note**: This directive takes priority over the expose_launchpad directive.
	implicit_flush	Specifies whether or not to instruct php to tell the output layer to flush itself automatically after every output block. If this directive is set to ON, it is equivalent to calling the php function flush() after every call to print() and echo() and for every HTML block. **Warning**: This directive is generally recommended for debugging purposes

		only, since turning it on can seriously degrade performance.
	output_buffering	Enables/disables output buffering. Output buffering enables you to send header lines (including cookies) even after you have sent the body content, however, php's output layer will be slowed down a bit.
	output_handler	You can redirect all of the output of your scripts to a function. For example, if you set output_handler to mb_output_handler(), character encoding will be transparently converted to the specified encoding. Setting any output handler automatically turns on output buffering. **Note**: You cannot use both mb_output_handler() with ob_iconv_handler() and you cannot use both ob_gzhandler() and zlib.output_compression.
	precision	Specifies the number of significant digits displayed after the decimal point for floating point numbers. See also bcmath.scale in the Extensions tab.
	serialize_precision	Store serialize_precision significant digits after the floating point.
	short_open_tag	Enables/disables the use of the short form of the php opening tag (<? ?>). If this directive is disabled, you have to use the long form of the php opening tag (<?php ?>). The <script>...</script> tags, like the long form tag, are recognized regardless of the value of this directive. **Note**: php can be used in combination with XML only if this directive is

		disabled.
	unserialize_callback_func	The unserialize() callback function will called (with the undefined class' name as parameter), if the unserializer finds an undefined class which should be instanciated. A warning appears if the specified function is not defined, or if the function doesn't include/implement the missing class. Therefore, only set this entry if you want to implement such a callback-function.
	y2k_compliance	Specifies whether or not the php script should be made year-2000 compliant. **Warning**: Making the php script **Y2K compliant** (by setting this directive to On) will cause problems with non-Y2K-compliant browsers.
	zend.ze1_compatibility_mode	Enable compatibility mode with Zend Engine 1 (php 4). It affects the cloning, casting (objects with no properties cast to FALSE or 0), and comparing of objects. In this mode, objects are passed by value instead of reference by default.
- Colors for Syntax Highlighting mode	highlight.bg	Specifies color used for highlighting a background. You can supply a different value either in RGB format or as a standard color name.
	highlight.comment	Specifies the color used for highlighting a comment. You can supply a different value either in RGB format or as a standard color name.
	highlight.default	Specifies the color used for highlighting a default. You can supply a different value either in RGB format or as a standard color name.
	highlight.html	Specifies the color used for highlighting

		html text. You can supply a different value either in RGB format or as a standard color name.
	highlight.keyword	Specifies the color used for highlighting a keyword. You can supply a different value either in RGB format or as a standard color name.
	highlight.string	Specifies the color used for highlighting a string. You can supply a different value either in RGB format or as a standard color name.
- Safe Mode	disable_classes	This directive allows you to disable certain classes for security reasons. It takes on a comma-delimited list of class names. disable_classes is not affected by Safe Mode.
	disable_functions	Letting the user call certain functions may constitute a potential security breach. This directive specifies a comma-delimited list of functions names that are disabled for security reasons. This directive is not affected by whether Safe Mode is enabled or disabled. **Warning**: If this directive is empty, php will let the user call any function.
	open_basedir	Limits the files that can be opened by php to the specified directory-tree, including the file itself. This directive is NOT affected by whether Safe Mode is turned On or Off.
	safe_mode	Enables/disables Safe Mode. Enabling Safe Mode imposes several restrictions on what php can do, for example, files can be opened only if they are in the document root. **Note**: CGI users should always enable Safe Mode.

	safe_mode_exec_dir	Letting the user run certain programs may constitute a potential security breach. This directive contains a directory name, such as usr/local/bin. When php is in Safe Mode, the user can run only those programs located in the given directory. system() and other functions that execute system programs will refuse to run programs in other directories. **Warning**: If this directive is empty, php will let the user run any program.
	safe_mode_gid	By default, Safe Mode does a UID compare check when opening files. If you want to relax this to a GID compare, then turn on safe_mode_gid. This specifies whether to use UID (FALSE) or GID (TRUE) checking upon file access.
	safe_mode_include_dirs	UID/GID checks are bypassed when including files from this directory and its subdirectories (directory must also be in include_path or full path must including). As of php 4.2.0, this directive can take a colon (semi-colon on Windows) separated path in a fashion similar to the include_path directive, rather than just a single directory.
Mail	mail.force_extra_parameters	Forces the addition of the specified parameters to be passed as extra parameters to the sendmail binary. These parameters will always replace the value of the 5th parameter to mail(), even in safe mode.
	sendmail_from	Specifies which "From:" mail address should be used in mail sent from php

		under Windows. This directive also sets the "Return-Path:" header.
	sendmail_path	Where the sendmail program can be found, usually /usr/sbin/sendmail or /usr/lib/sendmail. configure does an honest attempt of locating this one for you and set a default, but if it fails, you can set it here. Systems not using sendmail should set this directive to the sendmail wrapper/replacement their mail system offers, if any.
	SMTP	Used under Windows only. Specifies the host name or IP address of the SMTP server php which should be used for mail sent with the mail() function.
	smtp_port	Used under Windows only: Number of the port to connect to the server specified with the SMTP setting when sending mail with mail(); defaults to 25.
Misc.	browscap	Defines the name of the browser capabilities file. **Note:** For a description of the settings in the browser capabilities file itself, see the php function get_browser().
	ignore_user_abort	If false, scripts will be terminated as soon as they try to output something after a client has aborted their connection.
Paths and Directories	doc_root	Specifies the php root directory on the server. If php is configured in Safe Mode (the directive safe_mode is enabled), no files outside this directory are served. **Note**: This directive is used only if it is not empty.
	enable_dl	Specifies whether or not to enable the php function dl(), which loads php

		extensions at run time. **Note**: The function dl() does not work correctly under multithreaded servers, such as IIS or Zeus, and is automatically disabled on these servers.
	extension_dir	Defines the directory in which php should look for dynamically loadable extensions (modules).
	include_path	Defines a list of directories where the following functions will search for files: require(), include(), and fopen_with_path(). The format is same as the system's PATH environment variable: a list of directories separated by colons (:) in UNIX or by semicolons (;) in Windows. The default value for this directive is an empty string (""). Only the current directory will be searched.
	user_dir	Defines the base name of the directory used on a user's home directory for php files, for example, public_html. This is the directory under which php opens the script using /~username. **Note**: This directive is used only if it is not empty.
Resource Limits	max_execution_time	Specifies the maximum time in seconds that one script is allowed to run before it is terminated by the parser. This helps prevent poorly written scripts from tying up the server. The present default value is 30 seconds.
	max_input_nesting_level	Limits the nesting level of input variables.
	max_input_time	Sets the maximum time in seconds a script is allowed to receive input data, like POST, GET and file uploads. The

		present default value is 60 seconds.
	memory_limit	Specifies the maximum amount of memory in bytes that one script is allowed to allocate. This helps prevent poorly written scripts from tying up all the available memory on a server. The present default value is 128MB.
	realpath_cache_size	Determines the size of the realpath cache to be used by php. This value should be increased on systems where php opens many files, to reflect the quantity of the file operations performed.
	realpath_cache_tll	Duration of time (in seconds) for which to cache realpath information for a given file or directory. For systems with rarely changing files, consider increasing the value.

Appendix C - Zend Core Extensions

The following PHP Extensions are included in the Core version 2.5 installation package:

Legend:

- Extension installed but disabled by default.

+ Extension installed and enabled by default.

Extension Name	Description	Status
bcmath	**Arbitrary Precision Mathematics** - PHP offers the Binary Calculator which supports numbers of any size and precision, represented as strings.	+
bz2	**Bzip2 Compression** - The bzip2 functions are used to transparently read and write bzip2 (.bz2) compressed files.	-
calendar	**Calendar Conversions** - The calendar extension presents a series of functions to simplify converting between different calendar formats. The intermediary or standard it is based on is the Julian Day Count.	-
ctype	**Character Classifications** - Checks whether a character or string falls into a certain character class according to the current locale.	+
curl	**Client URL Library Functions -** Allows you to connect to and communicate with many different types of servers with many different types of protocols. **Note:** curlib libraries must be installed in order for this extension to function.	+
date	**Date Module -** Allows you to get the date from the server where your PHP scripts are running. You can use this function to format the date in many different ways.	+
dom	**DOM XML-** The DOM extension is the replacement for the DOM XML extension from PHP 4. The extension still contains many old functions, but they should no longer	+

	be used. In particular, functions that are not object-oriented should be avoided. The extension allows you to operate on an XML document with the DOM API.	
exif	**Exchangeable Image File Format** Data **-** With the exif extension you are able to work with image meta data.	-
ftp	**FTP Client -** The functions in this extension implement client access to file servers speaking the File Transfer Protocol (FTP). This extension is meant for detailed access to an FTP server providing a wide range of control to the executing script.	-
gd	**GD (Image Manipulation) -** With the GD library of image functions PHP can be used to create and manipulate image files in a variety of different image formats, including gif, png, jpg, wbmp, and xpm. Even more convenient, PHP can output image streams directly to a browser.	+
gettext		-
gmp	**GNU MP Library (Arbitrary Length Integers) -** These functions allow you to work with arbitrary-length integers using the GNU MP library. **Note:** GMP libraries must be installed in order for this extension to function.	-
i5com	The programming interface that is used to program the business logic.	+
ibm_db2	**IBM DB2 Database Access -** These functions enable you to access IBM DB2 Universal Database, IBM Cloudscape, and Apache Derby databases using the DB2 Call Level Interface (DB2 CLI).	+
iconv	**Character Set Conversion** - This module contains an interface to iconv character set conversion facility. With this module, you can turn a string represented by a local character set into the one represented by another character set, which may be the Unicode character set.	+

imap	**IMAP, POP3 and NNTP -** These functions are not limited to the IMAP protocol, despite their name. The underlying c-client library also supports NNTP, POP3 and local mailbox access methods.	+
JSON	Implements the **JavaScript Object Notation (JSON)** data-interchange format. The decoding is handled by a parser based on the JSON_checker by Douglas Crockford.	+
ldap	**OpenLDAP -** LDAP is the Lightweight Directory Access Protocol, and is a protocol used to access "Directory Servers". The Directory is a special kind of database that holds information in a tree structure. **Note:** LDAP libraries must be installed in order for this extension to function	-
mbstring	**Multibyte Character Processing -** Multibyte character encoding schemes were developed to express more than 256 characters in the regular bytewise coding system. When you manipulate (trim, split, splice, etc.) strings encoded in a multibyte encoding, you need to use special functions since two or more consecutive bytes may represent a single character in such encoding schemes.	+
mcrypt	**MCrypt -** This is an interface to the mcrypt library, which supports a wide variety of block algorithms such as DES, TripleDES, Blowfish (default), 3-WAY, SAFER-SK64, SAFER-SK128, TWOFISH, TEA, RC2 and GOST in CBC, OFB, CFB and ECB cipher modes. Additionally, it supports RC6 and IDEA which are considered "non-free". **Note:** Libmcrypt libraries must be installed in order for this extension to function.	-
mhash	**Hash Algorithms -** Mhash can be used to create checksums, message digests, message authentication codes, and more.	-

	Note: Libmhash libraries must be installed in order for this extension to function.	
ming	**Ming Functions for Flash** - An open-source (LGPL) library which allows you to create SWF ("Flash") format movies. **Warning:** This extension is experimental. The behaviour of this extension may change without notice in a future release of PHP. Use this extension at your own risk. **Note:** Ming libraries must be installed in order for this extension to function.	-
mssql	Allows you to access MS SQL database servers. **Note:** FreeTDS libraries must be installed in order for this extension to function. In addition, the following steps must be performed to allow Zend Core to connect to the MSSQL database server: 1. Open the freetds.conf file, located in /usr/local/Zend/Core/etc/ 2. Under the [SQLserver_definition] section, change the 'host' parameter to your SQL server address, (e.g. host = 10.1.3.12) and ensure the 'tds version' parameter matches the TDS version on your system.	+
mysql	**MySQL** - Allows you to access MySQL database servers. **Note:** MySQL libraries must be installed in order for this extension to function	+
mysqli	**MySQL Improved** - Allows you to access the functionality provided by MySQL 4.1 and above. **Note:** MySQL libraries must be installed in order for this extension to function	+

openssl	**OpenSSL** - This module uses the functions of >>OpenSSL for generation and verification of signatures and for sealing (encrypting) and opening (decrypting) data. **Note:** OpenSSL libraries must be installed in order for this extension to function	+
pcntl	**Process Control Functions -** Process Control support in PHP implements the Unix style of process creation, program execution, signal handling and process termination. Process Control should not be enabled within a Webserver environment and unexpected results may happen if any Process Control functions are used within a webserver environment.	+
pcre	**Perl Compatible Regular Expressions -** The syntax for patterns used in these functions closely resembles Perl.	+
pdo	**Base PDO (PHP Data Objects) Driver -** The PHP Data Objects (PDO) extension defines a lightweight, consistent interface for accessing databases in PHP. Each database driver that implements the PDO interface can expose database-specific features as regular extension functions.	+
pdo_mysql	Allows access to MySQL 3.x/4.0 databases	+
posix	**POSIX -** Contains an interface to those functions defined in the IEEE 1003.1 (POSIX.1) standards document which are not accessible through other means. **Warning**: Sensitive data can be retrieved with the POSIX functions, e.g. posix_getpwnam() and friends. None of the POSIX functions perform any kind of access checking when safe mode is enabled. It's therefore strongly advised to disable the POSIX extension.	+
reflection	**PHP Reflection Support -** PHP 5 comes with a complete reflection API that adds the ability to reverse-engineer classes, interfaces, functions and	+

	methods as well as extensions. The reflection API also offers ways of retrieving doc comments for functions, classes and methods.	
session	**Session Management -** Session support in PHP consists of a way to preserve certain data across subsequent accesses.	+
shmop	**Shared Memory -** Shmop is an easy-to-use set of functions that allows PHP to read, write, create and delete Unix shared memory segments.	-
simplexml	**SimpleXML** - The SimpleXML extension provides a very simple and easily usable toolset to convert XML to an object that can be processed with normal property selectors and array iterators. **Note:** libxml libraries must be installed in order for this extension to function	+
soap	**SOAP** - The SOAP extension can be used to write SOAP Servers and Clients. **Note:** libxml libraries must be installed in order for this extension to function	+
sockets	**Socket Communications -** The socket extension implements a low-level interface to the socket communication functions based on the popular BSD sockets, providing the possibility to act as a socket server as well as a client.	+
spl	**Standard PHP Library** - SPL is a collection of interfaces and classes that are meant to solve standard problems.	+
sqlite	**SQLite** - This is an extension for the SQLite Embeddable SQL Database Engine. SQLite is a C library that implements an embeddable SQL database engine. Programs that link with the SQLite library can have SQL database access without running a separate RDBMS process.	-
sysvmsg	Enables **System V messages support -** The messaging functions may be used to send and receive	-

	messages to/from other processes. They provide a simple and effective means of exchanging data between processes, without the need for setting up an alternative using Unix domain sockets.	
sysvsem	Enables **System V semaphore support** - Semaphores may be used to provide exclusive access to resources on the current machine, or to limit the number of processes that may simultaneously use a resource.	-
sysvshm	Enables **System V shared memory support -** Shared memory may be used to provide access to global variables.	-
standard	**Standard PHP functions**	+
tidy	**Tidy HTML Clean and Repair** - Tidy is a binding for the Tidy HTML clean and repair utility which allows you to not only clean and otherwise manipulate HTML documents, but also traverse the document tree. **Note:** libtidy libraries must be installed in order for this extension to function	-
tokenizer	**Interface to Zend Engine's PHP Scanner** - The tokenizer functions provide an interface to the PHP tokenizer embedded in the Zend Engine. Using these functions you may write your own PHP source analyzing or modification tools without having to deal with the language specification at the lexical level.	-
wddx	**WDDX (Web Distributed Data Exchange)** - These functions are intended for work with >>WDDX **Note:** expat (Apache) libraries must be installed in order for this extension to function	+
xml	**SAX XML** - XML (eXtensible Markup Language) is a data format for structured document interchange on the Web. This toolkit lets you parse, but not validate, XML documents. This extension lets you create XML parsers and then define handlers for different XML events.	+

xmlreader	**XML Reader** - The XMLReader extension is an XML Pull parser. The reader acts as a cursor going forward on the document stream and stopping at each node on the way. **Note:** libxml libraries must be installed in order for this extension to function	-
xmlwriter	**XML Writer -** Represents a writer that provides a non-cached, forward-only means of generating streams or files containing XML data. This extension can be used in an object oriented style or a procedural one. Every method documented describes the alternative procedural call.	+
xsl	**XSL Transformations** - The XSL extension implements the XSL standard, performing XSLT transformations using the libxslt library. **Note:** libxslt libraries must be installed in order for this extension to function	-
zip	**ZIP Archives** - Enables you to transparently read ZIP compressed archives and the files inside them. **Note:** Zlib libraries must be installed in order for this extension to function	-
zlib	**zlib Compression (Incl. gzip)** - Enables you to transparently read and write gzip (.gz) compressed files, through versions of most of the file system functions which work with gzip-compressed files (and uncompressed files without sockets).	+

Note:

Some extensions have dependencies on certain libraries.

For a full list of libraries installed with Zend Core, see Appendix D - Libraries.

To uninstall extensions / Zend Core components:

1. Open the Zend Core Setup Tool by running the command "go zendcore/zcmenu" in your i5/OS emulation screen.
2. Select Option 2 - Update via Zend Network menu, then Option 6 - Remove Zend Core Components.
   A list of erasable components will be displayed.
   Extensions will be prefixed with a "/ext".
3. Press Page Up or Page Down to scroll through the list.
4. Select which libraries to delete by marking an X next to the required library and pressing Enter.
5. Press F8 to remove any selected extensions.
6. Restart the web server in order for your changes to take effect.
   To restart your web server:
   In the Zend Core Setup Tool main menu, select Option 5 - Service Management menu and then select Option 6 - Restart Apache server instance.

Appendix D - Libraries

The following libraries are included in the Core version 2.5 installation package:

Library	Description
libcurl	A client-side URL transfer library supporting FTP, FTPS, HTTP, HTTPS, SCP, SFTP, TFTP, TELNET, DICT, FILE and LDAP. See http://curl.haxx.se for more information.
GMP	GMP is a library for arbitrary precision arithmetics, operating on signed integers, rational numbers, and floating point numbers. See http://gmplib.org for more information.
LDAP client libraries	The Lightweight Directory Access Protocol (LDAP) libraries provide access to X.500 directory services. See http://www.openldap.org for more information.
libmcrypt	Libmcrypt is a companion to Mcrypt. It contains the encryption functions and provides a standardized mechanism for accessing them. See http://sourceforge.net/projects/mcrypt for more information.
Libmhash	Libmhash provides a uniform interface to a large number of hash algorithms. These algorithms can be used to compute checksums, message digests, and other signatures. See http://mhash.sourceforge.net for more information.
Ming Libraries	Ming is an SWF ("Flash") file format output library. It is written in C, with wrappers for C++, Python, and PHP, plus rudimentary support for Ruby and Perl. See http://sourceforge.net/projects/ming for more information.
OpenSSL	OpenSSL is a library that provides cryptographic functionality to applications such as secure web servers. See http://www.openssl.org for more information.

t1	t1lib is a library written in C which implements functions for generating bitmaps from Adobe Type 1 fonts. For more information see: http://gnuwin32.sourceforge.net/packages/t1lib.htm
libxml	Libxml2 is the XML C parser and toolkit developed for the Gnome project (but usable outside of the Gnome platform). See http://www.xmlsoft.org for more information.
easycom	
libtidy	TLibTidy is a Pascal wrapper for the library version of the HTML Tidy program. See http://tidy.sourceforge.net for more information.
libxslt	Libxslt is the XSLT C library developed for the GNOME project. XSLT itself is an XML language to define transformation for XML. See http://xmlsoft.org/XSLT for more information.
MySQL Libraries	A collection of MySQL Libraries. See http://www.mysql.com for more information.
PostgresSQL Libraries	Included with its standard function library are hundreds of built-in functions that range from basic math and string operations to cryptography and Oracle compatibility. See http://www.postgresql.org for more information.
IBM DB2 Client Libraries	Provide connectivity to IBM DB2 databases. See http://www.ibm.com/db2 for more information.
libtds	Allows for connection to FreeTDS libraries. See http://www.freetds.org/reference/a00268.html for more information.

Note:
The listed libraries will be installed by default with Zend Core

To uninstall libraries:

7. Open the Zend Core Setup Tool by running the command "go zendcore/zcmenu" in your i5/OS emulation screen.
8. Select Option 2 - Update via Zend Network menu, then Option 6 - Remove Zend Core Components.
   A list of erasable components will be displayed.
   Libraries will be prefixed with a "/lib".
9. Press Page Up or Page Down to scroll through the list.
10. Select which libraries to delete by marking an X next to the required library and pressing Enter.
11. Press F8 to remove any selected libraries.
12. Restart the web server in order for your changes to take effect.
    To restart your web server:
    In the Zend Core Setup Tool main menu, select Option 5 - Service Management menu and then select Option 6 - Restart Apache server instance.

Appendix E - Misc. Directives Configuration Information

The following Misc. Directives can be configured from Zend Core:

Categories:

dbx - Database Abstraction Layer
Informix
Ingres II
mSQL
PostgressSQL
SQL
Sybase
Sybase-CT
Syslog
Verisign Payflow Pro

Other ini File Directives

dbx - database Abstract Layer	dbx.colnames_case	Specifies whether to return column names unchanged or converted to uppercase or lowercase.
Informix	ifx.allow_persistent	Specifies whether or not to allow persistent Informix connections.
	ifx.blobinfile	Set this directive to True if you want to return Informix BLOB columns in a file, or to False if you want to keep them in memory. You can override the setting of this directive at run time with the PHP function ifx_blobinfile_mode().
	ifx.byteasvarchar	Set this directive to True if you want to return Informix BYTE columns as normal strings in select statements, or to False if you want to use blob id parameters. You can override the setting of this directive at run time with the PHP function ifx_byteasvarchar().
	ifx.charasvarchar	Enables/disables trimming the trailing spaces from Informix CHAR columns when they are fetched. **Note**: Enabling this directive can be very helpful to Informix SE users.
	ifx.default_host	Defines the Informix default host to connect to if the default host is not defined in either ifx_connect() or

		ifx_pconnect(). Note: When PHP is in Safe Mode, this directive is not used.
	ifx.default_password	Defines the Informix default password to use if the default password is not defined in either ifx_connect() or ifx_pconnect(). Note: When PHP is in Safe Mode, this directive is not used.
	ifx.default_user	Defines the Informix default user ID to use if the default user ID is not defined in either ifx_connect() or ifx_pconnect(). Note: When PHP is in Safe Mode, this directive is not used.
	ifx.max_links	Specifies the maximum number of all Informix connections per process, including persistent connections. -1 means no limit.
	ifx.max_persistent	Specifies the maximum number of persistent Informix connections per process. -1 means no limit.
	ifx.nullformat	Set this directive to True if you want Informix NULL columns returned as the literal string "NULL", or to False if you want them returned as the empty string "". You can override the setting of this directive at run time with the PHP function ifx_ nullformat().
	ifx.textasvarchar	Set this directive to On if you want to return Informix TEXT columns as normal strings in select statements, or to False if you want to use blob id parameters. You can override the setting of this directive at run time with the PHP function ifx_textasvarchar().
Ingres II	ingres.allow_persistent	Specifies whether or not to allow persistent Ingres II connections.
	ingres.default_database	Defines the Ingres II default database, where

		the format is [node_id::]dbname[/srv_class]. **Note**: When PHP is in Safe Mode, this directive is not used.
	ingres.default_password	Defines the Ingres II default password. **Note**: When PHP is in Safe Mode, this directive is not used.
	ingres.default_user	Defines the Ingres II default user. **Note**: When PHP is in Safe Mode, this directive is not used.
	ingres.max_links	Specifies the maximum number of all Ingres II connections per process, including persistent connections. -1 means no limit.
	ingres.max_persistent	Specifies the maximum number of persistent Ingres II connections per process. -1 means no limit.
mSQL	msql.allow_persistent	Specifies whether or not to allow persistent mSQL connections.
	msql.max_links	Specifies the maximum number of all mSQL connections per process, including persistent connections. -1 means no limit.
	msql.max_persistent	Specifies the maximum number of persistent mSQL connections per process. -1 means no limit.
PostgressSQL	pgsql.allow_persistent	Specifies whether or not to allow persistent PostgresSQL connections.
	pgsql.auto_reset_persistent	Detect broken persistent links with pg_pconnect(). Needs a little overhead.
	pgsql.ignore_notice	Whether or not to ignore PostgreSQL backend notices.
	pgsql.log_notice	Whether or not to log PostgreSQL backends notice messages. The PHP directive pgsql.ignore_notice must be off in order to log notice messages.
	pgsql.max_links	Specifies the maximum number of all

		PostgresSQL connections per process, including persistent connections. -1 means no limit.
	pgsql.max_persistent	Specifies the maximum number of persistent PostgresSQL connections per process. -1 means no limit.
SQL	sql.safe_mode	If the SQL Safe Mode option is enabled the MySQL and Ingres extensions will ignore the supplied host, user and password information and will use only the default ones.
Sybase	sybase.allow_persisten t	Specifies whether or not to allow persistent Sybase connections.
	sybase.compatability_ mode	Specifies compatibility mode with the older versions of PHP 3.0. If this directive is on, it will cause PHP to automatically assign types to results according to their Sybase type, instead of treating all results as strings. Note: This compatibility mode probably will not stay around forever, so try making the necessary changes to your code and turning off this directive.
	sybase.max_links	Specifies the maximum number of all Sybase connections per process, including persistent connections. -1 means no limit.
	sybase.max_persistent	Specifies the maximum number of persistent Sybase connections per process. -1 means no limit.
	sybase.min_error_seve rity	Specifies the minimum Sybase error severity that PHP displays. Errors that have a severity that is lower than the value of this directive are not displayed.
	sybase.min_message_ severity	Specifies the minimum Sybase message severity that PHP displays. Messages

		that have a severity that is lower than the value of this directive are not displayed.
Sybase-CT	sybct.allow_persistent	Specifies whether or not to allow persistent Sybase-CT connections.
	sybct.max_links	Specifies the maximum number of all Sybase-CT connections per process, including persistent connections. -1 means no limit.
	sybct.max_persistent	Specifies the maximum number of persistent Sybase-CT connections per process. -1 means no limit.
	sybct.min_client_severity	Specifies the minimum Sybase-CT client message severity to display; client messages having a severity that is lower than the value of this directive are not displayed.
	sybct.min_server_severity	Specifies the minimum Sybase-CT server message severity to display; server messages having a severity that is lower than the value of this directive are not displayed.
Syslog	define_syslog_variables	Specifies whether or not to define the syslog variables, such as $LOG_PID, $LOG_CRON, etc. Performance is improved by setting this directive to 0. At run time you can always define the syslog variables by calling the PHP function define_syslog_variables().
Verisign Payflow Pro	pfpro.defaulthost	Defines the Verisign Payflow Pro default host that PHP connects to, i.e. the default Signio server. **Note**: For processing live transactions you cannot use the default value as it stands. One reasonable alternative is to change the default to be connect.signio.com.

		Note: When PHP is in Safe Mode, this directive is not used.
	pfpro.defaultport	Defines the Verisign Payflow Pro default port that PHP connects to. **Note:** When PHP is in Safe Mode, this directive is not used.
	pfpro.defaulttimeout	Specifies the Verisign Payflow Pro default timeout, in seconds. **Note:** The timeout countdown appears to begin only after a link to the processor has been established, therefore, in the event of DNS or network problems, your script could continue for a long period of time.

Appendix F - I5 Toolkit Templates

Zend Studio IDE for i5 comes complete with the following code templates containing i5 PHP API Toolkit functions:

I5 Template	Explanation
i5ActiveJobs	Enables retrieving the system's active jobs, it: 1. Connects to i5 server 2. Opens active job list 3. Gets array for an active job entry 4. Closes handle received from i5_job_list function 5. Closes connection to i5 server
i5Connect	Enables connecting to the i5 server, it: 1. Connects to i5 server 2. Closes connection to i5 server
i5DataAreaCreate	Creates the data area, it: 1. Connects to i5 server 2. Creates data area of given size 3. Closes connection to i5 server
i5DataAreaDelete	Enables deleting the data area, it: 1. Connects to i5 server 2. Deletes data area 3. Closes connection to i5 server
i5DataAreaRead	Enables reading from a data area, it: 1. Connects to i5 server 2. Reads from data area 3. Closes connection to i5 server
i5DataAreaWrite	Enables reading from a data area, it: 1. Connects to i5 server 2. Reads from the data area 3. Closes connection to i5 server
i5DtaqReceive	Enables reading data from the data queue without key, it: 1. Connects to i5 server 2. Reads data from the data queue without key 3. Closes connection to i5 server
i5DtaqReceiveKey	Enables reading data from the data queue with

	key, it: 1. Connects to i5 server 2. Reads data from the data queue with key 3. Closes connection to i5 server
i5DtaqSend	Enables putting data to the data queue without key, it: 1. Connects to i5 server 2. Puts data to the data queue without key 3. Closes connection to i5 server
i5DtaqSendKey	Enables putting data into the data queue without a key, it 1. Connects to i5 server 2. Puts data to the data queue without key 3. Closes connection to i5 server
i5JobLogs	Enables retrieving job log entries, it: 1. Connects to i5 server 2. Opens job log 3. Gets array for a job log entry 4. Closes handle received from i5_jobLog_list function 5. Closes connection to i5 server
i5ObjectListing	Enables getting an array with the message element for an object list entry, it: 1. Connects to i5 server 2. Opens object list 3. Gets for a object list entry 4. Closes handle received from i5_objects_list function 5. Closes connection to i5 server
i5Program	Enables calling a program and accept results from it, it: 1. Connects to i5 server 2. Opens a program or service procedure and prepares it to be run 3. Calls the program and optionally accepts results 4. Free program resource handle

	5. Closes connection to i5 server
i5ProgramService	Creates Web Services class enabling invoking an RPG program, it: 1. Connects to i5 server 2. Opens a program or service procedure and prepares it to be run 3. Calls the program and optionally accepts results 4. Free program resource handle 5. Closes connection to i5 server
i5Spool	Enables getting spool file data from the queue and getting the data from the spool file, it: 1. Connects to i5 server 2. Creates an pool file lists, of certain output queue or for all queues 3. Gets spool file data from the queue 4. Get the data from the spool file 5. Free spool list resource 6. Closes connection to i5 server
i5UserSpaceCreate	Creates a new user space object, it: 1. Connects to i5 server 2. Creates new user space object 3. Closes connection to i5 server
i5UserSpaceDelete	Enables deleting a user space object, it: 1. Connects to i5 server 2. Deletes user space object 3. Closes connection to i5 server
i5UserSpaceGet	Retrieves user space data, it: 1. Connects to i5 server 2. Opens a user space and prepares it to be run 3. Retrieves user space data 4. Closes connection to i5 server
i5UserSpacePut	Enables to add user space data, it: 1. Connects to i5 server 2. Opens a user space and prepares it to be run 3. Adds user space data 4. Closes connection to i5 server

Index

T