Function Handling »
« filter_var_array

- PHP Manual
- Function Reference
- Variable and Type Related Extensions
- Filter
- Filter Functions

Change language: English

Submit a Pull Request Report a Bug

filter_var

(PHP 5 >= 5.2.0, PHP 7, PHP 8)

filter_var — Filters a variable with a specified filter

Description ¶

filter_var(mixed `$value`, int `$filter` = **`FILTER_DEFAULT`**, array|int `$options` = 0): mixed

Parameters ¶

`value`

Value to filter. Note that scalar values are converted to string internally before they are filtered.

`filter`

The ID of the filter to apply. The Types of filters manual page lists the available filters.

If omitted, **`FILTER_DEFAULT`** will be used, which is equivalent to **`FILTER_UNSAFE_RAW`**. This will result in no filtering taking place by default.

`options`

Associative array of options or bitwise disjunction of flags. If filter accepts options, flags can be provided in "flags" field of array. For the "callback" filter, callable type should be passed. The callback must accept one argument, the value to be filtered, and return the value after filtering/sanitizing it.

```
<?php
// for filters that accept options, use this format
$options = array(
'options' => array(
'default' => 3, // value to return if the filter fails
// other options here
'min_range' => 0
),
'flags' => FILTER_FLAG_ALLOW_OCTAL,
);
$var = filter_var('0755', FILTER_VALIDATE_INT, $options);

// for filters that only accept flags, you can pass them directly
$var = filter_var('oops', FILTER_VALIDATE_BOOLEAN, FILTER_NULL_ON_FAILURE);
```

```
// for filters that only accept flags, you can also pass as an array
$var = filter_var('oops', FILTER_VALIDATE_BOOLEAN,
array('flags' => FILTER_NULL_ON_FAILURE));

// callback validate filter
function foo($value)
{
// Expected format: Surname, GivenNames
if (strpos($value, ", ") === false) return false;
list($surname, $givennames) = explode(", ", $value, 2);
$empty = (empty($surname) || empty($givennames));
$notstrings = (!is_string($surname) || !is_string($givennames));
if ($empty || $notstrings) {
return false;
} else {
return $value;
}
}
$var = filter_var('Doe, Jane Sue', FILTER_CALLBACK, array('options' => 'foo'));
?>
```

Return Values ¶

Returns the filtered data, or **`false`** if the filter fails.

Examples ¶

Example #1 A filter_var() example

```
<?php
var_dump(filter_var('bob@example.com', FILTER_VALIDATE_EMAIL));
var_dump(filter_var('http://example.com', FILTER_VALIDATE_URL, FILTER_FLAG_PATH_REQUIRED));
?>
```

The above example will output:

```
string(15) "bob@example.com"
bool(false)
```

See Also ¶

- filter_var_array() - Gets multiple variables and optionally filters them
- filter_input() - Gets a specific external variable by name and optionally filters it
- filter_input_array() - Gets external variables and optionally filters them
- Types of filters

+ add a note

User Contributed Notes 33 notes

up
down
173
cabrinosimone at gmail dot com ¶
9 years ago

```
Pay attention that the function will not validate "not latin" domains.

if (filter_var('уникум@из.рф', FILTER_VALIDATE_EMAIL)) {
    echo 'VALID';
} else {
    echo 'NOT VALID';
}
```

up
down
96
gt at kani dot hu ¶
9 years ago

```
I found some addresses that FILTER_VALIDATE_EMAIL rejects, but RFC5321 permits:
<?php
foreach (array(
        'localpart.ending.with.dot.@example.com',
        '(comment)localpart@example.com',
        '"this is v@lid!"@example.com',
        '"much.more unusual"@example.com',
        'postbox@com',
        'admin@mailserver1',
        '"()<>[]:,;@\\"\\\\!#$%&\'*+-/=?^_`{}| ~.a"@example.org',
        '" "@example.org',
    ) as $address) {
    echo "<p>$address is <b>".(filter_var($address, FILTER_VALIDATE_EMAIL) ? '' : 'not')."
valid</b></p>";
}
?>
Results:

localpart.ending.with.dot.@example.com is not valid
(comment)localpart@example.com is not valid
"this is v@lid!"@example.com is not valid
"much.more unusual"@example.com is not valid
postbox@com is not valid
admin@mailserver1 is not valid
"()<>[]:,;@\"\\!#$%&'*+-/=?^_`{}| ~.a"@example.org is not valid
" "@example.org is not valid

The documentation does not saying that FILTER_VALIDATE_EMAIL should pass the RFC5321, however you
can meet with these examples (especially with the first one). So this is a note, not a bug report.
```

up
down
52
divinity76 at gmail dot com ¶
5 years ago

```
note that FILTER_VALIDATE_BOOLEAN tries to be smart, recognizing words like Yes, No, Off, On, both
string and native types of true and false, and is not case-sensitive when validating strings.

<?php
$vals=array('on','On','ON','off','Off','OFF','yes','Yes','YES',
'no','No','NO',0,1,'0','1','true',
'True','TRUE','false','False','FALSE',true,false,'foo','bar');
foreach($vals as $val){
    echo var_export($val,true).': ';
var_dump(filter_var($val,FILTER_VALIDATE_BOOLEAN,FILTER_NULL_ON_FAILURE));
}
?>

outputs:
'on': bool(true)
```

```
'On': bool(true)
'ON': bool(true)
'off': bool(false)
'Off': bool(false)
'OFF': bool(false)
'yes': bool(true)
'Yes': bool(true)
'YES': bool(true)
'no': bool(false)
'No': bool(false)
'NO': bool(false)
0: bool(false)
1: bool(true)
'0': bool(false)
'1': bool(true)
'true': bool(true)
'True': bool(true)
'TRUE': bool(true)
'false': bool(false)
'False': bool(false)
'FALSE': bool(false)
true: bool(true)
false: bool(false)
'foo': NULL
'bar': NULL
```

up
down
3
rschuetz at hmcw dot com ¶
1 year ago

```
I cannot confirm what yactouat said. As of PHP 7.3, 0 will not be filtered out with
FILTER_VALIDATE_INT. It correctly returns 0, not false. Of course you have to check the return
value with an identity operator. Otherwise you cannot distinguish between 0 and false.
```

up
down
57
Andi, info at pragmamx dot org ¶
10 years ago

```
And this is also a valid url

http://example.com/"><script>alert(document.cookie)</script>
```

up
down
16
Steve ¶
4 years ago

```
The note from "hek" about HTML5 having patterns thus alleviating the need to filter in PHP is
completely wrong:  You still must filter input on the server side.  The HTML5 form inputs are
client-side, meaning they are completely under the user's control.  Only when you receive the data
in PHP is it server-side and under your control.  Once the data is under your control, then you
must filter/sanitize it properly.

This is true regardless of server-side language.  I would encourage the moderators to remove the
note from "hek" because it will mislead people with horrible consequences.

Steve
```

up
down

21
Anonymous ¶
7 years ago
```
FILTER_VALIDATE_URL allows:

filter_var('javascript://comment%0Aalert(1)', FILTER_VALIDATE_URL);

Where the %0A (URL encoded newline), in certain contexts, will split the comment from the JS code.

This can result in an XSS vulnerability.
```
up
down
2
Robert Vlach ¶
2 years ago
```
I won't recommend using this function to validate email addresses on a normal website. The problem
is that in accordance with RFC 3696 (Application Techniques for Checking and Transformation of
Names) the following email addresses would be considered as valid:

customer/department=shipping@example.com
$A12345@example.com
!def!xyz%abc@example.com
_somename@example.com
"Abc@def"@example.com

Hardly something I would accept in a live web app in 2020 :-/
```
up
down
5
mpyw628 at gmail dot com ¶
4 years ago
```
I wrote a JavaScript email validator fully compatible with PHP's filter_var() implementation.

mpyw/FILTER_VALIDATE_EMAIL.js: Email validation compatible with PHP's filter_var($value,
FILTER_VALIDATE_EMAIL)
https://github.com/mpyw/FILTER_VALIDATE_EMAIL.js
```
up
down
8
marcus at synchromedia dot co dot uk ¶
10 years ago
```
It's very likely that you actually want to detect all reserved ranges, not just private IPs, and
there's another constant for them that should be bitwise-OR'd with it.
<?php
function is_private_ip($ip) {
    return !filter_var($ip, FILTER_VALIDATE_IP, FILTER_FLAG_NO_PRIV_RANGE |
FILTER_FLAG_NO_RES_RANGE);
}
?>
```
up
down
6
jon dot bertsch at ucop dot edu ¶
13 years ago
```
Here's an actual example of the filter syntax with a flag since there doesn't appear to be a one
liner for this anywhere:
```

```
'hours' => array('filter'=>FILTER_SANITIZE_NUMBER_FLOAT, 'flags' => FILTER_FLAG_ALLOW_FRACTION,
'options'=> '.')
```

up
down
2
yactouat at hotmail dot com ¶
3 years ago

```
While getting familiar with filter_var( $var, FILTER_VALIDATE_INT ), I found interesting that 0
will be filtered out and therefore wont be considered as an int. Hope that helps someone not to be
stuck ;)

N.B.: if you need to accept 0's, you could use is_int()
```

up
down
5
dale dot liszka at gmail dot com ¶
14 years ago

```
Using the FILTER_CALLBACK requires an array to be passed as the options:

<?php
function toDash($x){
   return str_replace("_","-",$x);
}

echo filter_var("asdf_123",FILTER_CALLBACK,array("options"=>"toDash"));
// returns 'asdf-123'
?>
```

up
down
9
keevitaja at gmail dot com ¶
11 years ago

```
please note FILTER_VALIDATE_URL passes following url

http://example.ee/sdsf"f
```

up
down
7
dale dot liszka at gmail dot com ¶
14 years ago

```
Here is how to use multiple flags (for those who learn better by example, like me):

<?php
echo "|asdf".chr(9).chr(128)."_123|";
echo "\n";
// "bitwise conjunction" means logic OR / bitwise |
echo filter_var("|asdf".chr(9).chr(128)."_123\n|" ,FILTER_SANITIZE_STRING, FILTER_FLAG_STRIP_LOW |
FILTER_FLAG_STRIP_HIGH);

/*
Results:
|asdf    �_123|
|asdf_123|
*/
?>
```

up
down
0

Anonymous ¶

8 days ago

```
Pay attention:
questionmark in url is also valid

<?php
echo filter_var("http://test???test.com", FILTER_VALIDATE_URL)?"valid":"not valid"; #valid
?>
```

up
down
0

mmerlone at gmail dot com ¶

1 year ago

```
Be aware that FILTER_FLAG_PATH_REQUIRED is happy with a single slash (/), so:

<?php
$options = array('flags' => FILTER_FLAG_PATH_REQUIRED);
filter_var('http://example.com', FILTER_VALIDATE_URL, $options); // returns false
filter_var('http://example.com/', FILTER_VALIDATE_URL, $options); // returns 'http://example.com/'
?>
```

up
down
0

joe at bloe dot com ¶

8 years ago

```
"(comment)localpart@example.com"
is an invalid E-Mail address per RFC5322 (Appendix A.6.3):
"Also, the comments and white space throughout addresses, dates, and message identifiers are all
part of the obsolete syntax."
```

up
down
0

php at maisqi dot com ¶

11 years ago

```
FILTER_VALIDATE_URL does not support internationalized domain name (IDN). Valid or not, no domain
name with Unicode chars on it will pass validation.

We can circumvent this with a home grown solutions, but C code is C code, so I've gone for the
code bellow, which builds on filter_var().

<?php
$res = filter_var ($uri, FILTER_VALIDATE_URL);
if ($res) return $res;
// Check if it has unicode chars.
$l = mb_strlen ($uri);
if ($l !== strlen ($uri)) {
    // Replace wide chars by “X”.
    $s = str_repeat (' ', $l);
    for ($i = 0; $i < $l; ++$i) {
        $ch = mb_substr ($uri, $i, 1);
        $s [$i] = strlen ($ch) > 1 ? 'X' : $ch;
    }
    // Re-check now.
    $res = filter_var ($s, FILTER_VALIDATE_URL);
    if ($res) {    $uri = $res; return 1;    }
}
?>
```

```
The logic is simple. A non-ascii char is more than one byte long. We replace every one of those
chars by "X" and check again.

An alternative will be to punycode the URI before calling filter_var(), but PHP lacks native
support for punycode. I think my approach is effective. Please e-mail me if you think otherwise or
see room for improvement.
```

up
down
-1
dan at spiral8 dot net ¶
7 years ago

```
Here's a simple test using filter_var with FILTER_VALIDATE_URL.
(If you're using file_get_contents after this you will run into a problem, I was using: PHP 5.5.12
(cli))

<?php
$url = 'a://google.com';

$result = filter_var($url, FILTER_VALIDATE_URL);

if($result){
    echo 'Valid URL'.PHP_EOL;
}

var_dump($result);
?>

The result is:
Valid URL
string(14) "a://google.com"
```

up
down
-2
yoanlin93 at gmail dot com ¶
7 years ago

```
Some boolean conversions:

<?php
var_dump(filter_var('oops', FILTER_VALIDATE_BOOLEAN, array('flags' => FILTER_NULL_ON_FAILURE)));
// NULL

var_dump(filter_var('false', FILTER_VALIDATE_BOOLEAN, array('flags' => FILTER_NULL_ON_FAILURE)));
// bool(false)

var_dump(filter_var('true', FILTER_VALIDATE_BOOLEAN, array('flags' => FILTER_NULL_ON_FAILURE)));
// bool(true)

var_dump(filter_var(0, FILTER_VALIDATE_BOOLEAN, array('flags' => FILTER_NULL_ON_FAILURE)));
// bool(false)

var_dump(filter_var(1, FILTER_VALIDATE_BOOLEAN, array('flags' => FILTER_NULL_ON_FAILURE)));
// bool(true)

var_dump(filter_var('TRUE', FILTER_VALIDATE_BOOLEAN, array('flags' => FILTER_NULL_ON_FAILURE)));
// bool(true)

var_dump(filter_var('', FILTER_VALIDATE_BOOLEAN, array('flags' => FILTER_NULL_ON_FAILURE)));
// bool(false)
```

```
var_dump(filter_var('FALSE', FILTER_VALIDATE_BOOLEAN, array('flags' => FILTER_NULL_ON_FAILURE)));
// bool(false)
```

up
down
-1
crisp at tweakers dot net ¶
5 years ago

```
Note that only using FILTER_VALIDATE_URL to validate url's input may result in XSS:

$url = 'javascript://%0Aalert(document.cookie)';

if (filter_var($url, FILTER_VALIDATE_URL, FILTER_FLAG_SCHEME_REQUIRED)) {
    echo '<a href="' . $url . '">click</a>';
}

You should at least additionally check the actually used scheme.
```

up
down
-4
alex4home at gmail dot com ¶
10 years ago

```
Keep in mind that FILTER_VALIDATE_EMAIL will validate the email address according to standards.
However, giving the fact that organizations are free to restrict the forms of their own email
addresses, using ONLY this filter can you a lot of bounces.

gmail, yahoo, hotmail, aol have special rules

For example :
<?php

$email_a = '0hot\'mail_check@hotmail.com';
if (filter_var($email_a, FILTER_VALIDATE_EMAIL)) {
    echo "This (email_a) email address is considered valid.";
   //reported as valid
}

//there can be no  "0hotmail_check@hotmail.com"
//because hotmail will say "Your email address needs to start with a letter. Please try again."
even if you remove the '
?>
```

up
down
-1
Ant ¶
7 years ago

```
Note: filter_var with filter=FILTER_VALIDATE_URL used parse_url function
```

up
down
-3
dyer85 at gmail dot com ¶
14 years ago

```
Note that when using FILTER_VALIDATE_INT along with the FILTER_FLAG_ALLOW_HEX flag, the string
"2f", for example, is not validated successfully, because you must use the "0x" prefix, otherwise,
it treats the data as base 10.

The range options are also smart enough to recognize when the boundaries are exceeded in different
bases.
```

```
Here's an example:

<?php

$foo = '256';
$bar = '0x100';
var_dump(validate_int($foo)); // false, too large
var_dump(validate_int($bar)); // false, too large

function validate_int($input)
{
  return filter_var(
    $input,
    FILTER_VALIDATE_INT,

    // We must pass an associative array
    // to include the range check options.
    array(
      'flags'   => FILTER_FLAG_ALLOW_HEX,
      'options' => array('min_range' => 1, 'max_range' => 0xff)
    )
  );
}

?>
```

up
down
-4
drew_mirage at hotmail dot com ¶
9 years ago

```
One key thing to remember about filtering integers is that the value for the option max_range must
be less than or equal to the value of PHP_INT_MAX.

filter_var($someVariable, FILTER_VALIDATE_INT, array('options' => array('min_range' => 1,
'max_range' => SOME_VALUE_GREATER_THAN_PHP_INT_MAX)));

This will fail even if $someVariable is a valid integer in the expected range.

This can show up when you are attempting to validate a potential key for an unsigned MySQL INT
type (whose maximum value is 4294967295) on a 32-bit system, where the value of PHP_INT_MAX is
2147483647.
```

up
down
-4
buttflattery at gmail dot com ¶
7 years ago

```
FILTER_VALIDATE_URL validates a url like http://www.
```

up
down
-7
Martin L ¶
10 years ago

```
FILTER_SANITIZE_EMAIL header injection test.

<?php
$InjString = "\r\n|\n|%0A|%0D|bcc:|to:|cc:|Content-Type:|Mime-Type:|";
echo filter_var($InjString, FILTER_SANITIZE_EMAIL);
```

```
?>

||%0A|%0D|bcc|to|cc|Content-Type|Mime-Type|
```

up
down
-6
alice at deviant dot email ¶
6 years ago

```
Many people, myself included, have found that the FILTER_VALIDATE_EMAIL does not actually properly
work.

Below is a wrapper that I believe validates every legal routable address.

<?php

/*******************************************
*
* These are the function
*
*  check_username is called by check_email
*  - it compensates for bugs in the php
*    filter_var function.
*  - returns boolean
*
*  check_email is the function to use.
*  First argument is string, address to
*    check
*  Second argument is optional boolean,
*    whether or not to use DNS to validate
*    the domain name. Defaults to true
*  Returns boolean
*
*/

function check_username($uname) {
  //Only UTF-8 addresses are legal
  if (iconv('UTF-8', 'UTF-8', $input) != $input) {
      return FALSE;
  }
  //replace all characters above U+007F with letter U for simplicity of checking
  $uname = preg_replace('/[\x{007F}-\x{FFFF}]/u', 'U', $uname);

  //remove comments - only legal in format (comment) at beginning or end of username
  $s[] = '/^\([^\)]*\)/'; $s[] = '/\([^\)]*\)$/';
  $uname = preg_replace($s, '', $uname);
  //make sure we have something left
  if(strlen(trim($uname)) == 0) {
    return FALSE;
  }
  // check for legal dot usage
  if(substr_count($uname, '..') > 0) {
    return FALSE;
  }
  // convert \\ and \" to an A for simplicity
  $s[] = '/[\\\][\\\]/';
  $s[] = '/\\\"/';
  $uname = preg_replace($s, 'A', $uname);
  // check for illegal use of quotes
  if(preg_match('/[^.]+"[^.]+/', $uname)) {
    return FALSE;
  }
  // compensate for characters legal when in quotes
  $uname = preg_replace_callback('/"(.*)"/', function ($m) {
    $s[]="/[ \(\),\:;<>@\[\] ]/";
    return preg_replace($s,'Q',$m[1]);
    }, $uname);
  // check what we have left with filter_var
  return filter_var($uname . '@example.org', FILTER_VALIDATE_EMAIL);
}

function check_email($email, $dns_check=true) {
  $array = explode('@', $email);
  if(count($array) < 2) {
    return FALSE;
  }
  $domain = end($array);
  array_pop($array);
  if(function_exists('idn_to_ascii')) {
    //php filter no workie with unicode characters
    $domain = idn_to_ascii($domain);
  }
  $ipcheck = preg_replace(array('/^\[ipv6\:/i', '/^\[/', '/\]$/'), '', $domain);
  if(filter_var($ipcheck, FILTER_VALIDATE_IP)) {
    // it's an IP address
    if(! filter_var($ipcheck, FILTER_VALIDATE_IP, FILTER_FLAG_NO_PRIV_RANGE |
FILTER_FLAG_NO_RES_RANGE)) {
      return FALSE;
    }
  } else {
    // it's a domain name
    //   php bug - FILTER_VALIDATE_EMAIL doesn't like naked TLD
    if(! filter_var('user@a.' . $domain, FILTER_VALIDATE_EMAIL)) {
      return FALSE;
    }
    if($dns_check) {
      if(! dns_get_record($domain)) {
        return FALSE;
      }
    }
  }
  //now check legal username
  return check_username(implode('@', $array));
}
?>
```

It evaluates the address in two parts, first evaluating the host and if that legal it then evaluates the user name.

If there is a DNS problem *and* the default $dns_check value of true is used, valid will fail. If it is an international domain name, you have to have the php-intl package installed.

Enjoy.

up
down
-5
Tom ¶
9 years ago

```
It is important to note that though the data type of the first parameter of the function is stated
as "mixed", this is only one half of the truth.

While it accepts any data type, the first parameter will always be cast to string before being
validated or sanitized.

It seems that this function was designed strictly to be used on user input strings. For example:
from an online-form. When using it for anything other than that, you may see issues. So read the
documentation very carefully!

Especially note that there is an (to date) unresolved issue (#49510) concerning the Boolean filter
while using the FILTER_NULL_ON_FAILURE flag. Note that both (string) FALSE and FALSE are not
recognized as boolean values and will return NULL (not FALSE as you might expect).

I thus personally suggest that (to date) the best way to take the filter_var()-functions beyond
their original purpose (and allow future extension and customization) is to wrap them in your own
classes. This will allow you to work-around unexpected behavior on non-string input and add your
custom checks, or back-port filters or sanitizers that may be added in later versions of PHP.
(Especially since PHP currently still lacks filters and sanitizers for some of the more exotic
HTML5 input types, like "color". Thus there actually is a chance that we may see a need for custom
filters or backports at some point in the future.)
```

up
down
-3
Anonymous ¶
4 years ago

```
Replying to Andi:

This is NOT a valid URL, as the characters are not encoded

http://example.com/"><script>alert(document.cookie)</script>

This is a valid URL:

http://example.com/%22%3E%3Cscript%3Ealert%28document.cookie%29%3C%2Fscript%3E
```

up
down
-12
joelhy ¶
11 years ago

```
For those looking for private ip checking, there it is:
<?php
function is_private_ip($ip)
{
     return !filter_var($ip, FILTER_VALIDATE_IP, FILTER_FLAG_NO_PRIV_RANGE);
}
?>
```

up
down
-14
drtebi at yahoo ¶
13 years ago

```
Notice that filter_var with FILTER_VALIDATE_EMAIL does not work if you are trying to get a String
from an XML document e.g. via xpath.

I often use XML files as configuration files and use a function that returns a string from the
config file via xpath. While this worked fine before 5.2.11, it doesn't anymore (and shouldn't,
since it's an XML Element, not a String).
```

```
To overcome this problem, $variable can be type-casted:

<?php
$variable = fancyXmlGetFunction('from');
filter_var((String) $variable, FILTER_VALIDATE_EMAIL);
?>
```

+ add a note

- Filter Functions
    - filter_has_var
    - filter_id
    - filter_input_array
    - filter_input
    - filter_list
    - filter_var_array
    - filter_var

- Copyright © 2001-2023 The PHP Group
- My PHP.net
- Contact
- Other PHP.net sites
- Privacy policy