```
[PHP]

;;;;;;;;;;;
; WARNING ;
;;;;;;;;;;;
; This is the default settings file for new PHP installations.
; By default, PHP installs itself with a configuration suitable for
; development purposes, and *NOT* for production purposes.
; For several security-oriented considerations that should be taken
; before going online with your site, please consult php.ini-recommended
; and http://php.net/manual/en/security.php.


;;;;;;;;;;;;;;;;;;;
; About php.ini   ;
;;;;;;;;;;;;;;;;;;;
; This file controls many aspects of PHP's behavior.  In order for PHP to
; read it, it must be named 'php.ini'.  PHP looks for it in the current
; working directory, in the path designated by the environment variable
; PHPRC, and in the path that was defined in compile time (in that order).
; Under Windows, the compile-time path is the Windows directory.  The
; path in which the php.ini file is looked for can be overridden using
; the -c argument in command line mode.
;
; The syntax of the file is extremely simple.  Whitespace and Lines
; beginning with a semicolon are silently ignored (as you probably guessed).
; Section headers (e.g. [Foo]) are also silently ignored, even though
; they might mean something in the future.
;
; Directives are specified using the following syntax:
; directive = value
; Directive names are *case sensitive* - foo=bar is different from FOO=bar.
;
; The value can be a string, a number, a PHP constant (e.g. E_ALL or M_PI), one
; of the INI constants (On, Off, True, False, Yes, No and None) or an expression
; (e.g. E_ALL & ~E_NOTICE), or a quoted string ("foo").
;
; Expressions in the INI file are limited to bitwise operators and parentheses:
; |        bitwise OR
; &        bitwise AND
; ~        bitwise NOT
; !        boolean NOT
;
; Boolean flags can be turned on using the values 1, On, True or Yes.
; They can be turned off using the values 0, Off, False or No.
;
; An empty string can be denoted by simply not writing anything after the equal
; sign, or by using the None keyword:
;
;  foo =         ; sets foo to an empty string
;  foo = none    ; sets foo to an empty string
;  foo = "none"  ; sets foo to the string 'none'
;
; If you use constants in your value, and these constants belong to a
; dynamically loaded extension (either a PHP extension or a Zend extension),
; you may only use these constants *after* the line that loads the extension.
;
;
```

```
 59: ;;;;;;;;;;;;;;;;;;;
 60: ; About this file ;
 61: ;;;;;;;;;;;;;;;;;;;
 62: ; All the values in the php.ini-dist file correspond to the builtin
 63: ; defaults (that is, if no php.ini is used, or if you delete these lines,
 64: ; the builtin defaults will be identical).
 65:
 66:
 67: ;;;;;;;;;;;;;;;;;;;;
 68: ; Language Options ;
 69: ;;;;;;;;;;;;;;;;;;;;
 70:
 71: ; Enable the PHP scripting language engine under Apache.
 72: engine = On
 73:
 74: ; Enable compatibility mode with Zend Engine 1 (PHP 4.x)
 75: zend.ze1_compatibility_mode = Off
 76:
 77: ; Allow the <? tag.  Otherwise, only <?php and <script> tags are recognized.
 78: ; NOTE: Using short tags should be avoided when developing applications or
 79: ; libraries that are meant for redistribution, or deployment on PHP
 80: ; servers which are not under your control, because short tags may not
 81: ; be supported on the target server. For portable, redistributable code,
 82: ; be sure not to use short tags.
 83: short_open_tag = On
 84:
 85: ; Allow ASP-style <% %> tags.
 86: asp_tags = Off
 87:
 88: ; The number of significant digits displayed in floating point numbers.
 89: precision    =  12
 90:
 91: ; Enforce year 2000 compliance (will cause problems with non-compliant browsers)
 92: y2k_compliance = On
 93:
 94: ; Output buffering allows you to send header lines (including cookies) even
 95: ; after you send body content, at the price of slowing PHP's output layer a
 96: ; bit.  You can enable output buffering during runtime by calling the output
 97: ; buffering functions.  You can also enable output buffering for all files by
 98: ; setting this directive to On.  If you wish to limit the size of the buffer
 99: ; to a certain size - you can use a maximum number of bytes instead of 'On', as
100: ; a value for this directive (e.g., output_buffering=4096).
101: output_buffering = Off
102:
103: ; You can redirect all of the output of your scripts to a function.  For
104: ; example, if you set output_handler to "mb_output_handler", character
105: ; encoding will be transparently converted to the specified encoding.
106: ; Setting any output handler automatically turns on output buffering.
107: ; Note: People who wrote portable scripts should not depend on this ini
108: ;       directive. Instead, explicitly set the output handler using ob_start().
109: ;       Using this ini directive may cause problems unless you know what script
110: ;       is doing.
111: ; Note: You cannot use both "mb_output_handler" with "ob_iconv_handler"
112: ;       and you cannot use both "ob_gzhandler" and "zlib.output_compression".
113: ; Note: output_handler must be empty if this is set 'On' !!!!
114: ;       Instead you must use zlib.output_handler.
115: ;output_handler =
116:
```

```
117: ; Transparent output compression using the zlib library
118: ; Valid values for this option are 'off', 'on', or a specific buffer size
119: ; to be used for compression (default is 4KB)
120: ; Note: Resulting chunk size may vary due to nature of compression. PHP
121: ;       outputs chunks that are few hundreds bytes each as a result of
122: ;       compression. If you prefer a larger chunk size for better
123: ;       performance, enable output_buffering in addition.
124: ; Note: You need to use zlib.output_handler instead of the standard
125: ;       output_handler, or otherwise the output will be corrupted.
126: zlib.output_compression = Off
127: ;zlib.output_compression_level = -1
128:
129: ; You cannot specify additional output handlers if zlib.output_compression
130: ; is activated here. This setting does the same as output_handler but in
131: ; a different order.
132: ;zlib.output_handler =
133:
134: ; Implicit flush tells PHP to tell the output layer to flush itself
135: ; automatically after every output block.  This is equivalent to calling the
136: ; PHP function flush() after each and every call to print() or echo() and each
137: ; and every HTML block.  Turning this option on has serious performance
138: ; implications and is generally recommended for debugging purposes only.
139: implicit_flush = Off
140:
141: ; The unserialize callback function will be called (with the undefined class'
142: ; name as parameter), if the unserializer finds an undefined class
143: ; which should be instantiated.
144: ; A warning appears if the specified function is not defined, or if the
145: ; function doesn't include/implement the missing class.
146: ; So only set this entry, if you really want to implement such a
147: ; callback-function.
148: unserialize_callback_func=
149:
150: ; When floats & doubles are serialized store serialize_precision significant
151: ; digits after the floating point. The default value ensures that when floats
152: ; are decoded with unserialize, the data will remain the same.
153: serialize_precision = 100
154:
155: ; Whether to enable the ability to force arguments to be passed by reference
156: ; at function call time.  This method is deprecated and is likely to be
157: ; unsupported in future versions of PHP/Zend.  The encouraged method of
158: ; specifying which arguments should be passed by reference is in the function
159: ; declaration.  You're encouraged to try and turn this option Off and make
160: ; sure your scripts work properly with it in order to ensure they will work
161: ; with future versions of the language (you will receive a warning each time
162: ; you use this feature, and the argument will be passed by value instead of by
163: ; reference).
164: allow_call_time_pass_reference = On
165:
166: ;
167: ; Safe Mode
168: ;
169: safe_mode = Off
170:
171: ; By default, Safe Mode does a UID compare check when
172: ; opening files. If you want to relax this to a GID compare,
173: ; then turn on safe_mode_gid.
174: safe_mode_gid = Off
```

```
175:
176: ; When safe_mode is on, UID/GID checks are bypassed when
177: ; including files from this directory and its subdirectories.
178: ; (directory must also be in include_path or full path must
179: ; be used when including)
180: safe_mode_include_dir =
181:
182: ; When safe_mode is on, only executables located in the safe_mode_exec_dir
183: ; will be allowed to be executed via the exec family of functions.
184: safe_mode_exec_dir =
185:
186: ; Setting certain environment variables may be a potential security breach.
187: ; This directive contains a comma-delimited list of prefixes.  In Safe Mode,
188: ; the user may only alter environment variables whose names begin with the
189: ; prefixes supplied here.  By default, users will only be able to set
190: ; environment variables that begin with PHP_ (e.g. PHP_FOO=BAR).
191: ;
192: ; Note:  If this directive is empty, PHP will let the user modify ANY
193: ; environment variable!
194: safe_mode_allowed_env_vars = PHP_
195:
196: ; This directive contains a comma-delimited list of environment variables that
197: ; the end user won't be able to change using putenv().  These variables will be
198: ; protected even if safe_mode_allowed_env_vars is set to allow to change them.
199: safe_mode_protected_env_vars = LD_LIBRARY_PATH
200:
201: ; open_basedir, if set, limits all file operations to the defined directory
202: ; and below.  This directive makes most sense if used in a per-directory
203: ; or per-virtualhost web server configuration file. This directive is
204: ; *NOT* affected by whether Safe Mode is turned On or Off.
205: ;open_basedir =
206:
207: ; This directive allows you to disable certain functions for security reasons.
208: ; It receives a comma-delimited list of function names. This directive is
209: ; *NOT* affected by whether Safe Mode is turned On or Off.
210: disable_functions =
211:
212: ; This directive allows you to disable certain classes for security reasons.
213: ; It receives a comma-delimited list of class names. This directive is
214: ; *NOT* affected by whether Safe Mode is turned On or Off.
215: disable_classes =
216:
217: ; Colors for Syntax Highlighting mode.  Anything that's acceptable in
218: ; <span style="color: ???????"> would work.
219: ;highlight.string  = #DD0000
220: ;highlight.comment = #FF9900
221: ;highlight.keyword = #007700
222: ;highlight.bg      = #FFFFFF
223: ;highlight.default = #0000BB
224: ;highlight.html    = #000000
225:
226: ; If enabled, the request will be allowed to complete even if the user aborts
227: ; the request. Consider enabling it if executing long request, which may end up
228: ; being interrupted by the user or a browser timing out.
229: ; ignore_user_abort = On
230:
231: ; Determines the size of the realpath cache to be used by PHP. This value should
232: ; be increased on systems where PHP opens many files to reflect the quantity of
```

```
233: ; the file operations performed.
234: ; realpath_cache_size=16k
235:
236: ; Duration of time, in seconds for which to cache realpath information for a given
237: ; file or directory. For systems with rarely changing files, consider increasing
     this
238: ; value.
239: ; realpath_cache_ttl=120
240:
241: ;
242: ; Misc
243: ;
244: ; Decides whether PHP may expose the fact that it is installed on the server
245: ; (e.g. by adding its signature to the Web server header).  It is no security
246: ; threat in any way, but it makes it possible to determine whether you use PHP
247: ; on your server or not.
248: expose_php = On
249:
250:
251: ;;;;;;;;;;;;;;;;;;;
252: ; Resource Limits ;
253: ;;;;;;;;;;;;;;;;;;;
254:
255: max_execution_time = 30     ; Maximum execution time of each script, in seconds
256: max_input_time = 60 ; Maximum amount of time each script may spend parsing request
     data
257: ;max_input_nesting_level = 64 ; Maximum input variable nesting level
258: memory_limit = 128M      ; Maximum amount of memory a script may consume (128MB)
259:
260:
261: ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
262: ; Error handling and logging ;
263: ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
264:
265: ; error_reporting is a bit-field.  Or each number up to get desired error
266: ; reporting level
267: ; E_ALL             - All errors and warnings (doesn't include E_STRICT)
268: ; E_ERROR           - fatal run-time errors
269: ; E_RECOVERABLE_ERROR  - almost fatal run-time errors
270: ; E_WARNING         - run-time warnings (non-fatal errors)
271: ; E_PARSE           - compile-time parse errors
272: ; E_NOTICE          - run-time notices (these are warnings which often result
273: ;                     from a bug in your code, but it's possible that it was
274: ;                     intentional (e.g., using an uninitialized variable and
275: ;                     relying on the fact it's automatically initialized to an
276: ;                     empty string)
277: ; E_STRICT          - run-time notices, enable to have PHP suggest changes
278: ;                     to your code which will ensure the best interoperability
279: ;                     and forward compatibility of your code
280: ; E_CORE_ERROR      - fatal errors that occur during PHP's initial startup
281: ; E_CORE_WARNING    - warnings (non-fatal errors) that occur during PHP's
282: ;                     initial startup
283: ; E_COMPILE_ERROR   - fatal compile-time errors
284: ; E_COMPILE_WARNING - compile-time warnings (non-fatal errors)
285: ; E_USER_ERROR      - user-generated error message
286: ; E_USER_WARNING    - user-generated warning message
287: ; E_USER_NOTICE     - user-generated notice message
288: ;
```

```
289: ; Examples:
290: ;
291: ;   - Show all errors, except for notices and coding standards warnings
292: ;
293: ;error_reporting = E_ALL & ~E_NOTICE
294: ;
295: ;   - Show all errors, except for notices
296: ;
297: ;error_reporting = E_ALL & ~E_NOTICE | E_STRICT
298: ;
299: ;   - Show only errors
300: ;
301: ;error_reporting = E_COMPILE_ERROR|E_RECOVERABLE_ERROR|E_ERROR|E_CORE_ERROR
302: ;
303: ;   - Show all errors except for notices and coding standards warnings
304: ;
305: error_reporting  =  E_ALL & ~E_NOTICE
306:
307: ; Print out errors (as a part of the output).  For production web sites,
308: ; you're strongly encouraged to turn this feature off, and use error logging
309: ; instead (see below).  Keeping display_errors enabled on a production web site
310: ; may reveal security information to end users, such as file paths on your Web
311: ; server, your database schema or other information.
312: ;
313: ; possible values for display_errors:
314: ;
315: ; Off        - Do not display any errors
316: ; stderr     - Display errors to STDERR (affects only CGI/CLI binaries!)
317: ;
318: ;display_errors = "stderr"
319: ;
320: ; stdout (On) - Display errors to STDOUT
321: ;
322: display_errors = On
323:
324: ; Even when display_errors is on, errors that occur during PHP's startup
325: ; sequence are not displayed.  It's strongly recommended to keep
326: ; display_startup_errors off, except for when debugging.
327: display_startup_errors = Off
328:
329: ; Log errors into a log file (server-specific log, stderr, or error_log (below))
330: ; As stated above, you're strongly advised to use error logging in place of
331: ; error displaying on production web sites.
332: log_errors = Off
333:
334: ; Set maximum length of log_errors. In error_log information about the source is
335: ; added. The default is 1024 and 0 allows to not apply any maximum length at all.
336: log_errors_max_len = 1024
337:
338: ; Do not log repeated messages. Repeated errors must occur in same file on same
339: ; line until ignore_repeated_source is set true.
340: ignore_repeated_errors = Off
341:
342: ; Ignore source of message when ignoring repeated messages. When this setting
343: ; is On you will not log errors with repeated messages from different files or
344: ; source lines.
345: ignore_repeated_source = Off
346:
```

```
347: ; If this parameter is set to Off, then memory leaks will not be shown (on
348: ; stdout or in the log). This has only effect in a debug compile, and if
349: ; error reporting includes E_WARNING in the allowed list
350: report_memleaks = On
351:
352: ;report_zend_debug = 0
353:
354: ; Store the last error/warning message in $php_errormsg (boolean).
355: track_errors = Off
356:
357: ; Disable the inclusion of HTML tags in error messages.
358: ; Note: Never use this feature for production boxes.
359: ;html_errors = Off
360:
361: ; If html_errors is set On PHP produces clickable error messages that direct
362: ; to a page describing the error or function causing the error in detail.
363: ; You can download a copy of the PHP manual from http://www.php.net/docs.php
364: ; and change docref_root to the base URL of your local copy including the
365: ; leading '/'. You must also specify the file extension being used including
366: ; the dot.
367: ; Note: Never use this feature for production boxes.
368: ;docref_root = "/phpmanual/"
369: ;docref_ext = .html
370:
371: ; String to output before an error message.
372: ;error_prepend_string = "<font color=#ff0000>"
373:
374: ; String to output after an error message.
375: ;error_append_string = "</font>"
376:
377: ; Log errors to specified file.
378: ;error_log = filename
379:
380: ; Log errors to syslog (Event Log on NT, not valid in Windows 95).
381: ;error_log = syslog
382:
383:
384: ;;;;;;;;;;;;;;;;;
385: ; Data Handling ;
386: ;;;;;;;;;;;;;;;;;
387: ;
388: ; Note - track_vars is ALWAYS enabled as of PHP 4.0.3
389:
390: ; The separator used in PHP generated URLs to separate arguments.
391: ; Default is "&".
392: ;arg_separator.output = "&amp;"
393:
394: ; List of separator(s) used by PHP to parse input URLs into variables.
395: ; Default is "&".
396: ; NOTE: Every character in this directive is considered as separator!
397: ;arg_separator.input = ";&"
398:
399: ; This directive describes the order in which PHP registers GET, POST, Cookie,
400: ; Environment and Built-in variables (G, P, C, E & S respectively, often
401: ; referred to as EGPCS or GPC).  Registration is done from left to right, newer
402: ; values override older values.
403: variables_order = "EGPCS"
404:
```

```
405: ; Whether or not to register the EGPCS variables as global variables.  You may
406: ; want to turn this off if you don't want to clutter your scripts' global scope
407: ; with user data.  This makes most sense when coupled with track_vars - in which
408: ; case you can access all of the GPC variables through the $HTTP_*_VARS[],
409: ; variables.
410: ;
411: ; You should do your best to write your scripts so that they do not require
412: ; register_globals to be on;  Using form variables as globals can easily lead
413: ; to possible security problems, if the code is not very well thought of.
414: register_globals = On
415:
416: ; Whether or not to register the old-style input arrays, HTTP_GET_VARS
417: ; and friends.  If you're not using them, it's recommended to turn them off,
418: ; for performance reasons.
419: register_long_arrays = On
420:
421: ; This directive tells PHP whether to declare the argv&argc variables (that
422: ; would contain the GET information).  If you don't use these variables, you
423: ; should turn it off for increased performance.
424: register_argc_argv = On
425:
426: ; When enabled, the SERVER and ENV variables are created when they're first
427: ; used (Just In Time) instead of when the script starts. If these variables
428: ; are not used within a script, having this directive on will result in a
429: ; performance gain. The PHP directives register_globals, register_long_arrays,
430: ; and register_argc_argv must be disabled for this directive to have any affect.
431: auto_globals_jit = On
432:
433: ; Maximum size of POST data that PHP will accept.
434: post_max_size = 8M
435:
436: ; Magic quotes
437: ;
438:
439: ; Magic quotes for incoming GET/POST/Cookie data.
440: magic_quotes_gpc = On
441:
442: ; Magic quotes for runtime-generated data, e.g. data from SQL, from exec(), etc.
443: magic_quotes_runtime = Off
444:
445: ; Use Sybase-style magic quotes (escape ' with '' instead of \').
446: magic_quotes_sybase = Off
447:
448: ; Automatically add files before or after any PHP document.
449: auto_prepend_file =
450: auto_append_file =
451:
452: ; As of 4.0b4, PHP always outputs a character encoding by default in
453: ; the Content-type: header.  To disable sending of the charset, simply
454: ; set it to be empty.
455: ;
456: ; PHP's built-in default is text/html
457: default_mimetype = "text/html"
458: ;default_charset = "iso-8859-1"
459:
460: ; Always populate the $HTTP_RAW_POST_DATA variable.
461: ;always_populate_raw_post_data = On
462:
```

```
463:
464: ;;;;;;;;;;;;;;;;;;;;;;;
465: ; Paths and Directories ;
466: ;;;;;;;;;;;;;;;;;;;;;;;
467:
468: ; UNIX: "/path1:/path2"
469: ;include_path = ".:/php/includes"
470: ;
471: ; Windows: "\path1;\path2"
472: ;include_path = ".;c:\php\includes"
473:
474: ; The root of the PHP pages, used only if nonempty.
475: ; if PHP was not compiled with FORCE_REDIRECT, you SHOULD set doc_root
476: ; if you are running php as a CGI under any web server (other than IIS)
477: ; see documentation for security issues.  The alternate is to use the
478: ; cgi.force_redirect configuration below
479: doc_root =
480:
481: ; The directory under which PHP opens the script using /~username used only
482: ; if nonempty.
483: user_dir =
484:
485: ; Directory in which the loadable extensions (modules) reside.
486: extension_dir = "/ABCD/php/ext"
487:
488: ; Whether or not to enable the dl() function.  The dl() function does NOT work
489: ; properly in multithreaded servers, such as IIS or Zeus, and is automatically
490: ; disabled on them.
491: enable_dl = On
492:
493: ; cgi.force_redirect is necessary to provide security running PHP as a CGI under
494: ; most web servers.  Left undefined, PHP turns this on by default.  You can
495: ; turn it off here AT YOUR OWN RISK
496: ; **You CAN safely turn this off for IIS, in fact, you MUST.**
497: ; cgi.force_redirect = 1
498:
499: ; if cgi.nph is enabled it will force cgi to always sent Status: 200 with
500: ; every request.
501: ; cgi.nph = 1
502:
503: ; if cgi.force_redirect is turned on, and you are not running under Apache or
     Netscape
504: ; (iPlanet) web servers, you MAY need to set an environment variable name that PHP
505: ; will look for to know it is OK to continue execution.  Setting this variable MAY
506: ; cause security issues, KNOW WHAT YOU ARE DOING FIRST.
507: ; cgi.redirect_status_env = ;
508:
509: ; cgi.fix_pathinfo provides *real* PATH_INFO/PATH_TRANSLATED support for CGI.  PHP's
510: ; previous behaviour was to set PATH_TRANSLATED to SCRIPT_FILENAME, and to not grok
511: ; what PATH_INFO is.  For more information on PATH_INFO, see the cgi specs.  Setting
512: ; this to 1 will cause PHP CGI to fix it's paths to conform to the spec.  A setting
513: ; of zero causes PHP to behave as before.  Default is 1.  You should fix your
     scripts
514: ; to use SCRIPT_FILENAME rather than PATH_TRANSLATED.
515: ; cgi.fix_pathinfo=0
516:
517: ; FastCGI under IIS (on WINNT based OS) supports the ability to impersonate
518: ; security tokens of the calling client.  This allows IIS to define the
```

```
519: ; security context that the request runs under.  mod_fastcgi under Apache
520: ; does not currently support this feature (03/17/2002)
521: ; Set to 1 if running under IIS.  Default is zero.
522: ; fastcgi.impersonate = 1;
523:
524: ; Disable logging through FastCGI connection
525: ; fastcgi.logging = 0
526:
527: ; cgi.rfc2616_headers configuration option tells PHP what type of headers to
528: ; use when sending HTTP response code. If it's set 0 PHP sends Status: header that
529: ; is supported by Apache. When this option is set to 1 PHP will send
530: ; RFC2616 compliant header.
531: ; Default is zero.
532: ;cgi.rfc2616_headers = 0
533:
534:
535: ;;;;;;;;;;;;;;;;
536: ; File Uploads ;
537: ;;;;;;;;;;;;;;;;
538:
539: ; Whether to allow HTTP file uploads.
540: file_uploads = On
541:
542: ; Temporary directory for HTTP uploaded files (will use system default if not
543: ; specified).
544: ;upload_tmp_dir =
545:
546: ; Maximum allowed size for uploaded files.
547: upload_max_filesize = 2M
548:
549:
550: ;;;;;;;;;;;;;;;;;;
551: ; Fopen wrappers ;
552: ;;;;;;;;;;;;;;;;;;
553:
554: ; Whether to allow the treatment of URLs (like http:// or ftp://) as files.
555: allow_url_fopen = On
556:
557: ; Whether to allow include/require to open URLs (like http:// or ftp://) as files.
558: allow_url_include = Off
559:
560: ; Define the anonymous ftp password (your email address)
561: ;from="john@doe.com"
562:
563: ; Define the User-Agent string
564: ; user_agent="PHP"
565:
566: ; Default timeout for socket based streams (seconds)
567: default_socket_timeout = 60
568:
569: ; If your scripts have to deal with files from Macintosh systems,
570: ; or you are running on a Mac and need to deal with files from
571: ; unix or win32 systems, setting this flag will cause PHP to
572: ; automatically detect the EOL character in those files so that
573: ; fgets() and file() will work regardless of the source of the file.
574: ; auto_detect_line_endings = Off
575:
576:
```

```
577: ;;;;;;;;;;;;;;;;;;;;;
578: ; Dynamic Extensions ;
579: ;;;;;;;;;;;;;;;;;;;;;
580: ;
581: ; If you wish to have an extension loaded automatically, use the following
582: ; syntax:
583: ;
584: ;   extension=modulename.extension
585: ;
586: ; For example, on Windows:
587: ;
588: ;   extension=msql.dll
589: ;
590: ; ... or under UNIX:
591: ;
592: ;   extension=msql.so
593: ;
594: ; Note that it should be the name of the module only; no directory information
595: ; needs to go here.  Specify the location of the extension with the
596: ; extension_dir directive above.
597:
598:
599: ; Windows Extensions
600: ; Note that ODBC support is built in, so no dll is needed for it.
601: ; Note that many DLL files are located in the extensions/ (PHP 4) ext/ (PHP 5)
602: ; extension folders as well as the separate PECL DLL download (PHP 5).
603: ; Be sure to appropriately set the extension_dir directive.
604:
605:
606: ;extension=iconv.dll
607: ;extension=libxml2.dll
608: ;extension=libxslt.dll
609: ;extension=php_bz2.dll
610: ;extension=php_curl.dll
611: ;extension=php_dba.dll
612: ;extension=php_dbase.dll
613: ;extension=php_exif.dll
614: ;extension=php_fdf.dll
615: ;extension=php_gd2.dll
616: ;extension=php_gettext.dll
617: ;extension=php_gmp.dll
618: ;extension=php_ifx.dll
619: ;extension=php_imap.dll
620: ;extension=php_interbase.dll
621: ;extension=php_ldap.dll
622: ;extension=php_mbstring.dll
623: ;extension=php_mcrypt.dll
624: ;extension=php_mhash.dll
625: ;extension=php_mime_magic.dll
626: ;extension=php_ming.dll
627: ;extension=php_msql.dll
628: ;extension=php_mssql.dll
629: ;extension=php_mysql.dll
630: ;extension=php_mysqli.dll
631: ;extension=php_oci8.dll
632: ;extension=php_openssl.dll
633: ;extension=php_pdo.dll
634: ;extension=php_pdo_firebird.dll
```

```
635: ;extension=php_pdo_mssql.dll
636: ;extension=php_pdo_mysql.dll
637: ;extension=php_pdo_oci.dll
638: ;extension=php_pdo_oci8.dll
639: ;extension=php_pdo_odbc.dll
640: ;extension=php_pdo_pgsql.dll
641: ;extension=php_pdo_sqlite.dll
642: ;extension=php_pgsql.dll
643: ;extension=php_pspell.dll
644: ;extension=php_shmop.dll
645: ;extension=php_snmp.dll
646: ;extension=php_soap.dll
647: ;extension=php_sockets.dll
648: ;extension=php_sqlite.dll
649: ;extension=php_sybase_ct.dll
650: ;extension=php_tidy.dll
651: ;extension=php_xmlrpc.dll
652: ;extension=php_xsl.dll
653: ;extension=php_yaz.dll
654: ;extension=php_zip.dll
655: ;extension=yaz3.dll
656:
657:
658: ;;;;;;;;;;;;;;;;;;;
659: ; Module Settings ;
660: ;;;;;;;;;;;;;;;;;;;
661:
662: [Date]
663: ; Defines the default timezone used by the date functions
664: ;date.timezone =
665:
666: ;date.default_latitude = 31.7667
667: ;date.default_longitude = 35.2333
668:
669: ;date.sunrise_zenith = 90.583333
670: ;date.sunset_zenith = 90.583333
671:
672: [filter]
673: ;filter.default = unsafe_raw
674: ;filter.default_flags =
675:
676: [iconv]
677: ;iconv.input_encoding = ISO-8859-1
678: ;iconv.internal_encoding = ISO-8859-1
679: ;iconv.output_encoding = ISO-8859-1
680:
681: [sqlite]
682: ;sqlite.assoc_case = 0
683:
684: [xmlrpc]
685: ;xmlrpc_error_number = 0
686: ;xmlrpc_errors = 0
687:
688: [Pcre]
689: ;PCRE library backtracking limit.
690: ;pcre.backtrack_limit=100000
691:
692: ;PCRE library recursion limit.
```

```
693: ;Please note that if you set this value to a high number you may consume all
694: ;the available process stack and eventually crash PHP (due to reaching the
695: ;stack size limit imposed by the Operating System).
696: ;pcre.recursion_limit=100000
697:
698: [Syslog]
699: ; Whether or not to define the various syslog variables (e.g. $LOG_PID,
700: ; $LOG_CRON, etc.).  Turning it off is a good idea performance-wise.  In
701: ; runtime, you can define these variables by calling define_syslog_variables().
702: define_syslog_variables  = Off
703:
704: [mail function]
705: ; For Win32 only.
706: SMTP = localhost
707: smtp_port = 25
708:
709: ; For Win32 only.
710: ;sendmail_from = me@example.com
711:
712: ; For Unix only.  You may supply arguments as well (default: "sendmail -t -i").
713: ;sendmail_path =
714:
715: ; Force the addition of the specified parameters to be passed as extra parameters
716: ; to the sendmail binary. These parameters will always replace the value of
717: ; the 5th parameter to mail(), even in safe mode.
718: ;mail.force_extra_parameters =
719:
720: [SQL]
721: sql.safe_mode = Off
722:
723: [ODBC]
724: ;odbc.default_db    =  Not yet implemented
725: ;odbc.default_user  =  Not yet implemented
726: ;odbc.default_pw    =  Not yet implemented
727:
728: ; Allow or prevent persistent links.
729: odbc.allow_persistent = On
730:
731: ; Check that a connection is still valid before reuse.
732: odbc.check_persistent = On
733:
734: ; Maximum number of persistent links.  -1 means no limit.
735: odbc.max_persistent = -1
736:
737: ; Maximum number of links (persistent + non-persistent).  -1 means no limit.
738: odbc.max_links = -1
739:
740: ; Handling of LONG fields.  Returns number of bytes to variables.  0 means
741: ; passthru.
742: odbc.defaultlrl = 4096
743:
744: ; Handling of binary data.  0 means passthru, 1 return as is, 2 convert to char.
745: ; See the documentation on odbc_binmode and odbc_longreadlen for an explanation
746: ; of uodbc.defaultlrl and uodbc.defaultbinmode
747: odbc.defaultbinmode = 1
748:
749: [MySQL]
750: ; Allow or prevent persistent links.
```

```
751: mysql.allow_persistent = On
752:
753: ; Maximum number of persistent links.  -1 means no limit.
754: mysql.max_persistent = -1
755:
756: ; Maximum number of links (persistent + non-persistent).  -1 means no limit.
757: mysql.max_links = -1
758:
759: ; Default port number for mysql_connect().  If unset, mysql_connect() will use
760: ; the $MYSQL_TCP_PORT or the mysql-tcp entry in /etc/services or the
761: ; compile-time value defined MYSQL_PORT (in that order).  Win32 will only look
762: ; at MYSQL_PORT.
763: mysql.default_port =
764:
765: ; Default socket name for local MySQL connects.  If empty, uses the built-in
766: ; MySQL defaults.
767: mysql.default_socket =
768:
769: ; Default host for mysql_connect() (doesn't apply in safe mode).
770: mysql.default_host =
771:
772: ; Default user for mysql_connect() (doesn't apply in safe mode).
773: mysql.default_user =
774:
775: ; Default password for mysql_connect() (doesn't apply in safe mode).
776: ; Note that this is generally a *bad* idea to store passwords in this file.
777: ; *Any* user with PHP access can run 'echo get_cfg_var("mysql.default_password")
778: ; and reveal this password!  And of course, any users with read access to this
779: ; file will be able to reveal the password as well.
780: mysql.default_password =
781:
782: ; Maximum time (in seconds) for connect timeout. -1 means no limit
783: mysql.connect_timeout = 60
784:
785: ; Trace mode. When trace_mode is active (=On), warnings for table/index scans and
786: ; SQL-Errors will be displayed.
787: mysql.trace_mode = Off
788:
789: [MySQLi]
790:
791: ; Maximum number of links.  -1 means no limit.
792: mysqli.max_links = -1
793:
794: ; Default port number for mysqli_connect().  If unset, mysqli_connect() will use
795: ; the $MYSQL_TCP_PORT or the mysql-tcp entry in /etc/services or the
796: ; compile-time value defined MYSQL_PORT (in that order).  Win32 will only look
797: ; at MYSQL_PORT.
798: mysqli.default_port = 3306
799:
800: ; Default socket name for local MySQL connects.  If empty, uses the built-in
801: ; MySQL defaults.
802: mysqli.default_socket =
803:
804: ; Default host for mysql_connect() (doesn't apply in safe mode).
805: mysqli.default_host =
806:
807: ; Default user for mysql_connect() (doesn't apply in safe mode).
808: mysqli.default_user =
```

```
809:
810: ; Default password for mysqli_connect() (doesn't apply in safe mode).
811: ; Note that this is generally a *bad* idea to store passwords in this file.
812: ; *Any* user with PHP access can run 'echo get_cfg_var("mysqli.default_pw")
813: ; and reveal this password!  And of course, any users with read access to this
814: ; file will be able to reveal the password as well.
815: mysqli.default_pw =
816:
817: ; Allow or prevent reconnect
818: mysqli.reconnect = Off
819:
820: [mSQL]
821: ; Allow or prevent persistent links.
822: msql.allow_persistent = On
823:
824: ; Maximum number of persistent links.  -1 means no limit.
825: msql.max_persistent = -1
826:
827: ; Maximum number of links (persistent+non persistent).  -1 means no limit.
828: msql.max_links = -1
829:
830: [OCI8]
831: ; enables privileged connections using external credentials (OCI_SYSOPER,
     OCI_SYSDBA)
832: ;oci8.privileged_connect = Off
833:
834: ; Connection: The maximum number of persistent OCI8 connections per
835: ; process. Using -1 means no limit.
836: ;oci8.max_persistent = -1
837:
838: ; Connection: The maximum number of seconds a process is allowed to
839: ; maintain an idle persistent connection. Using -1 means idle
840: ; persistent connections will be maintained forever.
841: ;oci8.persistent_timeout = -1
842:
843: ; Connection: The number of seconds that must pass before issuing a
844: ; ping during oci_pconnect() to check the connection validity. When
845: ; set to 0, each oci_pconnect() will cause a ping. Using -1 disables
846: ; pings completely.
847: ;oci8.ping_interval = 60
848:
849: ; Tuning: This option enables statement caching, and specifies how
850: ; many statements to cache. Using 0 disables statement caching.
851: ;oci8.statement_cache_size = 20
852:
853: ; Tuning: Enables statement prefetching and sets the default number of
854: ; rows that will be fetched automatically after statement execution.
855: ;oci8.default_prefetch = 10
856:
857: ; Compatibility. Using On means oci_close() will not close
858: ; oci_connect() and oci_new_connect() connections.
859: ;oci8.old_oci_close_semantics = Off
860:
861: [PostgresSQL]
862: ; Allow or prevent persistent links.
863: pgsql.allow_persistent = On
864:
865: ; Detect broken persistent links always with pg_pconnect().
```

```
866: ; Auto reset feature requires a little overheads.
867: pgsql.auto_reset_persistent = Off
868:
869: ; Maximum number of persistent links.  -1 means no limit.
870: pgsql.max_persistent = -1
871:
872: ; Maximum number of links (persistent+non persistent).  -1 means no limit.
873: pgsql.max_links = -1
874:
875: ; Ignore PostgreSQL backends Notice message or not.
876: ; Notice message logging require a little overheads.
877: pgsql.ignore_notice = 0
878:
879: ; Log PostgreSQL backends Noitce message or not.
880: ; Unless pgsql.ignore_notice=0, module cannot log notice message.
881: pgsql.log_notice = 0
882:
883: [Sybase]
884: ; Allow or prevent persistent links.
885: sybase.allow_persistent = On
886:
887: ; Maximum number of persistent links.  -1 means no limit.
888: sybase.max_persistent = -1
889:
890: ; Maximum number of links (persistent + non-persistent).  -1 means no limit.
891: sybase.max_links = -1
892:
893: ;sybase.interface_file = "/usr/sybase/interfaces"
894:
895: ; Minimum error severity to display.
896: sybase.min_error_severity = 10
897:
898: ; Minimum message severity to display.
899: sybase.min_message_severity = 10
900:
901: ; Compatibility mode with old versions of PHP 3.0.
902: ; If on, this will cause PHP to automatically assign types to results according
903: ; to their Sybase type, instead of treating them all as strings.  This
904: ; compatibility mode will probably not stay around forever, so try applying
905: ; whatever necessary changes to your code, and turn it off.
906: sybase.compatability_mode = Off
907:
908: [Sybase-CT]
909: ; Allow or prevent persistent links.
910: sybct.allow_persistent = On
911:
912: ; Maximum number of persistent links.  -1 means no limit.
913: sybct.max_persistent = -1
914:
915: ; Maximum number of links (persistent + non-persistent).  -1 means no limit.
916: sybct.max_links = -1
917:
918: ; Minimum server message severity to display.
919: sybct.min_server_severity = 10
920:
921: ; Minimum client message severity to display.
922: sybct.min_client_severity = 10
923:
```

```
924: [bcmath]
925: ; Number of decimal digits for all bcmath functions.
926: bcmath.scale = 0
927:
928: [browscap]
929: ;browscap = extra/browscap.ini
930:
931: [Informix]
932: ; Default host for ifx_connect() (doesn't apply in safe mode).
933: ifx.default_host =
934:
935: ; Default user for ifx_connect() (doesn't apply in safe mode).
936: ifx.default_user =
937:
938: ; Default password for ifx_connect() (doesn't apply in safe mode).
939: ifx.default_password =
940:
941: ; Allow or prevent persistent links.
942: ifx.allow_persistent = On
943:
944: ; Maximum number of persistent links.  -1 means no limit.
945: ifx.max_persistent = -1
946:
947: ; Maximum number of links (persistent + non-persistent).  -1 means no limit.
948: ifx.max_links = -1
949:
950: ; If on, select statements return the contents of a text blob instead of its id.
951: ifx.textasvarchar = 0
952:
953: ; If on, select statements return the contents of a byte blob instead of its id.
954: ifx.byteasvarchar = 0
955:
956: ; Trailing blanks are stripped from fixed-length char columns.  May help the
957: ; life of Informix SE users.
958: ifx.charasvarchar = 0
959:
960: ; If on, the contents of text and byte blobs are dumped to a file instead of
961: ; keeping them in memory.
962: ifx.blobinfile = 0
963:
964: ; NULL's are returned as empty strings, unless this is set to 1.  In that case,
965: ; NULL's are returned as string 'NULL'.
966: ifx.nullformat = 0
967:
968: [Session]
969: ; Handler used to store/retrieve data.
970: session.save_handler = files
971:
972: ; Argument passed to save_handler.  In the case of files, this is the path
973: ; where data files are stored. Note: Windows users have to change this
974: ; variable in order to use PHP's session functions.
975: ;
976: ; As of PHP 4.0.1, you can define the path as:
977: ;
978: ;     session.save_path = "N;/path"
979: ;
980: ; where N is an integer.  Instead of storing all the session files in
981: ; /path, what this will do is use subdirectories N-levels deep, and
```

```
; store the session data in those directories.  This is useful if you
; or your OS have problems with lots of files in one directory, and is
; a more efficient layout for servers that handle lots of sessions.
;
; NOTE 1: PHP will not create this directory structure automatically.
;         You can use the script in the ext/session dir for that purpose.
; NOTE 2: See the section on garbage collection below if you choose to
;         use subdirectories for session storage
;
; The file storage module creates files using mode 600 by default.
; You can change that by using
;
;     session.save_path = "N;MODE;/path"
;
; where MODE is the octal representation of the mode. Note that this
; does not overwrite the process's umask.
;session.save_path = "/tmp"

; Whether to use cookies.
session.use_cookies = 1

;session.cookie_secure =

; This option enables administrators to make their users invulnerable to
; attacks which involve passing session ids in URLs; defaults to 0.
; session.use_only_cookies = 1

; Name of the session (used as cookie name).
session.name = PHPSESSID

; Initialize session on request startup.
session.auto_start = 0

; Lifetime in seconds of cookie or, if 0, until browser is restarted.
session.cookie_lifetime = 0

; The path for which the cookie is valid.
session.cookie_path = /

; The domain for which the cookie is valid.
session.cookie_domain =

; Whether or not to add the httpOnly flag to the cookie, which makes it
inaccessible to browser scripting languages such as JavaScript.
session.cookie_httponly =

; Handler used to serialize data.  php is the standard serializer of PHP.
session.serialize_handler = php

; Define the probability that the 'garbage collection' process is started
; on every session initialization.
; The probability is calculated by using gc_probability/gc_divisor,
; e.g. 1/100 means there is a 1% chance that the GC process starts
; on each request.

session.gc_probability = 1
session.gc_divisor     = 100

```

```
1039: ; After this number of seconds, stored data will be seen as 'garbage' and
1040: ; cleaned up by the garbage collection process.
1041: session.gc_maxlifetime = 1440
1042:
1043: ; NOTE: If you are using the subdirectory option for storing session files
1044: ;       (see session.save_path above), then garbage collection does *not*
1045: ;       happen automatically.  You will need to do your own garbage
1046: ;       collection through a shell script, cron entry, or some other method.
1047: ;       For example, the following script would is the equivalent of
1048: ;       setting session.gc_maxlifetime to 1440 (1440 seconds = 24 minutes):
1049: ;          cd /path/to/sessions; find -cmin +24 | xargs rm
1050:
1051: ; PHP 4.2 and less have an undocumented feature/bug that allows you to
1052: ; to initialize a session variable in the global scope, albeit register_globals
1053: ; is disabled.  PHP 4.3 and later will warn you, if this feature is used.
1054: ; You can disable the feature and the warning separately. At this time,
1055: ; the warning is only displayed, if bug_compat_42 is enabled.
1056:
1057: session.bug_compat_42 = 1
1058: session.bug_compat_warn = 1
1059:
1060: ; Check HTTP Referer to invalidate externally stored URLs containing ids.
1061: ; HTTP_REFERER has to contain this substring for the session to be
1062: ; considered as valid.
1063: session.referer_check =
1064:
1065: ; How many bytes to read from the file.
1066: session.entropy_length = 0
1067:
1068: ; Specified here to create the session id.
1069: session.entropy_file =
1070:
1071: ;session.entropy_length = 16
1072:
1073: ;session.entropy_file = /dev/urandom
1074:
1075: ; Set to {nocache,private,public,} to determine HTTP caching aspects
1076: ; or leave this empty to avoid sending anti-caching headers.
1077: session.cache_limiter = nocache
1078:
1079: ; Document expires after n minutes.
1080: session.cache_expire = 180
1081:
1082: ; trans sid support is disabled by default.
1083: ; Use of trans sid may risk your users security.
1084: ; Use this option with caution.
1085: ; - User may send URL contains active session ID
1086: ;   to other person via. email/irc/etc.
1087: ; - URL that contains active session ID may be stored
1088: ;   in publically accessible computer.
1089: ; - User may access your site with the same session ID
1090: ;   always using URL stored in browser's history or bookmarks.
1091: session.use_trans_sid = 0
1092:
1093: ; Select a hash function
1094: ; 0: MD5   (128 bits)
1095: ; 1: SHA-1 (160 bits)
1096: session.hash_function = 0
```

```
1097:
1098: ; Define how many bits are stored in each character when converting
1099: ; the binary hash data to something readable.
1100: ;
1101: ; 4 bits: 0-9, a-f
1102: ; 5 bits: 0-9, a-v
1103: ; 6 bits: 0-9, a-z, A-Z, "-", ","
1104: session.hash_bits_per_character = 4
1105:
1106: ; The URL rewriter will look for URLs in a defined set of HTML tags.
1107: ; form/fieldset are special; if you include them here, the rewriter will
1108: ; add a hidden <input> field with the info which is otherwise appended
1109: ; to URLs.  If you want XHTML conformity, remove the form entry.
1110: ; Note that all valid entries require a "=", even if no value follows.
1111: url_rewriter.tags = "a=href,area=href,frame=src,input=src,form=,fieldset="
1112:
1113: [MSSQL]
1114: ; Allow or prevent persistent links.
1115: mssql.allow_persistent = On
1116:
1117: ; Maximum number of persistent links.  -1 means no limit.
1118: mssql.max_persistent = -1
1119:
1120: ; Maximum number of links (persistent+non persistent).  -1 means no limit.
1121: mssql.max_links = -1
1122:
1123: ; Minimum error severity to display.
1124: mssql.min_error_severity = 10
1125:
1126: ; Minimum message severity to display.
1127: mssql.min_message_severity = 10
1128:
1129: ; Compatibility mode with old versions of PHP 3.0.
1130: mssql.compatability_mode = Off
1131:
1132: ; Connect timeout
1133: ;mssql.connect_timeout = 5
1134:
1135: ; Query timeout
1136: ;mssql.timeout = 60
1137:
1138: ; Valid range 0 - 2147483647.  Default = 4096.
1139: ;mssql.textlimit = 4096
1140:
1141: ; Valid range 0 - 2147483647.  Default = 4096.
1142: ;mssql.textsize = 4096
1143:
1144: ; Limits the number of records in each batch.  0 = all records in one batch.
1145: ;mssql.batchsize = 0
1146:
1147: ; Specify how datetime and datetim4 columns are returned
1148: ; On => Returns data converted to SQL server settings
1149: ; Off => Returns values as YYYY-MM-DD hh:mm:ss
1150: ;mssql.datetimeconvert = On
1151:
1152: ; Use NT authentication when connecting to the server
1153: mssql.secure_connection = Off
1154:
```

```
1155: ; Specify max number of processes. -1 = library default
1156: ; msdlib defaults to 25
1157: ; FreeTDS defaults to 4096
1158: ;mssql.max_procs = -1
1159:
1160: ; Specify client character set.
1161: ; If empty or not set the client charset from freetds.comf is used
1162: ; This is only used when compiled with FreeTDS
1163: ;mssql.charset = "ISO-8859-1"
1164:
1165: [Assertion]
1166: ; Assert(expr); active by default.
1167: ;assert.active = On
1168:
1169: ; Issue a PHP warning for each failed assertion.
1170: ;assert.warning = On
1171:
1172: ; Don't bail out by default.
1173: ;assert.bail = Off
1174:
1175: ; User-function to be called if an assertion fails.
1176: ;assert.callback = 0
1177:
1178: ; Eval the expression with current error_reporting().  Set to true if you want
1179: ; error_reporting(0) around the eval().
1180: ;assert.quiet_eval = 0
1181:
1182: [COM]
1183: ; path to a file containing GUIDs, IIDs or filenames of files with TypeLibs
1184: ;com.typelib_file =
1185: ; allow Distributed-COM calls
1186: ;com.allow_dcom = true
1187: ; autoregister constants of a components typlib on com_load()
1188: ;com.autoregister_typelib = true
1189: ; register constants casesensitive
1190: ;com.autoregister_casesensitive = false
1191: ; show warnings on duplicate constant registrations
1192: ;com.autoregister_verbose = true
1193:
1194: [mbstring]
1195: ; language for internal character representation.
1196: ;mbstring.language = Japanese
1197:
1198: ; internal/script encoding.
1199: ; Some encoding cannot work as internal encoding.
1200: ; (e.g. SJIS, BIG5, ISO-2022-*)
1201: ;mbstring.internal_encoding = EUC-JP
1202:
1203: ; http input encoding.
1204: ;mbstring.http_input = auto
1205:
1206: ; http output encoding. mb_output_handler must be
1207: ; registered as output buffer to function
1208: ;mbstring.http_output = SJIS
1209:
1210: ; enable automatic encoding translation according to
1211: ; mbstring.internal_encoding setting. Input chars are
1212: ; converted to internal encoding by setting this to On.
```

```
; Note: Do _not_ use automatic encoding translation for
;       portable libs/applications.
;mbstring.encoding_translation = Off

; automatic encoding detection order.
; auto means
;mbstring.detect_order = auto

; substitute_character used when character cannot be converted
; one from another
;mbstring.substitute_character = none;

; overload(replace) single byte functions by mbstring functions.
; mail(), ereg(), etc are overloaded by mb_send_mail(), mb_ereg(),
; etc. Possible values are 0,1,2,4 or combination of them.
; For example, 7 for overload everything.
; 0: No overload
; 1: Overload mail() function
; 2: Overload str*() functions
; 4: Overload ereg*() functions
;mbstring.func_overload = 0

[FrontBase]
;fbsql.allow_persistent = On
;fbsql.autocommit = On
;fbsql.show_timestamp_decimals = Off
;fbsql.default_database =
;fbsql.default_database_password =
;fbsql.default_host =
;fbsql.default_password =
;fbsql.default_user = "_SYSTEM"
;fbsql.generate_warnings = Off
;fbsql.max_connections = 128
;fbsql.max_links = 128
;fbsql.max_persistent = -1
;fbsql.max_results = 128

[gd]
; Tell the jpeg decode to libjpeg warnings and try to create
; a gd image. The warning will then be displayed as notices
; disabled by default
;gd.jpeg_ignore_warning = 0

[exif]
; Exif UNICODE user comments are handled as UCS-2BE/UCS-2LE and JIS as JIS.
; With mbstring support this will automatically be converted into the encoding
; given by corresponding encode setting. When empty mbstring.internal_encoding
; is used. For the decode settings you can distinguish between motorola and
; intel byte order. A decode setting cannot be empty.
;exif.encode_unicode = ISO-8859-15
;exif.decode_unicode_motorola = UCS-2BE
;exif.decode_unicode_intel    = UCS-2LE
;exif.encode_jis =
;exif.decode_jis_motorola = JIS
;exif.decode_jis_intel    = JIS

[Tidy]
; The path to a default tidy configuration file to use when using tidy
```

```
;tidy.default_config = /usr/local/lib/php/default.tcfg

; Should tidy clean and repair output automatically?
; WARNING: Do not use this option if you are generating non-html content
; such as dynamic images
tidy.clean_output = Off

[soap]
; Enables or disables WSDL caching feature.
soap.wsdl_cache_enabled=1
; Sets the directory name where SOAP extension will put cache files.
soap.wsdl_cache_dir="/tmp"
; (time to live) Sets the number of second while cached file will be used
; instead of original one.
soap.wsdl_cache_ttl=86400

; Local Variables:
; tab-width: 4
; End:

```