Radius

Introduction

This package is based on the libradius of FreeBSD. This PECL adds full support for Radius Authentication (» RFC 2865) and Radius Accounting (» RFC 2866). This package is available for Unix (tested on FreeBSD and Linux) and for Windows.

Note
An exact description for libradius can be found » here. A detailed description of the configuration file can be found » here.

Installing/Configuring

Requirements

No external libraries are needed to build this extension.

Installation

Howto install the package?

- untar the package (usually into php4/ext)
- rename radius-x.x to radius
- run ./buildconf in php4
- run ./configure --enable-radius
- make; make install

or if you would like to have it as .so:

- untar the package
- run phpize in the radius-x.x directory
- run ./configure in the radius-x.x directory
- make; make install

For Windows I recommend to use the *php_radius.dll* from » http://snaps.php.net/.
Unbundled PECL extensions may be downloaded from: » http://pecl4win.php.net/

Runtime Configuration

This extension has no configuration directives defined in *php.ini*.

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.

RADIUS_ACCESS_REQUEST ( )
Authentication Request

RADIUS_ACCESS_ACCEPT ( )
Access accepted

RADIUS_ACCESS_REJECT ( )
Access rejected

RADIUS_ACCOUNTING_REQUEST ( )
Accounting request

RADIUS_ACCOUNTING_RESPONSE ( )
Accounting response

RADIUS_ACCESS_CHALLENGE ( )
Accsess challenge

RADIUS_USER_NAME (string)
Username

RADIUS_USER_PASSWORD (string)
Password

RADIUS_CHAP_PASSWORD (string)
Chap Password: chappass = md5(ident + plaintextpass + challenge)

RADIUS_NAS_IP_ADDRESS (string)
NAS IP-Adress

RADIUS_NAS_PORT (int)
NAS Port

RADIUS_SERVICE_TYPE (int)
Type of Service, one of:

- **RADIUS_LOGIN**
- **RADIUS_FRAMED**
- **RADIUS_CALLBACK_LOGIN**
- **RADIUS_CALLBACK_FRAMED**
- **RADIUS_OUTBOUND**
- **RADIUS_ADMINISTRATIVE**
- **RADIUS_NAS_PROMPT**
- **RADIUS_AUTHENTICATE_ONLY**

- **RADIUS_CALLBACK_NAS_PROMPT**

RADIUS_FRAMED_PROTOCOL (int)
Framed Protocol, one of:

- **RADIUS_PPP**
- **RADIUS_SLIP**
- **RADIUS_ARAP**
- **RADIUS_GANDALF**
- **RADIUS_XYLOGICS**

RADIUS_FRAMED_IP_ADDRESS (string)
IP-Address

RADIUS_FRAMED_IP_NETMASK (string)
Netmask

RADIUS_FRAMED_ROUTING (int)
Routing

RADIUS_FILTER_ID (string)
Filter ID

RADIUS_FRAMED_MTU (int)
MTU

RADIUS_FRAMED_COMPRESSION (int)
Compression, one of:

- **RADIUS_COMP_NONE**
- **RADIUS_COMP_VJ**
- **RADIUS_COMP_IPXHDR**

RADIUS_LOGIN_IP_HOST (string)
Login IP Host

RADIUS_LOGIN_SERVICE (int)
Login Service

RADIUS_LOGIN_TCP_PORT (int)
Login TCP Port

RADIUS_REPLY_MESSAGE (string)
Reply Message

RADIUS_CALLBACK_NUMBER (string)
Callback Number

RADIUS_CALLBACK_ID (string)

Callback ID

RADIUS_FRAMED_ROUTE (string)
Framed Route

RADIUS_FRAMED_IPX_NETWORK (string)
Framed IPX Network

RADIUS_STATE (string)
State

RADIUS_CLASS (int)
Class

RADIUS_VENDOR_SPECIFIC (int)
Vendor specific attribute

RADIUS_SESSION_TIMEOUT (int)
Session timeout

RADIUS_IDLE_TIMEOUT (int)
Idle timeout

RADIUS_TERMINATION_ACTION (int)
Termination action

RADIUS_CALLED_STATION_ID (int)
Called Station Id

RADIUS_CALLING_STATION_ID (string)
Calling Station Id

RADIUS_NAS_IDENTIFIER (int)
NAS ID

RADIUS_PROXY_STATE (int)
Proxy State

RADIUS_LOGIN_LAT_SERVICE (int)
Login LAT Service

RADIUS_LOGIN_LAT_NODE (int)
Login LAT Node

RADIUS_LOGIN_LAT_GROUP (int)
Login LAT Group

RADIUS_FRAMED_APPLETALK_LINK (int)
Framed Appletalk Link

RADIUS_FRAMED_APPLETALK_NETWORK (int)
Framed Appletalk Network

RADIUS_FRAMED_APPLETALK_ZONE (int)
Framed Appletalk Zone

RADIUS_CHAP_CHALLENGE (string)
Challenge

RADIUS_NAS_PORT_TYPE (int)
NAS port type, one of:

- **RADIUS_ASYNC**
- **RADIUS_SYNC**
- **RADIUS_ISDN_SYNC**
- **RADIUS_ISDN_ASYNC_V120**
- **RADIUS_ISDN_ASYNC_V110**
- **RADIUS_VIRTUAL**
- **RADIUS_PIAFS**
- **RADIUS_HDLC_CLEAR_CHANNEL**
- **RADIUS_X_25**
- **RADIUS_X_75**
- **RADIUS_G_3_FAX**
- **RADIUS_SDSL**
- **RADIUS_ADSL_CAP**
- **RADIUS_ADSL_DMT**
- **RADIUS_IDSL**
- **RADIUS_ETHERNET**
- **RADIUS_XDSL**
- **RADIUS_CABLE**
- **RADIUS_WIRELESS_OTHER**
- **RADIUS_WIRELESS_IEEE_802_11**

RADIUS_PORT_LIMIT (int)
Port Limit

RADIUS_LOGIN_LAT_PORT (int)
Login LAT Port

RADIUS_CONNECT_INFO (string)
Connect info

RADIUS_ACCT_STATUS_TYPE (int)
Accounting status type, one of:

- **RADIUS_START**
- **RADIUS_STOP**
- **RADIUS_ACCOUNTING_ON**
- **RADIUS_ACCOUNTING_OFF**

RADIUS_ACCT_DELAY_TIME (int)
Accounting delay time

RADIUS_ACCT_INPUT_OCTETS (int)
Accounting input bytes

RADIUS_ACCT_OUTPUT_OCTETS (int)
Accounting output bytes

RADIUS_ACCT_SESSION_ID (int)
Accounting session ID

RADIUS_ACCT_AUTHENTIC (int)
Accounting authentic, one of:

- **RADIUS_AUTH_RADIUS**
- **RADIUS_AUTH_LOCAL**
- **RADIUS_AUTH_REMOTE**

RADIUS_ACCT_SESSION_TIME (int)
Accounting session time

RADIUS_ACCT_INPUT_PACKETS (int)
Accounting input packets

RADIUS_ACCT_OUTPUT_PACKETS (int)
Accounting output packets

RADIUS_ACCT_TERMINATE_CAUSE (int)
Accounting terminate cause, one of:

- **RADIUS_TERM_USER_REQUEST**
- **RADIUS_TERM_LOST_CARRIER**
- **RADIUS_TERM_LOST_SERVICE**
- **RADIUS_TERM_IDLE_TIMEOUT**
- **RADIUS_TERM_SESSION_TIMEOUT**
- **RADIUS_TERM_ADMIN_RESET**
- **RADIUS_TERM_ADMIN_REBOOT**
- **RADIUS_TERM_PORT_ERROR**
- **RADIUS_TERM_NAS_ERROR**
- **RADIUS_TERM_NAS_REQUEST**
- **RADIUS_TERM_NAS_REBOOT**
- **RADIUS_TERM_PORT_UNNEEDED**
- **RADIUS_TERM_PORT_PREEMPTED**
- **RADIUS_TERM_PORT_SUSPENDED**
- **RADIUS_TERM_SERVICE_UNAVAILABLE**
- **RADIUS_TERM_CALLBACK**
- **RADIUS_TERM_USER_ERROR**
- **RADIUS_TERM_HOST_REQUEST**

RADIUS_ACCT_MULTI_SESSION_ID (string)
Accounting multi session ID

RADIUS_ACCT_LINK_COUNT (int)
Accounting link count

RADIUS_VENDOR_MICROSOFT (int)
Microsoft specific vendor attributes (» RFC 2548), one of:

- **RADIUS_MICROSOFT_MS_CHAP_RESPONSE**
- **RADIUS_MICROSOFT_MS_CHAP_ERROR**
- **RADIUS_MICROSOFT_MS_CHAP_PW_1**
- **RADIUS_MICROSOFT_MS_CHAP_PW_2**
- **RADIUS_MICROSOFT_MS_CHAP_LM_ENC_PW**
- **RADIUS_MICROSOFT_MS_CHAP_NT_ENC_PW**
- **RADIUS_MICROSOFT_MS_MPPE_ENCRYPTION_POLICY**
- **RADIUS_MICROSOFT_MS_MPPE_ENCRYPTION_TYPES**
- **RADIUS_MICROSOFT_MS_RAS_VENDOR**
- **RADIUS_MICROSOFT_MS_CHAP_DOMAIN**
- **RADIUS_MICROSOFT_MS_CHAP_CHALLENGE**
- **RADIUS_MICROSOFT_MS_CHAP_MPPE_KEYS**
- **RADIUS_MICROSOFT_MS_BAP_USAGE**
- **RADIUS_MICROSOFT_MS_LINK_UTILIZATION_THRESHOLD**
- **RADIUS_MICROSOFT_MS_LINK_DROP_TIME_LIMIT**
- **RADIUS_MICROSOFT_MS_MPPE_SEND_KEY**
- **RADIUS_MICROSOFT_MS_MPPE_RECV_KEY**
- **RADIUS_MICROSOFT_MS_RAS_VERSION**
- **RADIUS_MICROSOFT_MS_OLD_ARAP_PASSWORD**
- **RADIUS_MICROSOFT_MS_NEW_ARAP_PASSWORD**
- **RADIUS_MICROSOFT_MS_ARAP_PASSWORD_CHANGE_REASON**
- **RADIUS_MICROSOFT_MS_FILTER**
- **RADIUS_MICROSOFT_MS_ACCT_AUTH_TYPE**
- **RADIUS_MICROSOFT_MS_ACCT_EAP_TYPE**
- **RADIUS_MICROSOFT_MS_CHAP2_RESPONSE**
- **RADIUS_MICROSOFT_MS_CHAP2_SUCCESS**
- **RADIUS_MICROSOFT_MS_CHAP2_PW**
- **RADIUS_MICROSOFT_MS_PRIMARY_DNS_SERVER**
- **RADIUS_MICROSOFT_MS_SECONDARY_DNS_SERVER**
- **RADIUS_MICROSOFT_MS_PRIMARY_NBNS_SERVER**
- **RADIUS_MICROSOFT_MS_SECONDARY_NBNS_SERVER**
- **RADIUS_MICROSOFT_MS_ARAP_CHALLENGE**

Examples

Howto start?

- get a radius resource
- configure the library
- create the request
- put attributes
- send the request
- receive attributes
- close the radius resource (optional)

Take also a look at the examples in this package.

The package contains an example php script. This script demonstrates howto authenticate with radius using PAP or CHAP (md5). If you authenticate with Microsoft Radius servers then its not possible to use CHAP (md5). If you would like to authenticate with Microsoft Servers you have to use MS-CHAPv1 or MS-CHAPv2, but its more complicated, because you need md4, sha1 and des to generate the right data. The enclosed examples demonstrate all authentication-methods, including MS-CHAPv1 and MS-CHAPv2. To get the MS-CHAP to work you need the mcrypt and the mhash extension, starting with version 1.2 of the package, the mcrypt extension is no longer needed.

Radius Functions

Contact Information

If you have comments, bugfixes, enhancements or want to help to develop this you can send me a mail at » mbretter@php.net. Binaries for Windows can be downloaded from » here.

radius_acct_open

radius_acct_open -- Creates a Radius handle for accounting

Description

resource **radius_acct_open** (void)

Return Values

Returns a handle on success, **FALSE** on error. This function only fails if insufficient memory is available.

Examples

Example #1 - radius_acct_open() example

```
<?php
$res = radius_acct_open ()
   or die ("Could not create handle");
print("Handle successfully created");
?>
```

radius_add_server

radius_add_server -- Adds a server

Description

bool **radius_add_server** (resource `$radius_handle`, string `$hostname`, int `$port`, string `$secret`, int `$timeout`, int `$max_tries`)

radius_add_server() may be called multiple times, and it may be used together with radius_config(). At most 10 servers may be specified. When multiple servers are given, they are tried in round-robin fashion until a valid response is received, or until each server's *`max_tries`* limit has been reached.

Parameters

`radius_handle`

`hostname`
: The *`hostname`* parameter specifies the server host, either as a fully qualified domain name or as a dotted-quad IP address in text form.

`port`
: The *`port`* specifies the UDP port to contact on the server. If port is given as 0, the library looks up the `radius/udp` or `radacct/udp` service in the network services database, and uses the port found there. If no entry is found, the library uses the standard Radius ports, 1812 for authentication and 1813 for accounting.

`secret`
: The shared secret for the server host is passed to the *`secret`* parameter. The Radius protocol ignores all but the leading 128 bytes of the shared secret.

`timeout`
: The timeout for receiving replies from the server is passed to the *`timeout`* parameter, in units of seconds.

`max_tries`
: The maximum number of repeated requests to make before giving up is passed into the *`max_tries`*.

Return Values

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

Examples

Example #2 - radius_add_server() **example**

```
<?php
if (!radius_add_server($res, 'radius.example.com', 1812, 'testing123', 3,
3)) {
   echo 'RadiusError:' . radius_strerror($res). "\n<br>";
   exit;
}
?>
```

See Also

- radius_config()

radius_auth_open

radius_auth_open -- Creates a Radius handle for authentication

Description

resource **radius_auth_open** (void)

Return Values

Returns a handle on success, **FALSE** on error. This function only fails if insufficient memory is available.

Examples

Example #3 - radius_auth_open() **example**

```
<?php
$radh = radius_auth_open()
   or die ("Could not create handle");
echo "Handle successfully created";
?>
```

radius_close

radius_close -- Frees all ressources

Description

bool **radius_close** (resource `$radius_handle`)

It is not needed to call this function because php frees all resources at the end of each request.

Return Values

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

radius_config

radius_config -- Causes the library to read the given configuration file

Description

bool **radius_config** (resource `$radius_handle`, string `$file`)

Before issuing any Radius requests, the library must be made aware of the servers it can contact. The easiest way to configure the library is to call radius_config(). radius_config() causes the library to read a configuration file whose format is described in » radius.conf.

Parameters

`radius_handle`

`file`
The pathname of the configuration file is passed as the file argument to radius_config(). The library can also be configured programmatically by calls to radius_add_server().

Return Values

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

See Also

- radius_add_server()

radius_create_request

radius_create_request -- Create accounting or authentication request

Description

bool **radius_create_request** (resource `$radius_handle`, int `$type`)

A Radius request consists of a code specifying the kind of request, and zero or more attributes which provide additional information. To begin constructing a new request, call radius_create_request().

Note
Attention: You must call this function, before you can put any attribute!

Parameters

`radius_handle`
Type is **RADIUS_ACCESS_REQUEST** or **RADIUS_ACCOUNTING_REQUEST**.

`type`
Its description

Return Values

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

Examples

Example #4 - radius_create_request() example

```
<?php
if (!radius_create_request($res, RADIUS_ACCESS_REQUEST)) {
   echo 'RadiusError:' . radius_strerror($res). "\n<br />";
   exit;
}
?>
```

See Also

- radius_send_request()

radius_cvt_addr

radius_cvt_addr -- Converts raw data to IP-Address

Description

string **radius_cvt_addr** (string `$data`)

Examples

Example #5 - radius_cvt_addr() **example**

```
<?php
while ($resa = radius_get_attr($res)) {

   if (!is_array($resa)) {
       printf ("Error getting attribute: %s\n",  radius_strerror($res));
       exit;
   }

   $attr = $resa['attr'];
   $data = $resa['data'];

   switch ($attr) {

   case RADIUS_FRAMED_IP_ADDRESS:
       $ip = radius_cvt_addr($data);
       echo "IP: $ip<br>\n";
       break;

   case RADIUS_FRAMED_IP_NETMASK:
       $mask = radius_cvt_addr($data);
       echo "MASK: $mask<br>\n";
       break;
   }
}
?>
```

See Also

- radius_cvt_int()
- radius_cvt_string()

radius_cvt_int

radius_cvt_int -- Converts raw data to integer

Description

int **radius_cvt_int** (string `$data`)

Examples

Example #6 - radius_cvt_int() example

```
<?php
while ($resa = radius_get_attr($res)) {

   if (!is_array($resa)) {
       printf ("Error getting attribute: %s\n",  radius_strerror($res));
       exit;
   }

   $attr = $resa['attr'];
   $data = $resa['data'];

   switch ($attr) {

   case RADIUS_FRAMED_MTU:
       $mtu = radius_cvt_int($data);
       echo "MTU: $mtu<br>\n";
       break;
   }
}
?>
```

See Also

- radius_cvt_addr()
- radius_cvt_string()

radius_cvt_string

radius_cvt_string -- Converts raw data to string

Description

string **radius_cvt_string** (string `$data`)

Examples

Example #7 - radius_cvt_string() example

```
<?php
while ($resa = radius_get_attr($res)) {

   if (!is_array($resa)) {
       printf ("Error getting attribute: %s\n",  radius_strerror($res));
       exit;
   }

   $attr = $resa['attr'];
   $data = $resa['data'];

   switch ($attr) {

   case RADIUS_FILTER_ID:
       $id = radius_cvt_string($data);
       echo "Filter ID: $id<br>\n";
       break;
   }
}
?>
```

See Also

- radius_cvt_addr()
- radius_cvt_int()

radius_demangle_mppe_key

radius_demangle_mppe_key -- Derives mppe-keys from mangled data

Description

string **radius_demangle_mppe_key** (resource `$radius_handle`, string `$mangled`)

When using MPPE with MS-CHAPv2, the send- and recv-keys are mangled (see » RFC 2548), however this function is useless, because I don't think that there is or will be a PPTP-MPPE implementation in PHP.

Return Values

Returns the demangled string, or **FALSE** on error.

radius_demangle

radius_demangle -- Demangles data

Description

string **radius_demangle** (resource `$radius_handle`, string `$mangled`)

Some data (Passwords, MS-CHAPv1 MPPE-Keys) is mangled for security reasons, and must be demangled before you can use them.

Return Values

Returns the demangled string, or **FALSE** on error.

radius_get_attr

radius_get_attr -- Extracts an attribute

Description

mixed **radius_get_attr** (resource `$radius_handle`)

Like Radius requests, each response may contain zero or more attributes. After a response has been received successfully by radius_send_request(), its attributes can be extracted one by one using radius_get_attr(). Each time radius_get_attr() is called, it gets the next attribute from the current response.

Return Values

Returns an associative array containing the attribute-type and the data, or error number <= 0.

Examples

Example #8 - radius_get_attr() example

```
<?php
while ($resa = radius_get_attr($res)) {

   if (!is_array($resa)) {
       printf("Error getting attribute: %s\n",  radius_strerror($res));
       exit;
   }

   $attr = $resa['attr'];
   $data = $resa['data'];
   printf("Got Attr:%d %d Bytes %s\n", $attr, strlen($data),
bin2hex($data));
}
?>
```

See Also

- radius_put_attr()
- radius_get_vendor_attr()
- radius_put_vendor_attr()
- radius_send_request()

radius_get_vendor_attr

radius_get_vendor_attr -- Extracts a vendor specific attribute

Description

array **radius_get_vendor_attr** (string `$data`)

If radius_get_attr() returns **RADIUS_VENDOR_SPECIFIC**, radius_get_vendor_attr() may be called to determine the vendor.

Return Values

Returns an associative array containing the attribute-type, vendor and the data, or **FALSE** on error.

Examples

Example #9 - radius_get_vendor_attr() example

```
<?php
while ($resa = radius_get_attr($res)) {

   if (!is_array($resa)) {
       printf ("Error getting attribute: %s\n",  radius_strerror($res));
       exit;
   }

   $attr = $resa['attr'];
   $data = $resa['data'];
   printf("Got Attr:%d %d Bytes %s\n", $attr, strlen($data),
bin2hex($data));
   if ($attr == RADIUS_VENDOR_SPECIFIC) {

       $resv = radius_get_vendor_attr($data);
       if (is_array($resv)) {
           $vendor = $resv['vendor'];
           $attrv = $resv['attr'];
           $datav = $resv['data'];
           printf("Got Vendor Attr:%d %d Bytes %s\n", $attrv,
strlen($datav), bin2hex($datav));
       }

   }
}
?>
```

See Also

- radius_get_attr()
- radius_put_vendor_attr()

radius_put_addr

radius_put_addr -- Attaches an IP-Address attribute

Description

bool **radius_put_addr** (resource `$radius_handle`, int `$type`, string `$addr`)

Warning
This function is currently not documented; only its argument list is available.

Return Values

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

radius_put_attr

radius_put_attr -- Attaches a binary attribute

Description

bool **radius_put_attr** (resource `$radius_handle`, int `$type`, string `$value`)

Warning
This function is currently not documented; only its argument list is available.

Return Values

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

Examples

Example #10 - radius_put_attr() **example**

```
<?php
mt_srand(time());
$chall = mt_rand();
$chapval = md5(pack('Ca*',1 , 'sepp' . $chall));
$pass = pack('CH*', 1, $chapval);
if (!radius_put_attr($res, RADIUS_CHAP_PASSWORD, $pass)) {
   echo 'RadiusError:' . radius_strerror($res). "\n<br />";
   exit;
}
?>
```

See Also

- radius_get_attr()
- radius_get_vendor_attr()
- radius_put_vendor_attr()

radius_put_int

radius_put_int -- Attaches an integer attribute

Description

bool **radius_put_int** (resource `$radius_handle`, int `$type`, int `$value`)

Warning
This function is currently not documented; only its argument list is available.

Return Values

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

Examples

Example #11 - radius_put_int() example

```
<?php
if (!radius_put_int($res, RAD_FRAMED_PROTOCOL, RAD_PPP)) {
  echo 'RadiusError:' . radius_strerror($res). "\n<br />";
  exit;
}
?>
```

See Also

- radius_put_string()
- radius_put_vendor_int()
- radius_put_vendor_string()

radius_put_string

radius_put_string -- Attaches a string attribute

Description

bool **radius_put_string** (resource `$radius_handle`, int `$type`, string `$value`)

Warning
This function is currently not documented; only its argument list is available.

Return Values

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

Examples

Example #12 - radius_put_string() **example**

```
<?php
if (!radius_put_string($res, RADIUS_USER_NAME, 'billy')) {
   echo 'RadiusError:' . radius_strerror($res). "\n<br />";
   exit;
}
?>
```

See Also

- radius_put_int()
- radius_put_vendor_int()
- radius_put_vendor_string()

radius_put_vendor_addr

radius_put_vendor_addr -- Attaches a vendor specific IP-Address attribute

Description

bool **radius_put_vendor_addr** (resource `$radius_handle`, int `$vendor`, int `$type`, string `$addr`)

Warning
This function is currently not documented; only its argument list is available.

Return Values

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

radius_put_vendor_attr

radius_put_vendor_attr -- Attaches a vendor specific binary attribute

Description

bool **radius_put_vendor_attr** (resource `$radius_handle`, int `$vendor`, int `$type`, string `$value`)

Warning
This function is currently not documented; only its argument list is available.

Return Values

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

Examples

Example #13 - radius_put_vendor_attr() **example**

```
<?php
if (!radius_put_vendor_attr($res, RADIUS_VENDOR_MICROSOFT,
RAD_MICROSOFT_MS_CHAP_CHALLENGE, $challenge)) {
   echo 'RadiusError:' . radius_strerror($res). "\n<br />";
   exit;
}
?>
```

radius_put_vendor_int

radius_put_vendor_int -- Attaches a vendor specific integer attribute

Description

bool **radius_put_vendor_int** (resource `$radius_handle`, int `$vendor`, int `$type`, int `$value`)

Warning
This function is currently not documented; only its argument list is available.

Return Values

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

radius_put_vendor_string

radius_put_vendor_string -- Attaches a vendor specific string attribute

Description

bool **radius_put_vendor_string** (resource `$radius_handle`, int `$vendor`, int `$type`, string `$value`)

Warning
This function is currently not documented; only its argument list is available.

Return Values

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

radius_request_authenticator

radius_request_authenticator -- Returns the request authenticator

Description

string **radius_request_authenticator** (resource `$radius_handle`)

The request authenticator is needed for demangling mangled data like passwords and encryption-keys.

Return Values

Returns the request authenticator as string, or **FALSE** on error.

See Also

- radius_demangle()

radius_send_request

radius_send_request -- Sends the request and waites for a reply

Description

int **radius_send_request** (resource `$radius_handle`)

After the Radius request has been constructed, it is sent by radius_send_request().

The radius_send_request() function sends the request and waits for a valid reply, retrying the defined servers in round-robin fashion as necessary.

Return Values

If a valid response is received, radius_send_request() returns the Radius code which specifies the type of the response. This will typically be **RADIUS_ACCESS_ACCEPT**, **RADIUS_ACCESS_REJECT**, or **RADIUS_ACCESS_CHALLENGE**. If no valid response is received, radius_send_request() returns **FALSE**.

See Also

- radius_create_request()

radius_server_secret

radius_server_secret -- Returns the shared secret

Description

string **radius_server_secret** (resource `$radius_handle`)

The shared secret is needed as salt for demangling mangled data like passwords and encryption-keys.

Return Values

Returns the server's shared secret as string, or **FALSE** on error.

radius_strerror

radius_strerror -- Returns an error message

Description

string **radius_strerror** (resource `$radius_handle`)

If Radius-functions fail then they record an error message. This error message can be retrieved with this function.

Return Values

Returns error messages as string from failed radius functions.