Multibyte String

Introduction

While there are many languages in which every necessary character can be represented by a one-to-one mapping to an 8-bit value, there are also several languages which require so many characters for written communication that they cannot be contained within the range a mere byte can code (A byte is made up of eight bits. Each bit can contain only two distinct values, one or zero. Because of this, a byte can only represent 256 unique values (two to the power of eight)). 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. Otherwise, if you apply a non-multibyte-aware string function to the string, it probably fails to detect the beginning or ending of the multibyte character and ends up with a corrupted garbage string that most likely loses its original meaning.

mbstring provides multibyte specific string functions that help you deal with multibyte encodings in PHP. In addition to that, *mbstring* handles character encoding conversion between the possible encoding pairs. *mbstring* is designed to handle Unicode-based encodings such as UTF-8 and UCS-2 and many single-byte encodings for convenience (listed below).

Installing/Configuring

Requirements

No external libraries are needed to build this extension.

Installation

mbstring is a non-default extension. This means it is not enabled by default. You must explicitly enable the module with the *configure* option. See the Install section for details.

The following configure options are related to the *mbstring* module.

- *--enable-mbstring*: Enable *mbstring* functions. This option is required to use *mbstring* functions. libmbfl is necesarry for *mbstring*. libmbfl is bundled with *mbstring*. If libmbfl is already installed on the system, *--with-libmbfl[=DIR]* can be specified to use the installed library. As of PHP 4.3.0, *mbstring* extension provides enhanced support for Simplified Chinese, Traditional Chinese, Korean, and Russian in addition to Japanese. For PHP 4.3.3 or before, To enable that feature, you will have to supply either one of the following options to the *LANG* parameter of *--enable-mbstring=LANG*; *--enable-mbstring=cn* for Simplified Chinese support, *--enable-mbstring=tw* for Traditional Chinese support, *--enable-mbstring=kr* for Korean support, *--enable-mbstring=ru* for Russian support, and *--enable-mbstring=ja* for Japanese support (default). To enable all supported encoding, use *--enable-mbstring=all*.

Note
As of PHP 4.3.4, all supported encoding by libmbfl is enabled with *--enable-mbstring*.

- *--enable-mbstr-enc-trans*: Enable HTTP input character encoding conversion using *mbstring* conversion engine. If this feature is enabled, HTTP input character encoding may be converted to *mbstring.internal_encoding* automatically.

Note
As of PHP 4.3.0, the option *--enable-mbstr-enc-trans* was eliminated and replaced with the runtime setting *mbstring.encoding_translation*. HTTP input character encoding conversion is enabled when this is set to *On* (the default is *Off*).

- *--disable-mbregex*: Disable regular expression functions with multibyte character support.

Runtime Configuration

The behaviour of these functions is affected by settings in *php.ini*.

mbstring configuration options

Name	Default	Changeable	Changelog
mbstring.language	"neutral"	PHP_INI_PERDIR	Available since PHP 4.3.0.
mbstring.detect_order	NULL	PHP_INI_ALL	Available since PHP 4.0.6.
mbstring.http_input	"pass"	PHP_INI_ALL	Available since PHP 4.0.6.
mbstring.http_output	"pass"	PHP_INI_ALL	Available since PHP 4.0.6.
mbstring.internal_encoding	NULL	PHP_INI_ALL	Available since PHP 4.0.6.
mbstring.script_encoding	NULL	PHP_INI_ALL	Available since PHP 4.3.0.
mbstring.substitute_character	NULL	PHP_INI_ALL	Available since PHP 4.0.6.
mbstring.func_overload	"0"	PHP_INI_PERDIR	PHP_INI_SYSTEM in PHP <= 4.2.3. Available since PHP 4.2.0.
mbstring.encoding_translation	"0"	PHP_INI_PERDIR	Available since PHP 4.3.0.
mbstring.strict_detection	"0"	PHP_INI_ALL	Available since PHP 5.1.2.

For the definition of the PHP_INI_* constants, please refer to ini_set().

Here's a short explanation of the configuration directives.

`mbstring.language` string
: The default national language setting (NLS) used in mbstring. Note that this option automagically defines *mbstring.internal_encoding* and *mbstring.internal_encoding* should be placed after *mbstring.language* in *php.ini*

mbstring.encoding_translation boolean
Enables the transparent character encoding filter for the incoming HTTP queries, which performs detection and conversion of the input encoding to the internal character encoding.

mbstring.internal_encoding string
Defines the default internal character encoding.

mbstring.http_input string
Defines the default HTTP input character encoding.

mbstring.http_output string
Defines the default HTTP output character encoding.

mbstring.detect_order string
Defines default character code detection order. See also mb_detect_order().

mbstring.substitute_character string
Defines character to substitute for invalid character encoding.

mbstring.func_overload string
Overloads a set of single byte functions by the mbstring counterparts. See Function overloading for more information.

mbstring.strict_detection boolean
Enables the strict encoding detection.

According to the » HTML 4.01 specification, Web browsers are allowed to encode a form being submitted with a character encoding different from the one used for the page. See mb_http_input() to detect character encoding used by browsers.

Although popular browsers are capable of giving a reasonably accurate guess to the character encoding of a given HTML document, it would be better to set the *charset* parameter in the *Content-Type* HTTP header to the appropriate value by header() or default_charset ini setting.

Example #1 - *php.ini* setting examples

```
; Set default language
mbstring.language        = Neutral; Set default language to Neutral(UTF-8)
(default)
mbstring.language        = English; Set default language to English
mbstring.language        = Japanese; Set default language to Japanese

;; Set default internal encoding
;; Note: Make sure to use character encoding works with PHP
mbstring.internal_encoding    = UTF-8  ; Set internal encoding to UTF-8

;; HTTP input encoding translation is enabled.
mbstring.encoding_translation = On
```

```
;; Set default HTTP input character encoding
;; Note: Script cannot change http_input setting.
mbstring.http_input           = pass    ; No conversion.
mbstring.http_input           = auto    ; Set HTTP input to auto
                                ; "auto" is expanded to
"ASCII,JIS,UTF-8,EUC-JP,SJIS"
mbstring.http_input           = SJIS    ; Set HTTP2 input to  SJIS
mbstring.http_input           = UTF-8,SJIS,EUC-JP ; Specify order

;; Set default HTTP output character encoding
mbstring.http_output          = pass    ; No conversion
mbstring.http_output          = UTF-8   ; Set HTTP output encoding to UTF-8

;; Set default character encoding detection order
mbstring.detect_order         = auto    ; Set detect order to auto
mbstring.detect_order         = ASCII,JIS,UTF-8,SJIS,EUC-JP ; Specify order

;; Set default substitute character
mbstring.substitute_character = 12307   ; Specify Unicode value
mbstring.substitute_character = none    ; Do not print character
mbstring.substitute_character = long    ; Long Example: U+3000,JIS+7E7E
```

Example #2 - *php.ini* **setting for** *EUC-JP* **users**

```
;; Disable Output Buffering
output_buffering      = Off

;; Set HTTP header charset
default_charset       = EUC-JP

;; Set default language to Japanese
mbstring.language = Japanese

;; HTTP input encoding translation is enabled.
mbstring.encoding_translation = On

;; Set HTTP input encoding conversion to auto
mbstring.http_input   = auto

;; Convert HTTP output to EUC-JP
mbstring.http_output  = EUC-JP

;; Set internal encoding to EUC-JP
mbstring.internal_encoding = EUC-JP

;; Do not print invalid characters
mbstring.substitute_character = none
```

Example #3 - *php.ini* setting for *SJIS* users

```
;; Enable Output Buffering
output_buffering     = On

;; Set mb_output_handler to enable output conversion
output_handler       = mb_output_handler

;; Set HTTP header charset
default_charset      = Shift_JIS

;; Set default language to Japanese
mbstring.language = Japanese

;; Set http input encoding conversion to auto
mbstring.http_input  = auto

;; Convert to SJIS
mbstring.http_output = SJIS

;; Set internal encoding to EUC-JP
mbstring.internal_encoding = EUC-JP

;; Do not print invalid characters
mbstring.substitute_character = none
```

Resource Types

This extension has no resource types defined.

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

MB_OVERLOAD_MAIL (integer)

MB_OVERLOAD_STRING (integer)

MB_OVERLOAD_REGEX (integer)

MB_CASE_UPPER (integer)

MB_CASE_LOWER (integer)

MB_CASE_TITLE (integer)

Summaries of supported encodings

Summaries of supported encodings
Name in the IANA character set registry: ISO-10646-UCS-4
Underlying character set: ISO 10646
Description: The Universal Character Set with 31-bit code space, standardized as UCS-4 by ISO/IEC 10646. It is kept synchronized with the latest version of the Unicode code map.
Additional note: If this name is used in the encoding conversion facility, the converter attempts to identify by the preceding BOM (byte order mark)in which endian the subsequent bytes are represented.
Name in the IANA character set registry: ISO-10646-UCS-4
Underlying character set: UCS-4
Description: See above.
Additional note: In contrast to *UCS-4*, strings are always assumed to be in big endian form.
Name in the IANA character set registry: ISO-10646-UCS-4
Underlying character set: UCS-4
Description: See above.
Additional note: In contrast to *UCS-4*, strings are always assumed to be in little endian form.
Name in the IANA character set registry: ISO-10646-UCS-2
Underlying character set: UCS-2
Description: The Universal Character Set with 16-bit code space, standardized as UCS-2 by ISO/IEC 10646. It is kept synchronized with the latest version of the unicode code map.
Additional note: If this name is used in the encoding conversion facility, the converter attempts to identify by the preceding BOM (byte order mark)in which endian the subsequent bytes are represented.
Name in the IANA character set registry: ISO-10646-UCS-2
Underlying character set: UCS-2
Description: See above.
Additional note: In contrast to *UCS-2*, strings are always assumed to be in big endian form.
Name in the IANA character set registry: ISO-10646-UCS-2
Underlying character set: UCS-2
Description: See above.
Additional note: In contrast to *UCS-2*, strings are always assumed to be in little endian form.
Name in the IANA character set registry: UTF-32
Underlying character set: Unicode
Description: Unicode Transformation Format of 32-bit unit width, whose encoding space refers to the Unicode's codeset standard. This encoding scheme wasn't identical to UCS-4 because the code space of Unicode were limited to a 21-bit value.
Additional note: If this name is used in the encoding conversion facility, the converter attempts to identify by the preceding BOM (byte order mark)in which endian the subsequent bytes are represented.
Name in the IANA character set registry: UTF-32BE
Underlying character set: Unicode
Description: See above
Additional note: In contrast to *UTF-32*, strings are always assumed to be in big endian form.

Name in the IANA character set registry: UTF-32LE
Underlying character set: Unicode
Description: See above
Additional note: In contrast to *UTF-32*, strings are always assumed to be in little endian form.
Name in the IANA character set registry: UTF-16
Underlying character set: Unicode
Description: Unicode Transformation Format of 16-bit unit width. It's worth a note that UTF-16 is no longer the same specification as UCS-2 because the surrogate mechanism has been introduced since Unicode 2.0 and UTF-16 now refers to a 21-bit code space.
Additional note: If this name is used in the encoding conversion facility, the converter attempts to identify by the preceding BOM (byte order mark)in which endian the subsequent bytes are represented.
Name in the IANA character set registry: UTF-16BE
Underlying character set: Unicode
Description: See above.
Additional note: In contrast to *UTF-16*, strings are always assumed to be in big endian form.
Name in the IANA character set registry: UTF-16LE
Underlying character set: Unicode
Description: See above.
Additional note: In contrast to *UTF-16*, strings are always assumed to be in little endian form.
Name in the IANA character set registry: UTF-8
Underlying character set: Unicode / UCS
Description: Unicode Transformation Format of 8-bit unit width.
Additional note: none
Name in the IANA character set registry: UTF-7
Underlying character set: Unicode
Description: A mail-safe transformation format of Unicode, specified in » RFC2152.
Additional note: none
Name in the IANA character set registry: (none)
Underlying character set: Unicode
Description: A variant of UTF-7 which is specialized for use in the » IMAP protocol.
Additional note: none
Name in the IANA character set registry: US-ASCII (preferred MIME name) / iso-ir-6 / ANSI_X3.4-1986 / ISO_646.irv:1991 / ASCII / ISO646-US / us / IBM367 / CP367 / csASCII
Underlying character set: ASCII / ISO 646
Description: American Standard Code for Information Interchange is a commonly-used 7-bit encoding. Also standardized as an international standard, ISO 646.
Additional note: (none)
Name in the IANA character set registry: EUC-JP (preferred MIME name) / Extended_UNIX_Code_Packed_Format_for_Japanese / csEUCPkdFmtJapanese
Underlying character set: Compound of US-ASCII / JIS X0201:1997 (hankaku kana part) / JIS X0208:1990 / JIS X0212:1990
Description: As you see the name is derived from an abbreviation of Extended UNIX Code Packed Format for Japanese, this encoding is mostly used on UNIX or alike platforms. The original encoding scheme, Extended UNIX Code, is designed on the basis of ISO 2022.
Additional note: The character set referred to by EUC-JP is different to IBM932 / CP932, which are used by OS/2® and Microsoft® Windows®. For information interchange with those platforms, use EUCJP-WIN instead.

Name in the IANA character set registry: Shift_JIS (preferred MIME name) / MS_Kanji / csShift_JIS
Underlying character set: Compound of JIS X0201:1997 / JIS X0208:1997
Description: Shift_JIS was developed in early 80's, at the time personal Japanese word processors were brought into the market, in order to maintain compatiblities with the legacy encoding scheme JIS X 0201:1976. According to the IANA definition the codeset of Shift_JIS is slightly different to IBM932 / CP932. However, the names "SJIS" / "Shift_JIS" are often wrongly used to refer to these codesets.
Additional note: For the CP932 codemap, use SJIS-WIN instead.
Name in the IANA character set registry: (none)
Underlying character set: Compound of JIS X0201:1997 / JIS X0208:1997 / IBM extensions / NEC extensions
Description: While this "encoding" uses the same encoding scheme as EUC-JP, the underlying character set is different. That is, some code points map to different characters than EUC-JP.
Additional note: none
Name in the IANA character set registry: Windows-31J / csWindows31J
Underlying character set: Compound of JIS X0201:1997 / JIS X0208:1997 / IBM extensions / NEC extensions
Description: While this "encoding" uses the same encoding scheme as Shift_JIS, the underlying character set is different. That means some code points map to different characters than Shift_JIS.
Additional note: (none)
Name in the IANA character set registry: ISO-2022-JP (preferred MIME name) / csISO2022JP
Underlying character set: US-ASCII / JIS X0201:1976 / JIS X0208:1978 / JIS X0208:1983
Description: » RFC1468
Additional note: (none)
Name in the IANA character set registry: JIS
Underlying character set:
Description:
Additional note:
Name in the IANA character set registry: ISO-8859-1
Underlying character set:
Description:
Additional note:
Name in the IANA character set registry: ISO-8859-2
Underlying character set:
Description:
Additional note:
Name in the IANA character set registry: ISO-8859-3
Underlying character set:
Description:
Additional note:
Name in the IANA character set registry: ISO-8859-4
Underlying character set:
Description:
Additional note:
Name in the IANA character set registry: ISO-8859-5
Underlying character set:
Description:

Additional note:
Name in the IANA character set registry: ISO-8859-6
Underlying character set:
Description:
Additional note:
Name in the IANA character set registry: ISO-8859-7
Underlying character set:
Description:
Additional note:
Name in the IANA character set registry: ISO-8859-8
Underlying character set:
Description:
Additional note:
Name in the IANA character set registry: ISO-8859-9
Underlying character set:
Description:
Additional note:
Name in the IANA character set registry: ISO-8859-10
Underlying character set:
Description:
Additional note:
Name in the IANA character set registry: ISO-8859-13
Underlying character set:
Description:
Additional note:
Name in the IANA character set registry: ISO-8859-14
Underlying character set:
Description:
Additional note:
Name in the IANA character set registry: ISO-8859-15
Underlying character set:
Description:
Additional note:
Name in the IANA character set registry: byte2be
Underlying character set:
Description:
Additional note:
Name in the IANA character set registry: byte2le
Underlying character set:
Description:
Additional note:
Name in the IANA character set registry: byte4be
Underlying character set:
Description:
Additional note:
Name in the IANA character set registry: byte4le
Underlying character set:
Description:
Additional note:
Name in the IANA character set registry: BASE64
Underlying character set:
Description:

Additional note:
Name in the IANA character set registry: HTML-ENTITIES
Underlying character set:
Description:
Additional note:
Name in the IANA character set registry: 7bit
Underlying character set:
Description:
Additional note:
Name in the IANA character set registry: 8bit
Underlying character set:
Description:
Additional note:
Name in the IANA character set registry: EUC-CN
Underlying character set:
Description:
Additional note:
Name in the IANA character set registry: CP936
Underlying character set:
Description:
Additional note:
Name in the IANA character set registry: HZ
Underlying character set:
Description:
Additional note:
Name in the IANA character set registry: EUC-TW
Underlying character set:
Description:
Additional note:
Name in the IANA character set registry: CP950
Underlying character set:
Description:
Additional note:
Name in the IANA character set registry: BIG-5
Underlying character set:
Description:
Additional note:
Name in the IANA character set registry: EUC-KR
Underlying character set:
Description:
Additional note:
Name in the IANA character set registry: UHC (CP949)
Underlying character set:
Description:
Additional note:
Name in the IANA character set registry: ISO-2022-KR
Underlying character set:
Description:
Additional note:
Name in the IANA character set registry: Windows-1251 (CP1251)
Underlying character set:
Description:

Additional note:
Name in the IANA character set registry: Windows-1252 (CP1252)
Underlying character set:
Description:
Additional note:
Name in the IANA character set registry: CP866 (IBM866)
Underlying character set:
Description:
Additional note:
Name in the IANA character set registry: KOI8-R
Underlying character set:
Description:
Additional note:

Basics of Japanese multi-byte encodings

Japanese characters can only be represented by multibyte encodings, and multiple encoding standards are used depending on platform and text purpose. To make matters worse, these encoding standards differ slightly from one another. In order to create a web application which would be usable in a Japanese environment, a developer has to keep these complexities in mind to ensure that the proper character encodings are used.

- Storage for a character can be up to six bytes
- Most Japanese multibyte characters appear twice as wide as single-byte characters. These characters are called "zen-kaku" in Japanese, which means "full width". Other, narrower, characters are called "han-kaku", which means "half width". The graphical properties of the characters, however, depends upon the type faces used to display them.
- Some character encodings use shift(escape) sequences defined in ISO-2022 to switch the code map of the specific code area (*00h* to *7fh*).
- ISO-2022-JP should be used in SMTP/NNTP, and headers and entities should be reencoded as per RFC requirements. Although those are not requisites, it's still a good idea because several popular user agents cannot recognize any other encoding methods.
- Web pages created for mobile phone services such as » i-mode, » Vodafone live!, or » EZweb are supposed to use Shift_JIS.

HTTP Input and Output

HTTP input/output character encoding conversion may convert binary data also. Users are supposed to control character encoding conversion if binary data is used for HTTP input/output.

Note

In PHP 4.3.2 or earlier versions, there was a limitation in this functionality that *mbstring* does not perform character encoding conversion in POST data if the *enctype* attribute in the *form* element is set to *multipart/form-data*. So you have to convert the incoming data by yourself in this case if necessary.

Beginning with PHP 4.3.3, if *enctype* for HTML form is set to *multipart/form-data* and *mbstring.encoding_translation* is set to On in *php.ini* the POST'ed variables and the names of uploaded files will be converted to the internal character encoding as well. However, the conversion isn't applied to the query keys.

- HTTP Input There is no way to control HTTP input character conversion from a PHP script. To disable HTTP input character conversion, it has to be done in *php.ini*.

  Example #4 - Disable HTTP input conversion in *php.ini*

  ```
  ;; Disable HTTP Input conversion
  mbstring.http_input = pass
  ;; Disable HTTP Input conversion (PHP 4.3.0 or higher)
  mbstring.encoding_translation = Off
  ```

  When using PHP as an Apache module, it is possible to override those settings in each Virtual Host directive in *httpd.conf* or per directory with*.htaccess*. Refer to the Configuration section and Apache Manual for details.
- HTTP Output There are several ways to enable output character encoding conversion. One is using *php.ini*, another is using ob_start() with mb_output_handler() as the *ob_start* callback function.

Example #5 - *php.ini* **setting example**

```
;; Enable output character encoding conversion for all PHP pages

;; Enable Output Buffering
output_buffering    = On

;; Set mb_output_handler to enable output conversion
```

```
output_handler      = mb_output_handler
```

Example #6 - Script example

```
<?php

// Enable output character encoding conversion only for this page

// Set HTTP output character encoding to SJIS
mb_http_output('SJIS');

// Start buffering and specify "mb_output_handler" as
// callback function
ob_start('mb_output_handler');

?>
```

Supported Character Encodings

Currently the following character encodings are supported by the *mbstring* module. Any of those Character encodings can be specified in the *encoding* parameter of *mbstring* functions.

The following character encodings are supported in this PHP extension:

- UCS-4
- UCS-4BE
- UCS-4LE
- UCS-2
- UCS-2BE
- UCS-2LE
- UTF-32
- UTF-32BE
- UTF-32LE
- UTF-16
- UTF-16BE
- UTF-16LE
- UTF-7
- UTF7-IMAP
- UTF-8
- ASCII
- EUC-JP
- SJIS
- eucJP-win
- SJIS-win
- ISO-2022-JP
- JIS
- ISO-8859-1
- ISO-8859-2
- ISO-8859-3
- ISO-8859-4
- ISO-8859-5

- ISO-8859-6
- ISO-8859-7
- ISO-8859-8
- ISO-8859-9
- ISO-8859-10
- ISO-8859-13
- ISO-8859-14
- ISO-8859-15
- byte2be
- byte2le
- byte4be
- byte4le
- BASE64
- HTML-ENTITIES
- 7bit
- 8bit
- EUC-CN
- CP936
- HZ
- EUC-TW
- CP950
- BIG-5
- EUC-KR
- UHC (CP949)
- ISO-2022-KR
- Windows-1251 (CP1251)
- Windows-1252 (CP1252)
- CP866 (IBM866)
- KOI8-R

Any *php.ini* entry which accepts an encoding name can also use the values " *auto* " and " *pass* ". *mbstring* functions which accept an encoding name can also use the value " *auto* ".

If " *pass* " is set, no character encoding conversion is performed.

If " *auto* " is set, it is expanded to the list of encodings defined per the NLS. For instance, if the NLS is set to *Japanese*, the value is assumed to be " *ASCII,JIS,UTF-8,EUC-JP,SJIS* ".

See also mb_detect_order()

Function Overloading Feature

You might often find it difficult to get an existing PHP application to work in a given multibyte environment. This happens because most PHP applications out there are written with the standard string functions such as substr(), which are known to not properly handle multibyte-encoded strings.

mbstring supports a 'function overloading' feature which enables you to add multibyte awareness to such an application without code modification by overloading multibyte counterparts on the standard string functions. For example, mb_substr() is called instead of substr() if function overloading is enabled. This feature makes it easy to port applications that only support single-byte encodings to a multibyte environment in many cases.

To use function overloading, set *mbstring.func_overload* in *php.ini* to a positive value that represents a combination of bitmasks specifying the categories of functions to be overloaded. It should be set to 1 to overload the mail() function. 2 for string functions, 4 for regular expression functions. For example, if it is set to 7, mail, strings and regular expression functions will be overloaded. The list of overloaded functions are shown below.

Functions to be overloaded

value of mbstring.func_overload	original function	overloaded function
1	mail()	mb_send_mail()
2	strlen()	mb_strlen()
2	strpos()	mb_strpos()
2	strrpos()	mb_strrpos()
2	substr()	mb_substr()
2	strtolower()	mb_strtolower()
2	strtoupper()	mb_strtoupper()
2	substr_count()	mb_substr_count()
4	ereg()	mb_ereg()
4	eregi()	mb_eregi()
4	ereg_replace()	mb_ereg_replace()
4	eregi_replace()	mb_eregi_replace()
4	split()	mb_split()

Note

It is not recommended to use the function overloading option in the per-directory context, because it's not confirmed yet to be stable enough in a production environment and may lead to undefined behaviour.

PHP Character Encoding Requirements

Encodings of the following types are safely used with PHP.

- A singlebyte encoding,
  - which has ASCII-compatible (ISO646 compatible) mappings for the characters in range of *00h* to *7fh*.
- A multibyte encoding,
  - which has ASCII-compatible mappings for the characters in range of *00h* to *7fh*.
  - which don't use ISO2022 escape sequences.
  - which don't use a value from *00h* to *7fh* in any of the compounded bytes that represents a single character.

These are examples of character encodings that are unlikely to work with PHP.

```
JIS, SJIS, ISO-2022-JP, BIG-5
```

Although PHP scripts written in any of those encodings might not work, especially in the case where encoded strings appear as identifiers or literals in the script, you can almost avoid using these encodings by setting up the *mbstring* 's transparent encoding filter function for incoming HTTP queries.

Note
It's highly discouraged to use SJIS, BIG5, CP936, CP949 and GB18030 for the internal encoding unless you are familiar with the parser, the scanner and the character encoding.

Note
If you are connecting to a database with PHP, it is recommended that you use the same character encoding for both the database and the *internal encoding* for ease of use and better performance. If you are using PostgreSQL, the character encoding used in the database and the one used in PHP may differ as it supports automatic character set conversion between the backend and the frontend.

Multibyte String Functions

References

Multibyte character encoding schemes and their related issues are fairly complicated, and are beyond the scope of this documentation. Please refer to the following URLs and other resources for further information regarding these topics.

- Unicode materials » http://www.unicode.org/
- Japanese/Korean/Chinese character information
  » http://examples.oreilly.com/cjkvinfo/doc/cjk.inf

mb_check_encoding

mb_check_encoding -- Check if the string is valid for the specified encoding

Description

bool **mb_check_encoding** ([string `$var` [, string `$encoding`]])

Checks if the specified byte stream is valid for the specified encoding. It is useful to prevent so-called "Invalid Encoding Attack".

Parameters

`var`
: The byte stream to check. If it is omitted, this function checks all the input from the beginning of the request.

`encoding`
: The expected encoding.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

mb_convert_case

mb_convert_case -- Perform case folding on a string

Description

string **mb_convert_case** (string `$str`, int `$mode` [, string `$encoding`])

Performs case folding on a string, converted in the way specified by *`mode`*.

Parameters

`str`
: The string being converted.

`mode`
: The mode of the conversion. It can be one of **MB_CASE_UPPER**, **MB_CASE_LOWER**, or **MB_CASE_TITLE**.

`encoding`
: The *`encoding`* parameter is the character encoding. If it is omitted, the internal character encoding value will be used.

Return Values

A case folded version of *`string`* converted in the way specified by *`mode`*.

Unicode

By contrast to the standard case folding functions such as strtolower() and strtoupper(), case folding is performed on the basis of the Unicode character properties. Thus the behaviour of this function is not affected by locale settings and it can convert any characters that have 'alphabetic' property, such as A-umlaut (Ä).

For more information about the Unicode properties, please see
» http://www.unicode.org/unicode/reports/tr21/.

Examples

Example #7 - mb_convert_case() example

```
<?php
$str = "mary had a Little lamb and she loved it so";
$str = mb_convert_case($str, MB_CASE_UPPER, "UTF-8");
echo $str; // Prints MARY HAD A LITTLE LAMB AND SHE LOVED IT SO
```

```
$str = mb_convert_case($str, MB_CASE_TITLE, "UTF-8");
echo $str; // Prints Mary Had A Little Lamb And She Loved It So
?>
```

See Also

- mb_strtolower()
- mb_strtoupper()
- strtolower()
- strtoupper()
- ucfirst()
- ucwords()

mb_convert_encoding

mb_convert_encoding -- Convert character encoding

Description

string **mb_convert_encoding** (string `$str`, string `$to_encoding` [, mixed `$from_encoding`])

Converts the character encoding of string *`str`* to *`to_encoding`* from optionally *`from_encoding`*.

Parameters

`str`
: The string being encoded.

`to_encoding`
: The type of encoding that *`str`* is being converted to.

`from_encoding`
: Is specified by character code names before conversion. It is either an array, or a comma separated enumerated list. If *`from_encoding`* is not specified, the internal encoding will be used. " *auto* " may be used, which expands to *"ASCII,JIS,UTF-8,EUC-JP,SJIS"*.

Return Values

The encoded string.

Examples

Example #8 - mb_convert_encoding() example

```
<?php
/* Convert internal character encoding to SJIS */
$str = mb_convert_encoding($str, "SJIS");

/* Convert EUC-JP to UTF-7 */
$str = mb_convert_encoding($str, "UTF-7", "EUC-JP");

/* Auto detect encoding from JIS, eucjp-win, sjis-win, then convert str to
UCS-2LE */
$str = mb_convert_encoding($str, "UCS-2LE", "JIS, eucjp-win, sjis-win");

/* "auto" is expanded to "ASCII,JIS,UTF-8,EUC-JP,SJIS" */
```

```
$str = mb_convert_encoding($str, "EUC-JP", "auto");
?>
```

See Also

- mb_detect_order()

mb_convert_kana

mb_convert_kana -- Convert "kana" one from another ("zen-kaku", "han-kaku" and more)

Description

string **mb_convert_kana** (string `$str` [, string `$option` [, string `$encoding`]])

Performs a "han-kaku" - "zen-kaku" conversion for string *`str`*. This function is only useful for Japanese.

Parameters

`str`
: The string being converted.

`option`
: The conversion option. The default value is *"KV"*. Specify with a combination of following options. The default value is *KV*.

Applicable Conversion Options

Option	Meaning
r	Convert "zen-kaku" alphabets to "han-kaku"
R	Convert "han-kaku" alphabets to "zen-kaku"
n	Convert "zen-kaku" numbers to "han-kaku"
N	Convert "han-kaku" numbers to "zen-kaku"
a	Convert "zen-kaku" alphabets and numbers to "han-kaku"
A	Convert "han-kaku" alphabets and numbers to "zen-kaku" (Characters included in "a", "A" options are U+0021 - U+007E excluding U+0022, U+0027, U+005C, U+007E)
s	Convert "zen-kaku" space to "han-kaku" (U+3000 -> U+0020)
S	Convert "han-kaku" space to "zen-kaku" (U+0020 -> U+3000)
k	Convert "zen-kaku kata-kana" to "han-kaku kata-kana"

K	Convert "han-kaku kata-kana" to "zen-kaku kata-kana"
h	Convert "zen-kaku hira-gana" to "han-kaku kata-kana"
H	Convert "han-kaku kata-kana" to "zen-kaku hira-gana"
c	Convert "zen-kaku kata-kana" to "zen-kaku hira-gana"
C	Convert "zen-kaku hira-gana" to "zen-kaku kata-kana"
V	Collapse voiced sound notation and convert them into a character. Use with "K","H"

`encoding`
: The *`encoding`* parameter is the character encoding. If it is omitted, the internal character encoding value will be used.

Return Values

The converted string.

Examples

Example #9 - mb_convert_kana() example

```
<?php
/* Convert all "kana" to "zen-kaku" "kata-kana" */
$str = mb_convert_kana($str, "KVC");

/* Convert "han-kaku" "kata-kana" to "zen-kaku" "kata-kana"
  and "zen-kaku" alpha-numeric to "han-kaku" */
$str = mb_convert_kana($str, "KVa");
?>
```

mb_convert_variables

mb_convert_variables -- Convert character code in variable(s)

Description

string **mb_convert_variables** (string `$to_encoding`, mixed `$from_encoding`, mixed `&$vars` [, mixed `&$...`])

Converts character encoding of variables *`vars`* in encoding *`from_encoding`* to encoding *`to_encoding`*.

mb_convert_variables() join strings in Array or Object to detect encoding, since encoding detection tends to fail for short strings. Therefore, it is impossible to mix encoding in single array or object.

Parameters

`to_encoding`
: The encoding that the string is being converted to.

`from_encoding`
: *`from_encoding`* is specified as an array or comma separated string, it tries to detect encoding from *`from-coding`*. When *`from_encoding`* is omitted, *detect_order* is used.

`vars`
: *`vars`* is the reference to the variable being converted. String, Array and Object are accepted. mb_convert_variables() assumes all parameters have the same encoding.

`...`
: Additional *`vars`*.

Return Values

The character encoding before conversion for success, or **FALSE** for failure.

Examples

Example #10 - mb_convert_variables() example

```
<?php
/* Convert variables $post1, $post2 to internal encoding */
$interenc = mb_internal_encoding();
$inputenc = mb_convert_variables($interenc, "ASCII,UTF-8,SJIS-win", $post1,
$post2);
```

```
?>
```

mb_decode_mimeheader

mb_decode_mimeheader -- Decode string in MIME header field

Description

string **mb_decode_mimeheader** (string `$str`)

Decodes encoded-word string *`str`* in MIME header.

Parameters

`str`
: The string being decoded.

Return Values

The decoded string in internal character encoding.

See Also

- mb_encode_mimeheader()

mb_decode_numericentity

mb_decode_numericentity -- Decode HTML numeric string reference to character

Description

string **mb_decode_numericentity** (string `$str`, array `$convmap` [, string `$encoding`])

Convert numeric string reference of string *`str`* in a specified block to character.

Parameters

`str`
: The string being decoded.

`convmap`
: *`convmap`* is an array that specifies the code area to convert.

`encoding`
: The *`encoding`* parameter is the character encoding. If it is omitted, the internal character encoding value will be used.

Return Values

The converted string.

Examples

Example #11 - *`convmap`* **example**

```
$convmap = array (
  int start_code1, int end_code1, int offset1, int mask1,
  int start_code2, int end_code2, int offset2, int mask2,
  ........
  int start_codeN, int end_codeN, int offsetN, int maskN );
// Specify Unicode value for start_codeN and end_codeN
// Add offsetN to value and take bit-wise 'AND' with maskN,
// then convert value to numeric string reference.
```

See Also

- mb_encode_numericentity()

mb_detect_encoding

mb_detect_encoding -- Detect character encoding

Description

string **mb_detect_encoding** (string `$str` [, mixed `$encoding_list` [, bool `$strict`]])

Detects character encoding in string *`str`*.

Parameters

`str`
The string being detected.

`encoding_list`
`encoding_list` is list of character encoding. Encoding order may be specified by array or comma separated list string. If *`encoding_list`* is omitted, detect_order is used.

`strict`
`strict` specifies whether to use the strict encoding detection or not. Default is **FALSE**.

Return Values

The detected character encoding.

Examples

Example #12 - mb_detect_encoding() example

```
<?php
/* Detect character encoding with current detect_order */
echo mb_detect_encoding($str);

/* "auto" is expanded to "ASCII,JIS,UTF-8,EUC-JP,SJIS" */
echo mb_detect_encoding($str, "auto");

/* Specify encoding_list character encoding by comma separated list */
echo mb_detect_encoding($str, "JIS, eucjp-win, sjis-win");

/* Use array to specify encoding_list  */
$ary[] = "ASCII";
$ary[] = "JIS";
$ary[] = "EUC-JP";
echo mb_detect_encoding($str, $ary);
?>
```

See Also

- mb_detect_order()

mb_detect_order

mb_detect_order -- Set/Get character encoding detection order

Description

mixed **mb_detect_order** ([mixed `$encoding_list`])

Sets the automatic character encoding detection order to *`encoding_list`*.

Parameters

`encoding_list`

`encoding_list` is an array or comma separated list of character encoding. ("auto" is expanded to "ASCII, JIS, UTF-8, EUC-JP, SJIS") If *`encoding_list`* is omitted, it returns the current character encoding detection order as array. This setting affects mb_detect_encoding() and mb_send_mail(). *mbstring* currently implements the following encoding detection filters. If there is an invalid byte sequence for the following encodings, encoding detection will fail. *UTF-8*, *UTF-7*, *ASCII*, *EUC-JP*, *SJIS*, *eucJP-win*, *SJIS-win*, *JIS* , *ISO-2022-JP* For *ISO-8859-**, *mbstring* always detects as *ISO-8859-**. For *UTF-16*, *UTF-32*, *UCS2* and *UCS4*, encoding detection will fail always.

Example #13 - Useless detect order example

```
; Always detect as ISO-8859-1
detect_order = ISO-8859-1, UTF-8

; Always detect as UTF-8, since ASCII/UTF-7 values are
; valid for UTF-8
detect_order = UTF-8, ASCII, UTF-7
```

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #14 - mb_detect_order() examples

```
<?php
/* Set detection order by enumerated list */
mb_detect_order("eucjp-win,sjis-win,UTF-8");

/* Set detection order by array */
```

```
$ary[] = "ASCII";
$ary[] = "JIS";
$ary[] = "EUC-JP";
mb_detect_order($ary);

/* Display current detection order */
echo implode(", ", mb_detect_order());
?>
```

See Also

- mb_internal_encoding()
- mb_http_input()
- mb_http_output()
- mb_send_mail()

mb_encode_mimeheader

mb_encode_mimeheader -- Encode string for MIME header

Description

string **mb_encode_mimeheader** (string `$str` [, string `$charset` [, string `$transfer_encoding` [, string `$linefeed` [, int `$indent`]]]])

Encodes a given string *`str`* by the MIME header encoding scheme.

Parameters

`str`
: The string being encoded.

`charset`
: *`charset`* specifies the name of the character set in which *`str`* is represented in. The default value is determined by the current NLS setting (*mbstring.language*).

`transfer_encoding`
: *`transfer_encoding`* specifies the scheme of MIME encoding. It should be either *"B"* (Base64) or *"Q"* (Quoted-Printable). Falls back to *"B"* if not given.

`linefeed`
: *`linefeed`* specifies the EOL (end-of-line) marker with which mb_encode_mimeheader() performs line-folding (a » RFC term, the act of breaking a line longer than a certain length into multiple lines. The length is currently hard-coded to 74 characters). Falls back to *"\r\n"* (CRLF) if not given.

`indent`

Return Values

A converted version of the string represented in ASCII.

ChangeLog

Version	Description
5.0.0	The *`indent`* parameter was added.

Examples

Example #15 - mb_encode_mimeheader() example

```
<?php
$name = ""; // kanji
$mbox = "kru";
$doma = "gtinn.mon";
$addr = mb_encode_mimeheader($name, "UTF-7", "Q") . " <" . $mbox . "@" .
$doma . ">";
echo $addr;
?>
```

Notes

Note

This function isn't designed to break lines at higher-level contextual break points (word boundaries, etc.). This behaviour may clutter up the original string with unexpected spaces.

See Also

- mb_decode_mimeheader()

mb_encode_numericentity

mb_encode_numericentity -- Encode character to HTML numeric string reference

Description

string **mb_encode_numericentity** (string `$str`, array `$convmap` [, string `$encoding`])

Converts specified character codes in string *`str`* from HTML numeric character reference to character code.

Parameters

`str`
: The string being encoded.

`convmap`
: *`convmap`* is array specifies code area to convert.

`encoding`
: The *`encoding`* parameter is the character encoding. If it is omitted, the internal character encoding value will be used.

Return Values

The converted string.

Examples

Example #16 - *`convmap`* **example**

```
$convmap = array (
int start_code1, int end_code1, int offset1, int mask1,
int start_code2, int end_code2, int offset2, int mask2,
........
int start_codeN, int end_codeN, int offsetN, int maskN );
// Specify Unicode value for start_codeN and end_codeN
// Add offsetN to value and take bit-wise 'AND' with maskN, then
// it converts value to numeric string reference.
```

Examples

Example #17 - mb_encode_numericentity() example

```
<?php
/* Convert Left side of ISO-8859-1 to HTML numeric character reference */
$convmap = array(0x80, 0xff, 0, 0xff);
$str = mb_encode_numericentity($str, $convmap, "ISO-8859-1");

/* Convert user defined SJIS-win code in block 95-104 to numeric
  string reference */
$convmap = array(
      0xe000, 0xe03e, 0x1040, 0xffff,
      0xe03f, 0xe0bb, 0x1041, 0xffff,
      0xe0bc, 0xe0fa, 0x1084, 0xffff,
      0xe0fb, 0xe177, 0x1085, 0xffff,
      0xe178, 0xe1b6, 0x10c8, 0xffff,
      0xe1b7, 0xe233, 0x10c9, 0xffff,
      0xe234, 0xe272, 0x110c, 0xffff,
      0xe273, 0xe2ef, 0x110d, 0xffff,
      0xe2f0, 0xe32e, 0x1150, 0xffff,
      0xe32f, 0xe3ab, 0x1151, 0xffff );
$str = mb_encode_numericentity($str, $convmap, "sjis-win");
?>
```

See Also

- mb_decode_numericentity()

mb_ereg_match

mb_ereg_match -- Regular expression match for multibyte string

Description

bool **mb_ereg_match** (string `$pattern`, string `$string` [, string `$option`])

A regular expression match for a multibyte string

Parameters

`pattern`
The regular expression pattern.

`string`
The string being evaluated.

`option`

Return Values

Returns **TRUE** if *`string`* matches the regular expression *`pattern`*, **FALSE** if not.

Notes

Note
The internal encoding or the character encoding specified by mb_regex_encoding() will be used as the character encoding for this function.

See Also

- mb_regex_encoding()
- mb_ereg()

mb_ereg_replace

mb_ereg_replace -- Replace regular expression with multibyte support

Description

string **mb_ereg_replace** (string `$pattern`, string `$replacement`, string `$string` [, string `$option`])

Scans *`string`* for matches to *`pattern`*, then replaces the matched text with *`replacement`*

Parameters

`pattern`
: The regular expression pattern. Multibyte characters may be used in *`pattern`*.

`replacement`
: The replacement text.

`string`
: The string being checked.

`option`
: Matching condition can be set by *`option`* parameter. If *i* is specified for this parameter, the case will be ignored. If *x* is specified, white space will be ignored. If *m* is specified, match will be executed in multiline mode and line break will be included in '.'. If *p* is specified, match will be executed in POSIX mode, line break will be considered as normal character. If *e* is specified, *`replacement`* string will be evaluated as PHP expression.

Return Values

The resultant string on success, or **FALSE** on error.

Notes

Note
The internal encoding or the character encoding specified by mb_regex_encoding() will be used as the character encoding for this function.

See Also

- mb_regex_encoding()
- mb_eregi_replace()

mb_ereg_search_getpos

mb_ereg_search_getpos -- Returns start point for next regular expression match

Description

int **mb_ereg_search_getpos** (void)

Returns the start point for the next regular expression match.

Parameters

This function has no parameters.

Return Values

mb_ereg_search_getpos() returns the point to start regular expression match for mb_ereg_search(), mb_ereg_search_pos(), mb_ereg_search_regs(). The position is represented by bytes from the head of string.

Notes

Note
The internal encoding or the character encoding specified by mb_regex_encoding() will be used as the character encoding for this function.

See Also

- mb_regex_encoding()
- mb_ereg_search_setpos()

mb_ereg_search_getregs

mb_ereg_search_getregs -- Retrieve the result from the last multibyte regular expression match

Description

array **mb_ereg_search_getregs** (void)

Retrieve the result from the last multibyte regular expression match

Parameters

This function has no parameters.

Return Values

An array including the sub-string of matched part by last mb_ereg_search(), mb_ereg_search_pos(), mb_ereg_search_regs(). If there are some matches, the first element will have the matched sub-string, the second element will have the first part grouped with brackets, the third element will have the second part grouped with brackets, and so on. It returns **FALSE** on error;

Notes

Note
The internal encoding or the character encoding specified by mb_regex_encoding() will be used as the character encoding for this function.

See Also

- mb_regex_encoding()
- mb_ereg_search_init()

mb_ereg_search_init

mb_ereg_search_init -- Setup string and regular expression for a multibyte regular expression match

Description

bool **mb_ereg_search_init** (string `$string` [, string `$pattern` [, string `$option`]])

mb_ereg_search_init() sets *`string`* and *`pattern`* for a multibyte regular expression. These values are used for mb_ereg_search(), mb_ereg_search_pos(), and mb_ereg_search_regs().

Parameters

This function has no parameters.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Notes

Note
The internal encoding or the character encoding specified by mb_regex_encoding() will be used as the character encoding for this function.

See Also

- mb_regex_encoding()
- mb_ereg_search_regs()

mb_ereg_search_pos

mb_ereg_search_pos -- Returns position and length of a matched part of the multibyte regular expression for a predefined multibyte string

Description

array **mb_ereg_search_pos** ([string `$pattern` [, string `$option`]])

Returns position and length of a matched part of the multibyte regular expression for a predefined multibyte string

The string for match is specified by mb_ereg_search_init(). If it is not specified, the previous one will be used.

Parameters

`pattern`
The search pattern.

`option`
The search option.

Return Values

An array including the position of a matched part for a multibyte regular expression. The first element of the array will be the beginning of matched part, the second element will be length (bytes) of matched part. It returns **FALSE** on error.

Notes

Note
The internal encoding or the character encoding specified by mb_regex_encoding() will be used as the character encoding for this function.

See Also

- mb_regex_encoding()
- mb_ereg_search_init()

mb_ereg_search_regs

mb_ereg_search_regs -- Returns the matched part of a multibyte regular expression

Description

array **mb_ereg_search_regs** ([string `$pattern` [, string `$option`]])

Returns the matched part of a multibyte regular expression.

Parameters

`pattern`
The search pattern.

`option`
The search option.

Return Values

mb_ereg_search_regs() executes the multibyte regular expression match, and if there are some matched part, it returns an array including substring of matched part as first element, the first grouped part with brackets as second element, the second grouped part as third element, and so on. It returns **FALSE** on error.

Notes

Note
The internal encoding or the character encoding specified by mb_regex_encoding() will be used as the character encoding for this function.

See Also

- mb_regex_encoding()
- mb_ereg_search_init()

mb_ereg_search_setpos

mb_ereg_search_setpos -- Set start point of next regular expression match

Description

bool **mb_ereg_search_setpos** (int `$position`)

mb_ereg_search_setpos() sets the starting point of a match for mb_ereg_search().

Parameters

`position`
The position to set.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Notes

Note
The internal encoding or the character encoding specified by mb_regex_encoding() will be used as the character encoding for this function.

See Also

- mb_regex_encoding()
- mb_ereg_search_init()

mb_ereg_search

mb_ereg_search -- Multibyte regular expression match for predefined multibyte string

Description

bool **mb_ereg_search** ([string `$pattern` [, string `$option`]])

Performs a multibyte regular expression match for a predefined multibyte string.

Parameters

`pattern`
The search pattern.

`option`
The search option.

Return Values

mb_ereg_search() returns **TRUE** if the multibyte string matches with the regular expression, or **FALSE** otherwise. The string for matching is set by mb_ereg_search_init(). If *`pattern`* is not specified, the previous one is used.

Notes

Note
The internal encoding or the character encoding specified by mb_regex_encoding() will be used as the character encoding for this function.

See Also

- mb_regex_encoding()
- mb_ereg_search_init()

mb_ereg

mb_ereg -- Regular expression match with multibyte support

Description

int **mb_ereg** (string `$pattern`, string `$string` [, array `$regs`])

Executes the regular expression match with multibyte support.

Parameters

`pattern`
The search pattern.

`string`
The search string.

`regs`
Contains a substring of the matched string.

Return Values

Executes the regular expression match with multibyte support, and returns *1* if matches are found. If the optional *`regs`* parameter was specified, the function returns the byte length of matched part, and the array *`regs`* will contain the substring of matched string. The function returns *1* if it matches with the empty string. If no matches are found or an error happens, **FALSE** will be returned.

Notes

Note
The internal encoding or the character encoding specified by mb_regex_encoding() will be used as the character encoding for this function.

See Also

- mb_regex_encoding()
- mb_eregi()

mb_eregi_replace

mb_eregi_replace -- Replace regular expression with multibyte support ignoring case

Description

string **mb_eregi_replace** (string `$pattern`, string `$replace`, string `$string` [, string `$option`])

Scans *`string`* for matches to *`pattern`*, then replaces the matched text with *`replacement`*.

Parameters

`pattern`
: The regular expression pattern. Multibyte characters may be used. The case will be ignored.

`replace`
: The replacement text.

`string`
: The searched string.

`option`
: *`option`* has the same meaning as in mb_ereg_replace().

Return Values

The resultant string or **FALSE** on error.

Notes

Note
The internal encoding or the character encoding specified by mb_regex_encoding() will be used as the character encoding for this function.

See Also

- mb_regex_encoding()
- mb_ereg_replace()

mb_eregi

mb_eregi -- Regular expression match ignoring case with multibyte support

Description

int **mb_eregi** (string `$pattern`, string `$string` [, array `$regs`])

Executes the case insensitive regular expression match with multibyte support.

Parameters

`pattern`
The regular expression pattern.

`string`
The string being searched.

`regs`
Contains a substring of the matched string.

Return Values

Executes the regular expression match with multibyte support, and returns *1* if matches are found. If the optional *`regs`* parameter was specified, the function returns the byte length of matched part, and the array *`regs`* will contain the substring of matched string. The function returns *1* if it matches with the empty string. If no matches are found or an error happens, **FALSE** will be returned.

Notes

Note
The internal encoding or the character encoding specified by mb_regex_encoding() will be used as the character encoding for this function.

See Also

- mb_regex_encoding()
- mb_ereg()

mb_get_info

mb_get_info -- Get internal settings of mbstring

Description

mixed **mb_get_info** ([string `$type`])

mb_get_info() returns the internal setting parameters of mbstring.

Parameters

`type`
: If *`type`* isn't specified or is specified to "all", an array having the elements "internal_encoding", "http_output", "http_input", "func_overload", "mail_charset", "mail_header_encoding", "mail_body_encoding" will be returned. If *`type`* is specified as "http_output", "http_input", "internal_encoding", "func_overload", the specified setting parameter will be returned.

Return Values

An array of type information if *`type`* is not specified, otherwise a specific *`type`*.

ChangeLog

Version	Description
5.1.3	The element types "mail_charset", "mail_header_encoding", and "mail_body_encoding" were made available.

See Also

- mb_regex_encoding()
- mb_http_output()

mb_http_input

mb_http_input -- Detect HTTP input character encoding

Description

mixed **mb_http_input** ([string `$type`])

Detects the HTTP input character encoding.

Parameters

`type`
: Input string specifies the input type. "G" for GET, "P" for POST, "C" for COOKIE, "S" for string, "L" for list, and "I" for the whole list (will return array). If type is omitted, it returns the last input type processed.

Return Values

The character encoding name, as per the *`type`*. If mb_http_input() does not process specified HTTP input, it returns **FALSE**.

See Also

- mb_internal_encoding()
- mb_http_output()
- mb_detect_order()

mb_http_output

mb_http_output -- Set/Get HTTP output character encoding

Description

mixed **mb_http_output** ([string `$encoding`])

Set/Get the HTTP output character encoding. Output after this function is converted to *`encoding`*.

Parameters

`encoding`
: If *`encoding`* is set, mb_http_output() sets the HTTP output character encoding to *`encoding`*. If *`encoding`* is omitted, mb_http_output() returns the current HTTP output character encoding.

Return Values

If *`encoding`* is omitted, mb_http_output() returns the current HTTP output character encoding. Otherwise, Returns **TRUE** on success or **FALSE** on failure.

See Also

- mb_internal_encoding()
- mb_http_input()
- mb_detect_order()

mb_internal_encoding

mb_internal_encoding -- Set/Get internal character encoding

Description

mixed **mb_internal_encoding** ([string `$encoding`])

Set/Get the internal character encoding

Parameters

`encoding`

`encoding` is the character encoding name used for the HTTP input character encoding conversion, HTTP output character encoding conversion, and the default character encoding for string functions defined by the mbstring module.

Return Values

If *`encoding`* is set, then Returns **TRUE** on success or **FALSE** on failure. If *`encoding`* is omitted, then the current character encoding name is returned.

Examples

Example #18 - mb_internal_encoding() example

```
<?php
/* Set internal character encoding to UTF-8 */
mb_internal_encoding("UTF-8");

/* Display current internal character encoding */
echo mb_internal_encoding();
?>
```

Notes

Note

The internal encoding or the character encoding specified by mb_regex_encoding() will be used as the character encoding for this function.

See Also

- mb_http_input()
- mb_http_output()
- mb_detect_order()

mb_language

mb_language -- Set/Get current language

Description

mixed **mb_language** ([string `$language`])

Set/Get the current language.

Parameters

`language`
: Used for encoding e-mail messages. Valid languages are "Japanese", "ja","English","en" and "uni" (UTF-8). mb_send_mail() uses this setting to encode e-mail. Language and its setting is ISO-2022-JP/Base64 for Japanese, UTF-8/Base64 for uni, ISO-8859-1/quoted printable for English.

Return Values

If *`language`* is set and *`language`* is valid, it returns **TRUE**. Otherwise, it returns **FALSE**. When *`language`* is omitted, it returns the language name as a string. If no language is set previously, it then returns **FALSE**.

See Also

- mb_send_mail()

mb_list_encodings

mb_list_encodings -- Returns an array of all supported encodings

Description

array **mb_list_encodings** (void)

Returns an array containing all supported encodings.

Parameters

This function has no parameters.

Return Values

Returns a numerically indexed array.

Errors/Exceptions

This function does not emit any errors.

Examples

Example #19 - mb_list_encodings() example

```
<?php

print_r(mb_list_encodings());

?>
```

The above example will output something similar to:

```
Array
(
   [0] => pass
   [1] => auto
   [2] => wchar
   [3] => byte2be
   [4] => byte2le
   [5] => byte4be
   [6] => byte4le
   [7] => BASE64
   [8] => UUENCODE
   [9] => HTML-ENTITIES
   [10] => Quoted-Printable
   [11] => 7bit
   [12] => 8bit
```

```
    [13] => UCS-4
    [14] => UCS-4BE
    [15] => UCS-4LE
    [16] => UCS-2
    [17] => UCS-2BE
    [18] => UCS-2LE
    [19] => UTF-32
    [20] => UTF-32BE
    [21] => UTF-32LE
    [22] => UTF-16
    [23] => UTF-16BE
    [24] => UTF-16LE
    [25] => UTF-8
    [26] => UTF-7
    [27] => UTF7-IMAP
    [28] => ASCII
    [29] => EUC-JP
    [30] => SJIS
    [31] => eucJP-win
    [32] => SJIS-win
    [33] => JIS
    [34] => ISO-2022-JP
    [35] => Windows-1252
    [36] => ISO-8859-1
    [37] => ISO-8859-2
    [38] => ISO-8859-3
    [39] => ISO-8859-4
    [40] => ISO-8859-5
    [41] => ISO-8859-6
    [42] => ISO-8859-7
    [43] => ISO-8859-8
    [44] => ISO-8859-9
    [45] => ISO-8859-10
    [46] => ISO-8859-13
    [47] => ISO-8859-14
    [48] => ISO-8859-15
    [49] => EUC-CN
    [50] => CP936
    [51] => HZ
    [52] => EUC-TW
    [53] => BIG-5
    [54] => EUC-KR
    [55] => UHC
    [56] => ISO-2022-KR
    [57] => Windows-1251
    [58] => CP866
    [59] => KOI8-R
)
```

mb_output_handler

mb_output_handler -- Callback function converts character encoding in output buffer

Description

string **mb_output_handler** (string `$contents`, int `$status`)

mb_output_handler() is ob_start() callback function. mb_output_handler() converts characters in the output buffer from internal character encoding to HTTP output character encoding.

Parameters

`contents`
The contents of the output buffer.

`status`
The status of the output buffer.

Return Values

The converted string.

ChangeLog

Version	Description
4.1.0	This handler now adds the charset HTTP header when the following conditions are met: • Does not set *Content-Type*, using header(). • The default MIME type begins with *text/*. • The mbstring.http_input setting is something other than *pass*.

Examples

Example #20 - mb_output_handler() example

```
<?php
mb_http_output("UTF-8");
ob_start("mb_output_handler");
?>
```

Notes

Note

If you want to output some binary data such as image from PHP script with PHP 4.3.0 or later, Content-Type: header must be send using header() before any binary data was send to client (e.g. header("Content-Type: image/png")). If Content-Type: header was send, output character encoding conversion will not be performed.

Note that if 'Content-Type: text/*' was send using header(), the sending data is regarded as text, encoding conversion will be performed using character encoding settings.

If you want to output some binary data such as image from PHP script with PHP 4.2.x or earlier, you must set output encoding to "pass" using mb_http_output().

See Also

- ob_start()

mb_parse_str

mb_parse_str -- Parse GET/POST/COOKIE data and set global variable

Description

bool **mb_parse_str** (string `$encoded_string` [, array `&$result`])

Parses GET/POST/COOKIE data and sets global variables. Since PHP does not provide raw POST/COOKIE data, it can only be used for GET data for now. It parses URL encoded data, detects encoding, converts coding to internal encoding and set values to the *`result`* array or global variables.

Parameters

`encoded_string`
: The URL encoded data.

`result`
: An array containing decoded and character encoded converted values.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- mb_detect_order()
- mb_internal_encoding()

mb_preferred_mime_name

mb_preferred_mime_name -- Get MIME charset string

Description

string **mb_preferred_mime_name** (string `$encoding`)

Get a MIME charset string for a specific encoding.

Parameters

`encoding`
The encoding being checked.

Return Values

The MIME *charset* string for character encoding *`encoding`*.

Examples

Example #21 - mb_preferred_mime_string() example

```
<?php
$outputenc = "sjis-win";
mb_http_output($outputenc);
ob_start("mb_output_handler");
header("Content-Type: text/html; charset=" .
mb_preferred_mime_name($outputenc));
?>
```

mb_regex_encoding

mb_regex_encoding -- Returns current encoding for multibyte regex as string

Description

mixed **mb_regex_encoding** ([string `$encoding`])

Returns the current encoding for a multibyte regex as a string.

Parameters

`encoding`
: The *`encoding`* parameter is the character encoding. If it is omitted, the internal character encoding value will be used.

Return Values

Returns the character encoding used by multibyte regex functions.

See Also

- mb_internal_encoding()
- mb_ereg()

mb_regex_set_options

mb_regex_set_options -- Set/Get the default options for mbregex functions

Description

string **mb_regex_set_options** ([string `$options`])

Sets the default options described by *`options`* for multibyte regex functions.

Parameters

`options`
: The options to set.

Return Values

The previous options. If *`options`* is omitted, it returns the string that describes the current options.

See Also

- mb_split()
- mb_ereg()
- mb_eregi()

mb_send_mail

mb_send_mail -- Send encoded mail

Description

bool **mb_send_mail** (string `$to`, string `$subject`, string `$message` [, string `$additional_headers` [, string `$additional_parameter`]])

Sends email. Headers and messages are converted and encoded according to the mb_language() setting. It's a wrapper function for mail(), so see also mail() for details.

Parameters

`to`
The mail addresses being sent to. Multiple recipients may be specified by putting a comma between each address in *`to`*. This parameter is not automatically encoded.

`subject`
The subject of the mail.

`message`
The message of the mail.

`additional_headers`
`additional_headers` is inserted at the end of the header. This is typically used to add extra headers. Multiple extra headers are separated with a newline ("\n").

`additional_parameter`
`additional_parameter` is a MTA command line parameter. It is useful when setting the correct Return-Path header when using sendmail.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

ChangeLog

Version	Description
5.0.0	The *Content-Type* and *Content-Transfer-Encoding* headers may be redefined as of PHP 5.0.0. Before this time, the values defined by mb_language() are always used.

See Also

- mail()
- **mb_encode_mimeheade()**
- mb_language()

mb_split

mb_split -- Split multibyte string using regular expression

Description

array **mb_split** (string `$pattern`, string `$string` [, int `$limit`])

Split a multibyte *`string`* using regular expression *`pattern`* and returns the result as an array.

Parameters

`pattern`
: The regular expression pattern.

`string`
: The string being split.

`limit`
: If optional parameter *`limit`* is specified, it will be split in *`limit`* elements as maximum.

Return Values

The result as an array.

Notes

Note
The internal encoding or the character encoding specified by mb_regex_encoding() will be used as the character encoding for this function.

See Also

- mb_regex_encoding()
- mb_ereg()

mb_strcut

mb_strcut -- Get part of string

Description

string **mb_strcut** (string `$str`, int `$start` [, int `$length` [, string `$encoding`]])

mb_strcut() performs equivalent operation as mb_substr() with different method. If *`start`* position is multi-byte character's second byte or larger, it starts from first byte of multi-byte character.

It subtracts string from *`str`* that is shorter than *`length`* AND character that is not part of multi-byte string or not being middle of shift sequence.

Parameters

`str`
: The string being cut.

`start`
: The position that begins the cut.

`length`
: The string being decoded.

`encoding`
: The *`encoding`* parameter is the character encoding. If it is omitted, the internal character encoding value will be used.

Return Values

mb_strcut() returns the portion of *`str`* specified by the *`start`* and *`length`* parameters.

See Also

- mb_substr()
- mb_internal_encoding()

mb_strimwidth

mb_strimwidth -- Get truncated string with specified width

Description

string **mb_strimwidth** (string `$str`, int `$start`, int `$width` [, string `$trimmarker` [, string `$encoding`]])

Truncates string *`str`* to specified *`width`*.

Parameters

`str`
: The string being decoded.

`start`
: The start position offset. Number of characters from the beginning of string. (First character is 0)

`width`
: The width of the desired trim.

`trimmarker`
: A string that is added to the end of string when string is truncated.

`encoding`
: The *`encoding`* parameter is the character encoding. If it is omitted, the internal character encoding value will be used.

Return Values

The truncated string. If *`trimmarker`* is set, *`trimmarker`* is appended to the return value.

Examples

Example #22 - mb_strimwidth() example

```
<?php
$str = mb_strimwidth($str, 0, 40, "..>");
?>
```

See Also

- mb_strwidth()
- mb_internal_encoding()

mb_stripos

mb_stripos -- Finds position of first occurrence of a string within another, case insensitive

Description

int **mb_stripos** (string `$haystack`, string `$needle` [, int `$offset` [, string `$encoding`]])

mb_stripos() returns the numeric position of the first occurrence of *`needle`* in the *`haystack`* string. Unlike mb_strpos(), mb_stripos() is case-insensitive. If *`needle`* is not found, it returns **FALSE**.

Parameters

`haystack`
: The string from which to get the position of the first occurrence of *`needle`*

`needle`
: The string to find in *`haystack`*

`offset`
: The position in *`haystack`* to start searching

`encoding`
: Character encoding name to use. If it is omitted, internal character encoding is used.

Return Values

Return the numeric position of the first occurrence of *`needle`* in the *`haystack`* string, or **FALSE** if *`needle`* is not found.

See Also

- stripos()
- strpos()
- mb_strpos()

mb_stristr

mb_stristr -- Finds first occurrence of a string within another, case insensitive

Description

string **mb_stristr** (string `$haystack`, string `$needle` [, bool `$part` [, string `$encoding`]])

mb_stristr() finds the first occurrence of *`needle`* in *`haystack`* and returns the portion of *`haystack`*. Unlike mb_strstr(), mb_stristr() is case-insensitive. If *`needle`* is not found, it returns **FALSE**.

Parameters

`haystack`
: The string from which to get the first occurrence of *`needle`*

`needle`
: The string to find in *`haystack`*

`part`
: Determines which portion of *`haystack`* this function returns. If set to **TRUE**, it returns all of *`haystack`* from the beginning to the first occurrence of *`needle`*. If set to **FALSE**, it returns all of *`haystack`* from the first occurrence of *`needle`* to the end, Default value is **FALSE**.

`encoding`
: Character encoding name to use. If it is omitted, internal character encoding is used.

Return Values

Returns the portion of *`haystack`*, or **FALSE** if *`needle`* is not found.

See Also

- stristr()
- strstr()
- mb_strstr()

mb_strlen

mb_strlen -- Get string length

Description

int **mb_strlen** (string `$str` [, string `$encoding`])

Gets the length of a string.

Parameters

`str`
: The string being checked for length.

`encoding`
: The *`encoding`* parameter is the character encoding. If it is omitted, the internal character encoding value will be used.

Return Values

Returns the number of characters in string *`str`* having character encoding *`encoding`*. A multi-byte character is counted as 1.

See Also

- mb_internal_encoding()
- strlen()

mb_strpos

mb_strpos -- Find position of first occurrence of string in a string

Description

int **mb_strpos** (string `$haystack`, string `$needle` [, int `$offset` [, string `$encoding`]])

Finds position of the first occurrence of a string in a string.

Performs a multi-byte safe strpos() operation based on number of characters. The first character's position is 0, the second character position is 1, and so on.

Parameters

`haystack`
The string being checked.

`needle`
The position counted from the beginning of *`haystack`*.

`offset`
The search offset. If it is not specified, 0 is used.

`encoding`
The *`encoding`* parameter is the character encoding. If it is omitted, the internal character encoding value will be used.

Return Values

Returns the numeric position of the first occurrence of *`needle`* in the *`haystack`* string. If *`needle`* is not found, it returns **FALSE**.

See Also

- **mb_()**
- mb_internal_encoding()
- strpos()

mb_strrchr

mb_strrchr -- Finds the last occurrence of a character in a string within another

Description

string **mb_strrchr** (string `$haystack`, string `$needle` [, bool `$part` [, string `$encoding`]])

mb_strrchr() finds the last occurrence of *`needle`* in *`haystack`* and returns the portion of *`haystack`*. If *`needle`* is not found, it returns **FALSE**.

Parameters

`haystack`
: The string from which to get the last occurrence of *`needle`*

`needle`
: The string to find in *`haystack`*

`part`
: Determines which portion of *`haystack`* this function returns. If set to **TRUE**, it returns all of *`haystack`* from the beginning to the last occurrence of *`needle`*. If set to **FALSE**, it returns all of *`haystack`* from the last occurrence of *`needle`* to the end, Default value is **FALSE**.

`encoding`
: Character encoding name to use. If it is omitted, internal character encoding is used.

Return Values

Returns the portion of *`haystack`*. or **FALSE** if *`needle`* is not found.

See Also

- strrchr()
- mb_strstr()
- mb_strrichr()

mb_strrichr

mb_strrichr -- Finds the last occurrence of a character in a string within another, case insensitive

Description

string **mb_strrichr** (string `$haystack`, string `$needle` [, bool `$part` [, string `$encoding`]])

mb_strrichr() finds the last occurrence of *`needle`* in *`haystack`* and returns the portion of *`haystack`*. Unlike mb_strrchr(), mb_strrichr() is case-insensitive. If *`needle`* is not found, it returns **FALSE**.

Parameters

`haystack`
: The string from which to get the last occurrence of *`needle`*

`needle`
: The string to find in *`haystack`*

`part`
: Determines which portion of *`haystack`* this function returns. If set to **TRUE**, it returns all of *`haystack`* from the beginning to the last occurrence of *`needle`*. If set to **FALSE**, it returns all of *`haystack`* from the last occurrence of *`needle`* to the end, Default value is **FALSE**.

`encoding`
: Character encoding name to use. If it is omitted, internal character encoding is used.

Return Values

Returns the portion of *`haystack`*. or **FALSE** if *`needle`* is not found.

See Also

- mb_stristr()
- mb_strrchr()

mb_strripos

mb_strripos -- Finds position of last occurrence of a string within another, case insensitive

Description

int **mb_strripos** (string `$haystack`, string `$needle` [, int `$offset` [, string `$encoding`]])

mb_strripos() performs multi-byte safe strripos() operation based on number of characters. *`needle`* position is counted from the beginning of *`haystack`*. First character's position is 0. Second character position is 1. Unlike mb_strrpos(), mb_strripos() is case-insensitive.

Parameters

`haystack`
: The string from which to get the position of the last occurrence of *`needle`*

`needle`
: The string to find in *`haystack`*

`offset`
: The position in *`haystack`* to start searching

`encoding`
: Character encoding name to use. If it is omitted, internal character encoding is used.

Return Values

Return the numeric position of the last occurrence of *`needle`* in the *`haystack`* string, or **FALSE** if *`needle`* is not found.

See Also

- strripos()
- strrpos()
- mb_strrpos()

mb_strrpos

mb_strrpos -- Find position of last occurrence of a string in a string

Description

int **mb_strrpos** (string `$haystack`, string `$needle` [, int `$offset` [, string `$encoding`]])

Performs a multibyte safe strrpos() operation based on the number of characters. *`needle`* position is counted from the beginning of *`haystack`*. First character's position is 0. Second character position is 1.

Parameters

`haystack`
The string being checked, for the last occurrence of *`needle`*

`needle`
The string to find in *`haystack`*.

`offset`
May be specified to begin searching an arbitrary number of characters into the string. Negative values will stop searching at an arbitrary point prior to the end of the string.

`encoding`
The *`encoding`* parameter is the character encoding. If it is omitted, the internal character encoding value will be used.

Return Values

Returns the numeric position of the last occurrence of *`needle`* in the *`haystack`* string. If *`needle`* is not found, it returns **FALSE**.

ChangeLog

Version	Description
5.2.0	Added the optional parameter *`offset`*.

Notes

Note
The *encoding* parameter was moved from the third position to the fourth in PHP 5.2.0. For backward compatibility, *encoding* can be specified as the third parameter, but doing so is deprecated and will be removed in the future.

Note
The internal encoding or the character encoding specified by mb_regex_encoding() will be used as the character encoding for this function.

See Also

- mb_strpos()
- mb_internal_encoding()
- strrpos()

mb_strstr

mb_strstr -- Finds first occurrence of a string within another

Description

string **mb_strstr** (string `$haystack`, string `$needle` [, bool `$part` [, string `$encoding`]])

mb_strstr() finds the first occurrence of *`needle`* in *`haystack`* and returns the portion of *`haystack`*. If *`needle`* is not found, it returns **FALSE**.

Parameters

`haystack`
: The string from which to get the first occurrence of *`needle`*

`needle`
: The string to find in *`haystack`*

`part`
: Determines which portion of *`haystack`* this function returns. If set to **TRUE**, it returns all of *`haystack`* from the beginning to the first occurrence of *`needle`*. If set to **FALSE**, it returns all of *`haystack`* from the first occurrence of *`needle`* to the end, Default value is **FALSE**.

`encoding`
: Character encoding name to use. If it is omitted, internal character encoding is used.

Return Values

Returns the portion of *`haystack`*, or **FALSE** if *`needle`* is not found.

See Also

- stristr()
- strstr()
- mb_stristr()

mb_strtolower

mb_strtolower -- Make a string lowercase

Description

string **mb_strtolower** (string `$str` [, string `$encoding`])

Returns *`str`* with all alphabetic characters converted to lowercase.

Parameters

`str`
The string being lowercased.

`encoding`
The *`encoding`* parameter is the character encoding. If it is omitted, the internal character encoding value will be used.

Return Values

`str` with all alphabetic characters converted to lowercase.

Unicode

For more information about the Unicode properties, please see
» http://www.unicode.org/unicode/reports/tr21/.

By contrast to strtolower(), 'alphabetic' is determined by the Unicode character properties. Thus the behaviour of this function is not affected by locale settings and it can convert any characters that have 'alphabetic' property, such as A-umlaut (Ä).

Examples

Example #23 - mb_strtolower() example

```
<?php
$str = "Mary Had A Little Lamb and She LOVED It So";
$str = mb_strtolower($str);
echo $str; // Prints mary had a little lamb and she loved it so
?>
```

See Also

- mb_strtoupper()
- mb_convert_case()
- strtolower()

mb_strtoupper

mb_strtoupper -- Make a string uppercase

Description

string **mb_strtoupper** (string `$str` [, string `$encoding`])

Returns *`str`* with all alphabetic characters converted to uppercase.

Parameters

`str`
The string being uppercased.

`encoding`
The *`encoding`* parameter is the character encoding. If it is omitted, the internal character encoding value will be used.

Return Values

`str` with all alphabetic characters converted to uppercase.

Unicode

For more information about the Unicode properties, please see
» http://www.unicode.org/unicode/reports/tr21/.

By contrast to strtoupper(), 'alphabetic' is determined by the Unicode character properties. Thus the behaviour of this function is not affected by locale settings and it can convert any characters that have 'alphabetic' property, such as a-umlaut (ä).

Examples

Example #24 - mb_strtoupper() example

```
<?php
$str = "Mary Had A Little Lamb and She LOVED It So";
$str = mb_strtoupper($str);
echo $str; // Prints MARY HAD A LITTLE LAMB AND SHE LOVED IT SO
?>
```

See Also

- mb_strtolower()
- mb_convert_case()
- strtoupper()

mb_strwidth

mb_strwidth -- Return width of string

Description

int **mb_strwidth** (string `$str` [, string `$encoding`])

Returns the width of string *`str`*.

Multi-byte characters are usually twice the width of single byte characters.

Characters width

Chars	Width
U+0000 - U+0019	0
U+0020 - U+1FFF	1
U+2000 - U+FF60	2
U+FF61 - U+FF9F	1
U+FFA0 -	2

Parameters

`str`
: The string being decoded.

`encoding`
: The *`encoding`* parameter is the character encoding. If it is omitted, the internal character encoding value will be used.

Return Values

The width of string *`str`*.

See Also

- mb_strimwidth()
- mb_internal_encoding()

mb_substitute_character

mb_substitute_character -- Set/Get substitution character

Description

mixed **mb_substitute_character** ([mixed `$substrchar`])

Specifies a substitution character when input character encoding is invalid or character code does not exist in output character encoding. Invalid characters may be substituted **NULL** (no output), string or integer value (Unicode character code value).

This setting affects mb_convert_encoding(), mb_convert_variables(), mb_output_handler() , and mb_send_mail().

Parameters

`substrchar`
Specify the Unicode value as an integer, or as one of the following string s:

- "none" : no output
- "long" : Output character code value (Example: U+3000,JIS+7E7E)

Return Values

If *`substchar`* is set, it returns **TRUE** for success, otherwise returns **FALSE**. If *`substchar`* is not set, it returns the Unicode value, or " *none* " or " *long* ".

Examples

Example #25 - mb_substitute_character() example

```
<?php
/* Set with Unicode U+3013 (GETA MARK) */
mb_substitute_character(0x3013);

/* Set hex format */
mb_substitute_character("long");

/* Display current setting */
echo mb_substitute_character();
?>
```

mb_substr_count

mb_substr_count -- Count the number of substring occurrences

Description

int **mb_substr_count** (string `$haystack`, string `$needle` [, string `$encoding`])

Counts the number of times the *`needle`* substring occurs in the *`haystack`* string.

Parameters

`haystack`
The string being checked.

`needle`
The string being found.

`encoding`
The *`encoding`* parameter is the character encoding. If it is omitted, the internal character encoding value will be used.

Return Values

The number of times the *`needle`* substring occurs in the *`haystack`* string.

Examples

Example #26 - mb_substr_count() example

```
<?php
echo mb_substr_count("This is a test", "is"); // prints out 2
?>
```

See Also

- mb_strpos()
- mb_substr()
- substr_count()

mb_substr

mb_substr -- Get part of string

Description

string **mb_substr** (string `$str`, int `$start` [, int `$length` [, string `$encoding`]])

Performs a multi-byte safe substr() operation based on number of characters. Position is counted from the beginning of *`str`*. First character's position is 0. Second character position is 1, and so on.

Parameters

`str`
: The string being checked.

`start`
: The first position used in *`str`*.

`length`
: The maximum length of the returned string.

`encoding`
: The *`encoding`* parameter is the character encoding. If it is omitted, the internal character encoding value will be used.

Return Values

mb_substr() returns the portion of *`str`* specified by the *`start`* and *`length`* parameters.

See Also

- mb_strcut()
- mb_internal_encoding()