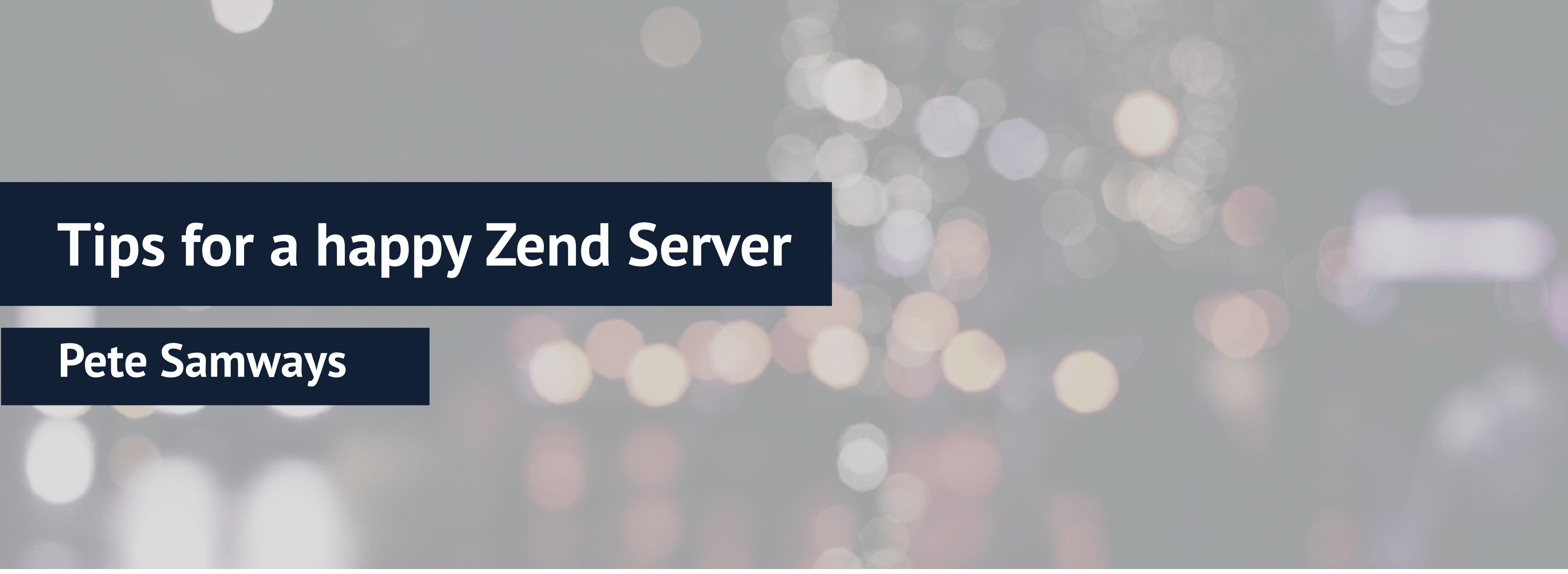
Tips for a happy Zend Server

Part of the PHP on IBM i series



Tips for a happy Zend Server

Pete Samways

In our first PHP on i guide, Pete Samways looked at how to get started with PHP on IBM i. Server side, this involved installing Zend Server – the PHP engine that makes things happen.

In this guide, Pete shares some tips on how to configure Zend Server to avoid problems and get the best out of it.

Configure your character sets

Firstly, I cannot stress this strongly enough, you ***must*** get your CCSID set up right from the start.

Of all the challenges we have had to overcome with PHP on the IBM i in the past, the biggest and ugliest have been related to CCSIDs.

Before we even look at Zend Server, some database admin. Any file you plan to access from PHP should have a proper CCSID. A proper CCSIS is one that isn't 65535. I don't care what it is, just not 65535.

Personally, I'd extend this rule to cover any file you plan to access from anywhere, but it's not my job to tell you how to run your life or database.

The transparent nature of CCSID 65535 means that it's impossible to convert from this to other character sets with any consistency or predictability. Not a great place to be if you want your data to make sense on the web.

If you're planning to capture data in your PHP programs, it might also be worth considering CCSID 1208 for the files that you will save it in. This is the CCSID for UTF-8, and means you'll be ready to store any tricky international characters that users might throw at you. The drawback with this CCSID is, however, that you will only be able to access your data using SQL, not record level access within your legacy programs.

And now to Zend Server itself.

We're going to change some configuration files in the IFS, an act that itself can suffer from character set issues if done using a 5250 session. Personally, I use an FTP client like FileZilla to access any files that need to be changed so that they are edited locally on my PC, and I'd recommend you do the same.

And always take a backup copy before you change anything.

The first file to edit is `httpd.conf` in folder `/www/zendsvr6/conf` which is the configuration file for the HTTP server instance generated by Zend Server.

Open it up, and change the following lines to look like:

```
DefaultFsCCSID 00285
 CGIJobCCSID 00285
 DefaultNetCCSID 01208
```

The values of 285 for the first two settings work for us here in the UK, but you could set them to a value more appropriate to your location. The net CCSID of 1208, as mentioned earlier, indicates UTF-8.

Also, look for an `AddCharset` directive relating to UTF-8. If one doesn't exist, add one like:

```
AddCharset UTF-8 .htm .html .xml
```

Save the file and load it back into the IFS.

Next, we're going to make a small change to file `fastcgi.conf`, also in folder `/www/zendsvr6/conf`.

This file contains configuration for the FastCGI jobs which do a lot of the busy work such as running SQLs and calling native programs.

Switch on compression

Open the file, and look for the first (very large) directive. This will start with either `Server` or `DynamicServer`, depending on whether FastCGI is configured to be static or dynamic – more on this later.

Within the directive, look for `SetEnv` sections that relate to CCSID and LANG. They'll look something like:

```
    SetEnv="CCSID=819" SetEnv="LANG=en_US"
```

Change these values to look like:

```
    SetEnv="CCSID=1208" SetEnv="LANG=C"
```

Save the file and load it back into the IFS.

Now we're going to change file `toolkit.ini` in folder `/usr/local/zendsvr6/share/ToolkitAPI` which is the configuration file for the XML Toolkit. This is a separate piece of software that allows us to easily access the native database and functions on the IBM i.

Open the file and look for the line where the `encoding` property is set, and change it from the default of ISO-8859-1 to UTF-8.

After your changes the line in question should look like:

```
    encoding = "UTF-8"
```

Save the file and load it back into the IFS.

The last job is to change a directive relating to the PHP DB2 support module. Fortunately, we can use the Zend console to change this, much less scary than FTPing files around the place.

Log into the Zend console, then navigate to the Configurations tab, and within that the PHP tab.

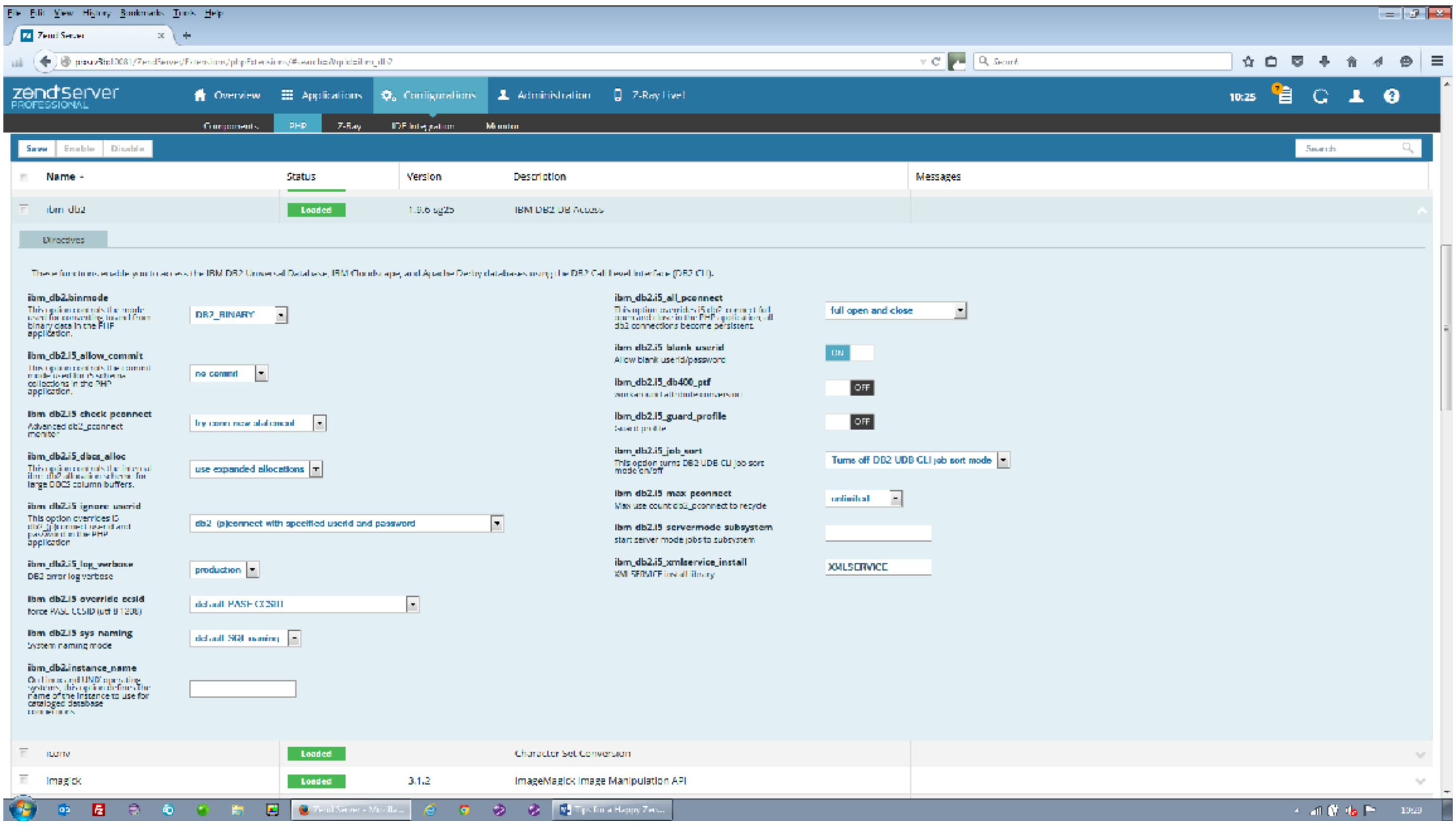
This page lists all the PHP modules installed, and allows you to change the associated configuration.

You're looking for a section marked ibm_db2. When you find it, click on it to show its directives.

The key value for us here is the ibm_db2.i5_dbcs_alloc directive – quite a mouthful. This should be set to use expanded allocations. Without this you can end up with truncated strings appearing in your database.

After all this, you'll want to end and restart Zend Server.

This configuration should hopefully mean you avoid any character set problems when you start accessing your database and native resources such as RPG programs from your PHP scripts.



	Name	Status	Version	Description	Messages
	ibm_db2	Loaded	1.9.6 sg25	IBM DB2 DB Access	

These functions enable you to access the IBM DB2 Universal Database, IBM Cloudscape, and Apache Derby databases using the DB2 Call Level Interface (DB2 CLI).

Directive	Description	Value
ibm_db2.binmode	This option controls the mode used for converting to and from binary data in the PHP application.	DB2_BINARY
ibm_db2.i5_allow_commit	This option controls the commit mode used for i5 schema collections in the PHP application.	no commit
ibm_db2.i5_check_pconnect	Advanced db2_pconnect monitor	[illegible]
ibm_db2.i5_dbcs_alloc	This option controls the internal ibm_db2 allocation scheme for large DBCS column buffers.	use expanded allocations
ibm_db2.i5_ignore_userid	This option overrides i5 db2_(p)connect userid and password in the PHP application	db2_(p)connect with specified userid and password
ibm_db2.i5_log_verbose	DB2 error log verbose	production
ibm_db2.i5_override_ccsid	force PASE CCSID (utf 8 1208)	default PASE CCSID
ibm_db2.i5_sys_naming	System naming mode	default SQL naming
ibm_db2.instance_name	On Linux and UNIX operating systems, this option defines the name of the instance to use for cataloged database connections.	
ibm_db2.i5_all_pconnect	This option overrides i5 db2_connect full open and close in the PHP application, all db2 connections become persistent.	full open and close
ibm_db2.i5_blank_userid	Allow blank userid/password	ON
ibm_db2.i5_db400_ptf	[illegible]	OFF
ibm_db2.i5_guard_profile	Guard profile	OFF
ibm_db2.i5_job_sort	This option turns DB2 UDB CLI job sort mode on/off	Turns off DB2 UDB CLI job sort mode
ibm_db2.i5_max_pconnect	Max use count db2_pconnect to recycle	unlimited
ibm_db2.i5_servermode_subsystem	start server mode jobs to subsystem	
ibm_db2.i5_xmlservice_install	XMLSERVICE install library	XMLSERVICE

	Name	Status	Version	Description	Messages
	iconv	Loaded		Character Set Conversion	
	imagick	Loaded	3.1.2	ImageMagick Image Manipulation API	

Proximity For information about PHP on i, visit www.proximity.co.uk/resources/php-on-i

Use Multibyte Support

Once you've got the CCSID aspects of Zend Server running sweetly, it's worth considering switching on Multibyte character support.

This PHP configuration allows you to use character sets that use more than one byte per character, a must if you want to be able to handle anything other than the most basic ASCII symbols.

Again, we can use the Zend console to change this set up.

In the Zend console, navigate to the Configurations tab, and within that the PHP tab.

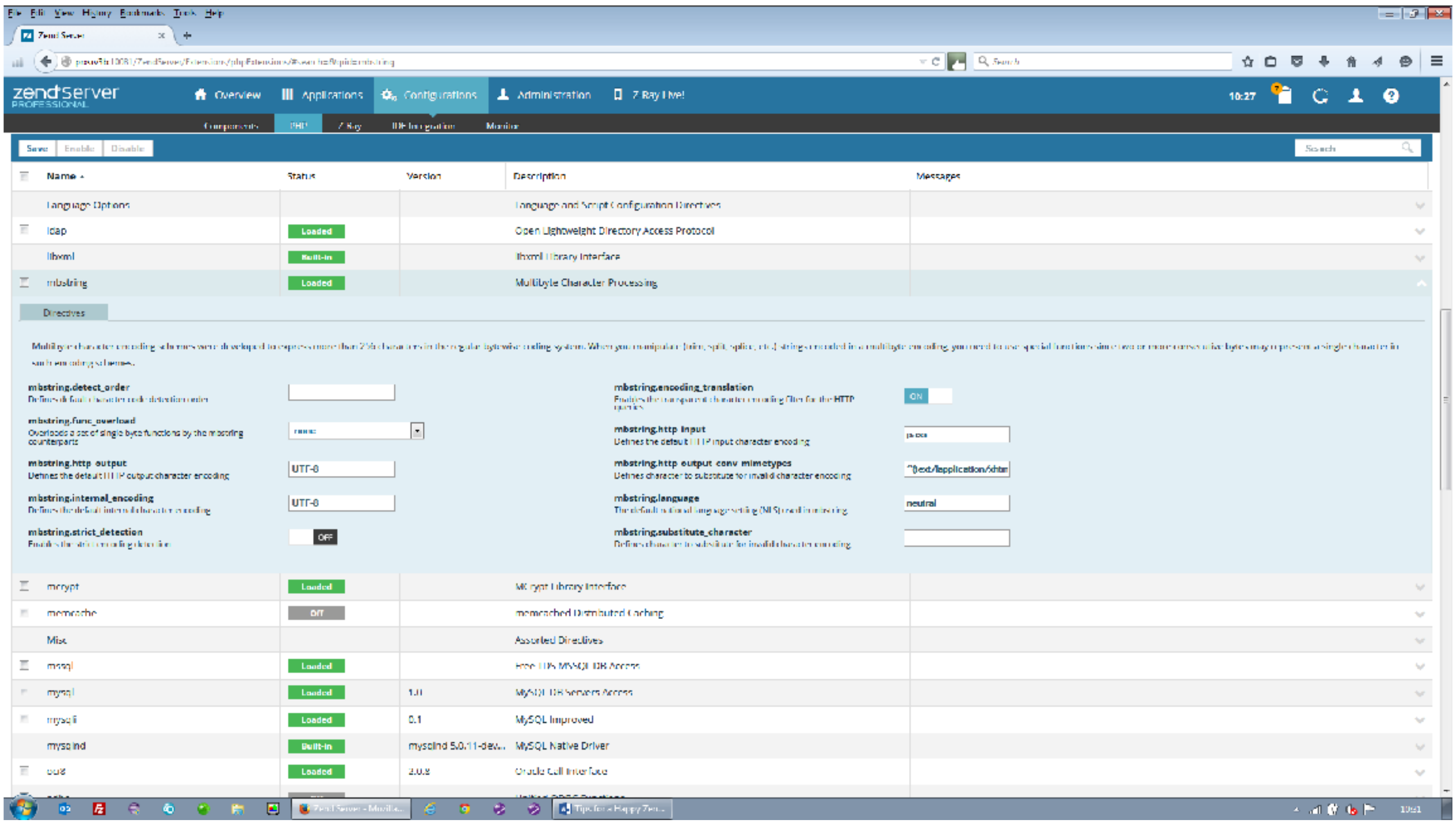
Find the section marked mbstring, and click on it to show its directives.

I recommend the following settings:

mbstring.http_output	UTF-8
mbstring.internal_encoding	UTF-8
mbstring.encoding_translation	On
mbstring.http_input	Pass
mbstring.language	neutral

Having gone to all this trouble, it's worth pointing out that when you start to write your PHP code you should be using the multibyte string handling functions that the language provides. There are single byte functions provided for backward compatibility, but these should be avoided.

For more information on the PHP multibyte functions, see the php.net web site section on Multibyte support.



Name	Status	Version	Description	Messages
Language Options			Language and Script Configuration Directives	
ldap	Loaded		Open Lightweight Directory Access Protocol	
libxml	Built-in		libxml Library Interface	
mbstring	Loaded		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.

mbstring.detect_order Defines default character code detection order		**mbstring.encoding_translation** Enables the transparent character encoding filter for the HTTP input	ON
mbstring.func_overload Overloads a set of single byte functions by the mbstring counterparts	none	**mbstring.http_input** Defines the default HTTP input character encoding	pass
mbstring.http_output Defines the default HTTP output character encoding	UTF-8	**mbstring.http_output_conv_mimetypes** Defines character to substitute for invalid character encoding	^(text/|application/xhtm
mbstring.internal_encoding Defines the default internal character encoding	UTF-8	**mbstring.language** The default national language setting (NLS) used in mbstring	neutral
mbstring.strict_detection Enables the strict encoding detection	OFF	**mbstring.substitute_character** Defines character to substitute for invalid character encoding	

Name	Status	Version	Description	Messages
mcrypt	Loaded		MCrypt Library Interface	
memcache	Off		memcached Distributed Caching	
Misc			Assorted Directives	
mssql	Loaded		Free TDS MSSQL DB Access	
mysql	Loaded	1.0	MySQL DB Servers Access	
mysqli	Loaded	0.1	MySQL Improved	
mysqlnd	Built-in	mysqlnd 5.0.11-dev...	MySQL Native Driver	
oci8	Loaded	2.0.8	Oracle Call Interface	

Proximity For information about PHP on i, visit www.proximity.co.uk/resources/php-on-i

Switch on compression

For some strange reason, Zend Server on the IBM i comes out of the box with compression switched off.

Compression is a feature of the HTTP server that reduces the size of the data being passed back and forth to the browser.

Less data means faster transactions, and a more pleasant user experience.

As a beginner this might not seem important, but trust me, once you start deploying applications to growing user bases it rapidly becomes an issue. So it's good to get this quick win in place early on.

As this is a feature of the HTTP server, we're back into the realm of editing configuration files, specifically the `httpd.conf` in folder `/www/zendsvr6/conf` that we saw earlier.

At the top of this file you'll see a load of line starting `LoadModule`. If it does not already exist, add one as follows:

```
LoadModule deflate_module
/QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
```

And below the LoadModule lines, add a line as follows:

```
   AddOutputFilterByType DEFLATE
       application/x-httpd-php
       application/json text/css
       application/x-javascript
       application/javascript text/html
```

This tells the HTTP server to compress any eligible data, assuming that the browser request indicates that compression is allowed, which is most cases.

Go Dynamic

I'll end with a slightly contentious one.

It's possible to configure the FastCGI jobs in two ways, as I mentioned earlier.

Static configuration – what you get out of the box – prestarts a set number of jobs to handle requests. This number of jobs is configurable but fixed.

Dynamic configuration is more flexible. It starts with fewer jobs started, but will happily start more jobs to handle increases in workload.

It's generally recommended to use dynamic configuration. We use it, and have found it to work well. The contentious point is that occasionally it can perform badly, and the reason for this is unclear – according to the Young i Professionals website, it's probably due to a rare combination of PTFs.

If you do want to go dynamic, Zend very helpfully supply a dynamic version of the configuration file `fastcgi_dynamic.conf` in folder `/www/zendsvr6/conf`. Simply rename the existing file, then rename the dynamic version to `fastcgi.conf` and you're in business.

Don't forget to apply the CCSID changes to the dynamic version, though...

About the author:

Pete Samways is a development manager at Proximity. He is an expert in the development, modernisation and mobilisation of IBM i applications, particularly in the logistics sector.

He is responsible for delivering projects on behalf of Gebrüder Weiss and Agility amongst others. You can follow Pete on Twitter @pete_samways

Useful links and information about PHP on i

Zend is the leading provider of enterprise-grade applications for PHP. The PHP engine for IBM i, Zend Server, will run natively on your IBM i and is available as a simple and free download to all IBM i users.

BCD Software has been supporting PHP on IBM i since it first launched a specialised PHP version of WebSmart in 2007. Proximity is the exclusive UK and Ireland Partner for BCD Software.

PHP is the world's most popular programming language for web and mobile applications.

Partnering with some of the world's foremost software companies, Proximity develops, delivers, maintains and supports high performance solutions and applications for leading global companies in the logistics, manufacturing, retail and finance sectors.

Part of the PHP on IBM i series

For information about PHP on i, visit www.proximity.co.uk/resources/php-on-i

Proximity Group

t: +44 (0) 113 393 3360

e: info@proximity.co.uk

Leeds office

4-6 Kerry Hill, Horsforth,

Leeds, LS18 4AY

Nottingham office

Pure Offices, Lakeview Drive,

Nottingham, NG15 0DT