IMAP, POP3 and NNTP

Introduction

These functions enable you to operate with the IMAP protocol, as well as the NNTP, POP3 and local mailbox access methods.

Be warned however, that some of IMAP functions will not work correctly with the POP protocol.

Installing/Configuring

Requirements

This extension requires the c-client library to be installed. Grab the latest version from » ftp://ftp.cac.washington.edu/imap/ and compile it.

It's important that you do not copy the IMAP source files directly into the system include directory as there may be conflicts. Instead, create a new directory inside the system include directory, such as */usr/local/imap-2000b/* (location and name depend on your setup and IMAP version), and inside this new directory create additional directories named *lib/* and *include/*. From the *c-client* directory from your IMAP source tree, copy all the **.h* files into *include/* and all the **.c* files into *lib/*. Additionally when you compiled IMAP, a file named *c-client.a* was created. Also put this in the *lib/* directory but rename it as *libc-client.a* .

Note
To build the c-client library with SSL or/and Kerberos support read the docs supplied with the package.

Note
In Mandrake Linux, the IMAP library (*libc-client.a*) is compiled without Kerberos support. A separate version with SSL (*client-PHP4.a*) is installed. The library must be recompiled in order to add Kerberos support.

Installation

To get these functions to work, you have to compile PHP with *--with-imap[=DIR]*, where DIR is the c-client install prefix. From our example above, you would use *--with-imap=/usr/local/imap-2000b*. This location depends on where you created this directory according to the description above. Windows users may include the *php_imap.dll* DLL in *php.ini*. IMAP is not supported on systems earlier that Windows 2000. This is because it uses encryption functions in order to enable SSL connections to the mail servers.

Note
Depending how the c-client was configured, you might also need to add *--with-imap-ssl=/path/to/openssl/* and/or *--with-kerberos=/path/to/kerberos* into the PHP configure line.

Warning
The IMAP, recode, YAZ and Cyrus extensions cannot be used in conjuction, because they share the same internal symbols.

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.

NIL (integer)

OP_DEBUG (integer)

OP_READONLY (integer)
Open mailbox read-only

OP_ANONYMOUS (integer)
Don't use or update a*.newsrc* for news (NNTP only)

OP_SHORTCACHE (integer)

OP_SILENT (integer)

OP_PROTOTYPE (integer)

OP_HALFOPEN (integer)
For IMAP and NNTP names, open a connection but don't open a mailbox.

OP_EXPUNGE (integer)

OP_SECURE (integer)

CL_EXPUNGE (integer)
silently expunge the mailbox before closing when calling imap_close()

FT_UID (integer)
The parameter is a UID

FT_PEEK (integer)
Do not set the \Seen flag if not already set

FT_NOT (integer)

FT_INTERNAL (integer)
The return string is in internal format, will not canonicalize to CRLF.

FT_PREFETCHTEXT (integer)

ST_UID (integer)
The sequence argument contains UIDs instead of sequence numbers

ST_SILENT (integer)

ST_SET (integer)

CP_UID (integer)
the sequence numbers contain UIDS

CP_MOVE (integer)
Delete the messages from the current mailbox after copying with imap_mail_copy()

SE_UID (integer)
Return UIDs instead of sequence numbers

SE_FREE (integer)

SE_NOPREFETCH (integer)
Don't prefetch searched messages

SO_FREE (integer)

SO_NOSERVER (integer)

SA_MESSAGES (integer)

SA_RECENT (integer)

SA_UNSEEN (integer)

SA_UIDNEXT (integer)

SA_UIDVALIDITY (integer)

SA_ALL (integer)

LATT_NOINFERIORS (integer)

This mailbox has no "children" (there are no mailboxes below this one).

LATT_NOSELECT (integer)
This is only a container, not a mailbox - you cannot open it.

LATT_MARKED (integer)
This mailbox is marked. Only used by UW-IMAPD.

LATT_UNMARKED (integer)
This mailbox is not marked. Only used by UW-IMAPD.

SORTDATE (integer)
Sort criteria for imap_sort(): message Date

SORTARRIVAL (integer)
Sort criteria for imap_sort(): arrival date

SORTFROM (integer)
Sort criteria for imap_sort(): mailbox in first From address

SORTSUBJECT (integer)
Sort criteria for imap_sort(): message subject

SORTTO (integer)
Sort criteria for imap_sort(): mailbox in first To address

SORTCC (integer)
Sort criteria for imap_sort(): mailbox in first cc address

SORTSIZE (integer)
Sort criteria for imap_sort(): size of message in octets

TYPETEXT (integer)

TYPEMULTIPART (integer)

TYPEMESSAGE (integer)

TYPEAPPLICATION (integer)

TYPEAUDIO (integer)

TYPEIMAGE (integer)

TYPEVIDEO (integer)

TYPEOTHER (integer)

ENC7BIT (integer)

ENC8BIT (integer)

ENCBINARY (integer)

ENCBASE64 (integer)

ENCQUOTEDPRINTABLE (integer)

ENCOTHER (integer)

IMAP_OPENTIMEOUT (integer)

IMAP_READTIMEOUT (integer)

IMAP_WRITETIMEOUT (integer)

IMAP_CLOSETIMEOUT (integer)

LATT_REFERRAL (integer)

LATT_HASCHILDREN (integer)

LATT_HASNOCHILDREN (integer)

TYPEMODEL (integer)

IMAP Functions

See Also

This document can't go into detail on all the topics touched by the provided functions. Further information is provided by the documentation of the c-client library source (*docs/internal.txt*). and the following RFC documents:

- » RFC2821: Simple Mail Transfer Protocol (SMTP).
- » RFC2822: Standard for ARPA internet text messages.
- » RFC2060: Internet Message Access Protocol (IMAP) Version 4rev1.
- » RFC1939: Post Office Protocol Version 3 (POP3).
- » RFC977: Network News Transfer Protocol (NNTP).
- » RFC2076: Common Internet Message Headers.
- » RFC2045, » RFC2046, » RFC2047, » RFC2048 &» RFC2049: Multipurpose Internet Mail Extensions (MIME).

A detailed overview is also available in the books » Programming Internet Email by David Wood and » Managing IMAP by Dianna Mullet & Kevin Mullet.

imap_8bit

imap_8bit -- Convert an 8bit string to a quoted-printable string

Description

string **imap_8bit** (string `$string`)

Convert an 8bit string to a quoted-printable string (according to » RFC2045, section 6.7).

Parameters

`string`
The 8bit string to convert

Return Values

Returns a quoted-printable string.

See Also

- imap_qprint()

imap_alerts

imap_alerts -- Returns all IMAP alert messages that have occurred

Description

array **imap_alerts** (void)

Returns all of the IMAP alert messages generated since the last imap_alerts() call, or the beginning of the page.

When imap_alerts() is called, the alert stack is subsequently cleared. The IMAP specification requires that these messages be passed to the user.

Return Values

Returns an array of all of the IMAP alert messages generated or **FALSE** if no alert messages are available.

See Also

- imap_errors()

imap_append

imap_append -- Append a string message to a specified mailbox

Description

bool **imap_append** (resource `$imap_stream`, string `$mailbox`, string `$message` [, string `$options`])

Appends a string *`message`* to the specified *`mailbox`*.

Parameters

`imap_stream`
An IMAP stream returned by imap_open().

`mailbox`
The mailbox name, see imap_open() for more information

`message`
The message to be append, as a string When talking to the Cyrus IMAP server, you must use "\r\n" as your end-of-line terminator instead of "\n" or the operation will fail

`options`
If provided, the *`options`* will also be written to the *`mailbox`*

Return Values

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

Examples

Example #1 - imap_append() example

```
<?php
$stream = imap_open("{imap.example.org}INBOX.Drafts", "username",
"password");

$check = imap_check($stream);
echo "Msg Count before append: ". $check->Nmsgs . "\n";

imap_append($stream, "{imap.example.org}INBOX.Drafts"
                  , "From: me@example.com\r\n"
                  . "To: you@example.com\r\n"
                  . "Subject: test\r\n"
                  . "\r\n"
```

```
                    . "this is a test message, please ignore\r\n"
                    );

$check = imap_check($stream);
echo "Msg Count after append : ". $check->Nmsgs . "\n";

imap_close($stream);
?>
```

imap_base64

imap_base64 -- Decode BASE64 encoded text

Description

string **imap_base64** (string `$text`)

Decodes the given BASE-64 encoded *`text`*.

Parameters

`text`
: The encoded text

Return Values

Returns the decoded message as a string.

See Also

- imap_binary()
- base64_encode()
- base64_decode()
- » RFC2045, Section 6.8

imap_binary

imap_binary -- Convert an 8bit string to a base64 string

Description

string **imap_binary** (string `$string`)

Convert an 8bit string to a base64 string according to » RFC2045, Section 6.8.

Parameters

`string`
The 8bit string

Return Values

Returns a base64 encoded string.

See Also

- imap_base64()

imap_body

imap_body -- Read the message body

Description

string **imap_body** (resource `$imap_stream`, int `$msg_number` [, int `$options`])

imap_body() returns the body of the message, numbered *`msg_number`* in the current mailbox.

imap_body() will only return a verbatim copy of the message body. To extract single parts of a multipart MIME-encoded message you have to use imap_fetchstructure() to analyze its structure and imap_fetchbody() to extract a copy of a single body component.

Parameters

`imap_stream`
An IMAP stream returned by imap_open().

`msg_number`
The message number

`options`
The optional *`options`* are a bit mask with one or more of the following:

- **FT_UID** - The *`msg_number`* is a UID
- **FT_PEEK** - Do not set the \Seen flag if not already set
- **FT_INTERNAL** - The return string is in internal format, will not canonicalize to CRLF.

Return Values

Returns the body of the specified message, as a string.

imap_bodystruct

imap_bodystruct -- Read the structure of a specified body section of a specific message

Description

object **imap_bodystruct** (resource `$imap_stream`, int `$msg_number`, string `$section`)

Read the structure of a specified body section of a specific message.

Parameters

`imap_stream`
An IMAP stream returned by imap_open().

`msg_number`
The message number

`section`
The body section to read

Return Values

Returns the information in an object, for a detailed description of the object structure and properties see imap_fetchstructure().

See Also

- imap_fetchstructure()

imap_check

imap_check -- Check current mailbox

Description

object **imap_check** (resource `$imap_stream`)

Checks information about the current mailbox.

Parameters

`imap_stream`
An IMAP stream returned by imap_open().

Return Values

Returns the information in an object with following properties:

- Date - current system time formatted according to » RFC2822
- Driver - protocol used to access this mailbox: POP3, IMAP, NNTP
- Mailbox - the mailbox name
- Nmsgs - number of messages in the mailbox
- Recent - number of recent messages in the mailbox

Returns **FALSE** on failure.

Examples

Example #2 - imap_check() example

```
<?php

$imap_obj = imap_check($imap_stream);
var_dump($imap_obj);

?>
```

The above example will output something similar to:

```
object(stdClass)(5) {
 ["Date"]=>
```

```
 string(37) "Wed, 10 Dec 2003 17:56:54 +0100 (CET)"
 ["Driver"]=>
 string(4) "imap"
 ["Mailbox"]=>
 string(54)
 "{www.example.com:143/imap/user="foo@example.com"}INBOX"
 ["Nmsgs"]=>
 int(1)
 ["Recent"]=>
 int(0)
}
```

imap_clearflag_full

imap_clearflag_full -- Clears flags on messages

Description

bool **imap_clearflag_full** (resource `$imap_stream`, string `$sequence`, string `$flag` [, string `$options`])

This function causes a store to delete the specified *`flag`* to the flags set for the messages in the specified *`sequence`*.

Parameters

`imap_stream`
An IMAP stream returned by imap_open().

`sequence`
A sequence of message numbers. You can enumerate desired messages with the *X,Y* syntax, or retrieve all messages within an interval with the *X:Y* syntax

`flag`
The flags which you can unset are "\\Seen", "\\Answered", "\\Flagged", "\\Deleted", and "\\Draft" (as defined by » RFC2060)

`options`
`options` are a bit mask and may contain the single option:

- **ST_UID** - The sequence argument contains UIDs instead of sequence numbers

Return Values

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

See Also

- imap_setflag_full()

imap_close

imap_close -- Close an IMAP stream

Description

bool **imap_close** (resource `$imap_stream` [, int `$flag`])

Closes the imap stream.

Parameters

`imap_stream`
: An IMAP stream returned by imap_open().

`flag`
: If set to **CL_EXPUNGE**, the function will silently expunge the mailbox before closing, removing all messages marked for deletion. You can achieve the same thing by using imap_expunge()

Return Values

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

See Also

- imap_open()

imap_createmailbox

imap_createmailbox -- Create a new mailbox

Description

bool **imap_createmailbox** (resource `$imap_stream`, string `$mailbox`)

Creates a new mailbox specified by *`mailbox`*.

Parameters

`imap_stream`
An IMAP stream returned by imap_open().

`mailbox`
The mailbox name, see imap_open() for more information. Names containing international characters should be encoded by imap_utf7_encode()

Return Values

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

Examples

Example #3 - imap_createmailbox() example

```
<?php
$mbox = imap_open("{imap.example.org}", "username", "password", OP_HALFOPEN)
    or die("can't connect: " . imap_last_error());

$name1 = "phpnewbox";
$name2 = imap_utf7_encode("phpnewb&ouml;x");

$newname = $name1;

echo "Newname will be '$name1'<br />\n";

// we will now create a new mailbox "phptestbox" in your inbox folder,
// check its status after creation and finaly remove it to restore
// your inbox to its initial state

if (@imap_createmailbox($mbox,
imap_utf7_encode("{imap.example.org}INBOX.$newname"))) {
   $status = @imap_status($mbox, "{imap.example.org}INBOX.$newname",
SA_ALL);
   if ($status) {
```

```
        echo "your new mailbox '$name1' has the following status:<br />\n";
        echo "Messages:   " . $status->messages    . "<br />\n";
        echo "Recent:     " . $status->recent      . "<br />\n";
        echo "Unseen:     " . $status->unseen      . "<br />\n";
        echo "UIDnext:    " . $status->uidnext     . "<br />\n";
        echo "UIDvalidity:" . $status->uidvalidity . "<br />\n";

        if (imap_renamemailbox($mbox, "{imap.example.org}INBOX.$newname",
"{imap.example.org}INBOX.$name2")) {
            echo "renamed new mailbox from '$name1' to '$name2'<br />\n";
            $newname = $name2;
        } else {
            echo "imap_renamemailbox on new mailbox failed: " .
imap_last_error() . "<br />\n";
        }
    } else {
        echo "imap_status on new mailbox failed: " . imap_last_error() . "<br
/>\n";
    }

    if (@imap_deletemailbox($mbox, "{imap.example.org}INBOX.$newname")) {
        echo "new mailbox removed to restore initial state<br />\n";
    } else {
        echo "imap_deletemailbox on new mailbox failed: " . implode("<br
/>\n", imap_errors()) . "<br />\n";
    }

} else {
    echo "could not create new mailbox: " . implode("<br />\n",
imap_errors()) . "<br />\n";
}

imap_close($mbox);
?>
```

See Also

- imap_renamemailbox()
- imap_deletemailbox()

imap_delete

imap_delete -- Mark a message for deletion from current mailbox

Description

bool **imap_delete** (resource `$imap_stream`, int `$msg_number` [, int `$options`])

Marks messages listed in *`msg_number`* for deletion. Messages marked for deletion will stay in the mailbox until either imap_expunge() is called or imap_close() is called with the optional parameter **CL_EXPUNGE**.

Parameters

`imap_stream`
An IMAP stream returned by imap_open().

`msg_number`
The message number

`options`
You can set the **FT_UID** which tells the function to treat the *`msg_number`* argument as an *UID*.

Return Values

Returns **TRUE**.

Examples

Example #4 - imap_delete() example

```
<?php

$mbox = imap_open("{imap.example.org}INBOX", "username", "password")
   or die("Can't connect: " . imap_last_error());

$check = imap_mailboxmsginfo($mbox);
echo "Messages before delete: " . $check->Nmsgs . "<br />\n";

imap_delete($mbox, 1);

$check = imap_mailboxmsginfo($mbox);
echo "Messages after  delete: " . $check->Nmsgs . "<br />\n";

imap_expunge($mbox);
```

```
$check = imap_mailboxmsginfo($mbox);
echo "Messages after expunge: " . $check->Nmsgs . "<br />\n";

imap_close($mbox);
?>
```

Notes

Note
POP3 mailboxes do not have their message flags saved between connections, so imap_expunge() must be called during the same connection in order for messages marked for deletion to actually be purged.

See Also

- imap_undelete()
- imap_expunge()
- imap_close()

imap_deletemailbox

imap_deletemailbox -- Delete a mailbox

Description

bool **imap_deletemailbox** (resource `$imap_stream`, string `$mailbox`)

Deletes the specified *`mailbox`*.

Parameters

`imap_stream`
An IMAP stream returned by imap_open().

`mailbox`
The mailbox name, see imap_open() for more information

Return Values

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

See Also

- imap_createmailbox()
- imap_renamemailbox()
- imap_open() for the format of *`mbox`*

imap_errors

imap_errors -- Returns all of the IMAP errors that have occured

Description

array **imap_errors** (void)

Gets all of the IMAP errors (if any) that have occurred during this page request or since the error stack was reset.

When imap_errors() is called, the error stack is subsequently cleared.

Return Values

This function returns an array of all of the IMAP error messages generated since the last imap_errors() call, or the beginning of the page. Returns **FALSE** if no error messages are available.

See Also

- imap_last_error()
- imap_alerts()

imap_expunge

imap_expunge -- Delete all messages marked for deletion

Description

bool **imap_expunge** (resource `$imap_stream`)

Deletes all the messages marked for deletion by imap_delete(), imap_mail_move(), or imap_setflag_full().

Parameters

`imap_stream`
An IMAP stream returned by imap_open().

Return Values

Returns **TRUE**.

imap_fetch_overview

imap_fetch_overview -- Read an overview of the information in the headers of the given message

Description

array **imap_fetch_overview** (resource `$imap_stream`, string `$sequence` [, int `$options`])

This function fetches mail headers for the given *`sequence`* and returns an overview of their contents.

Parameters

`imap_stream`
: An IMAP stream returned by imap_open().

`sequence`
: A message sequence description. You can enumerate desired messages with the *X,Y* syntax, or retrieve all messages within an interval with the *X:Y* syntax

`options`
: *`sequence`* will contain a sequence of message indices or UIDs, if this parameter is set to **FT_UID**.

Return Values

Returns an array of objects describing one message header each. The object will only define a property if it exists. The possible properties are:

- subject - the messages subject
- from - who sent it
- to - recipient
- date - when was it sent
- message_id - Message-ID
- references - is a reference to this message id
- in_reply_to - is a reply to this message id
- size - size in bytes
- uid - UID the message has in the mailbox
- msgno - message sequence number in the mailbox
- recent - this message is flagged as recent

- flagged - this message is flagged
- answered - this message is flagged as answered
- deleted - this message is flagged for deletion
- seen - this message is flagged as already read
- draft - this message is flagged as being a draft

Examples

Example #5 - imap_fetch_overview() example

```
<?php
$mbox = imap_open("{imap.example.org:143}INBOX", "username", "password")
    or die("can't connect: " . imap_last_error());

$MC = imap_check($mbox);

// Fetch an overview for all messages in INBOX
$result = imap_fetch_overview($mbox,"1:{$MC->Nmsgs}",0);
foreach ($result as $overview) {
   echo "#{$overview->msgno} ({$overview->date}) - From: {$overview->from}
   {$overview->subject}\n";
}
imap_close($mbox);
?>
```

See Also

- imap_fetchheader()

imap_fetchbody

imap_fetchbody -- Fetch a particular section of the body of the message

Description

string **imap_fetchbody** (resource `$imap_stream`, int `$msg_number`, string `$part_number` [, int `$options`])

Fetch of a particular section of the body of the specified messages. Body parts are not decoded by this function.

Parameters

`imap_stream`
An IMAP stream returned by imap_open().

`msg_number`
The message number

`part_number`
The part number. It is a string of integers delimited by period which index into a body part list as per the IMAP4 specification

`options`
A bitmask with one or more of the following:

- **FT_UID** - The *`msg_number`* is a UID
- **FT_PEEK** - Do not set the \Seen flag if not already set
- **FT_INTERNAL** - The return string is in internal format, will not canonicalize to CRLF.

Return Values

Returns a particular section of the body of the specified messages as a text string.

See Also

- imap_savebody()
- imap_fetchstructure()

imap_fetchheader

imap_fetchheader -- Returns header for a message

Description

string **imap_fetchheader** (resource `$imap_stream`, int `$msg_number` [, int `$options`])

This function causes a fetch of the complete, unfiltered » RFC2822 format header of the specified message.

Parameters

`imap_stream`
An IMAP stream returned by imap_open().

`msg_number`
The message number

`options`
The possible *`options`* are:

- **FT_UID** - The *`msgno`* argument is a UID
- **FT_INTERNAL** - The return string is in "internal" format, without any attempt to canonicalize to CRLF newlines
- **FT_PREFETCHTEXT** - The RFC822.TEXT should be pre-fetched at the same time. This avoids an extra RTT on an IMAP connection if a full message text is desired (e.g. in a "save to local file" operation)

Return Values

Returns the header of the specified message as a text string.

See Also

- imap_fetch_overview()

imap_fetchstructure

imap_fetchstructure -- Read the structure of a particular message

Description

object **imap_fetchstructure** (resource `$imap_stream`, int `$msg_number` [, int `$options`])

Fetches all the structured information for a given message.

Parameters

`imap_stream`
An IMAP stream returned by imap_open().

`msg_number`
The message number

`options`
This optional parameter only has a single option, **FT_UID**, which tells the function to treat the *`msg_number`* argument as a *UID*.

Return Values

Returns an object includes the envelope, internal date, size, flags and body structure along with a similar object for each mime attachment. The structure of the returned objects is as follows:

Returned Objects for imap_fetchstructure()

type	Primary body type
encoding	Body transfer encoding
ifsubtype	**TRUE** if there is a subtype string
subtype	MIME subtype
ifdescription	**TRUE** if there is a description string
description	Content description string
ifid	**TRUE** if there is an identification string
id	Identification string

lines	Number of lines
bytes	Number of bytes
ifdisposition	**TRUE** if there is a disposition string
disposition	Disposition string
ifdparameters	**TRUE** if the dparameters array exists
dparameters	An array of objects where each object has an "attribute" and a "value" property corresponding to the parameters on the Content-disposition MIME header.
ifparameters	**TRUE** if the parameters array exists
parameters	An array of objects where each object has an "attribute" and a "value" property.
parts	An array of objects identical in structure to the top-level object, each of which corresponds to a MIME body part.

Primary body type

0	text
1	multipart
2	message
3	application
4	audio
5	image
6	video
7	other

Transfer encodings

0	7BIT
1	8BIT
2	BINARY
3	BASE64
4	QUOTED-PRINTABLE
5	OTHER

See Also

- imap_fetchbody()
- imap_bodystruct()

imap_get_quota

imap_get_quota -- Retrieve the quota level settings, and usage statics per mailbox

Description

array **imap_get_quota** (resource `$imap_stream`, string `$quota_root`)

Retrieve the quota level settings, and usage statics per mailbox.

For a non-admin user version of this function, please see the imap_get_quotaroot() function of PHP.

Parameters

`imap_stream`
An IMAP stream returned by imap_open().

`quota_root`
`quota_root` should normally be in the form of *user.name* where name is the mailbox you wish to retrieve information about.

Return Values

Returns an array with integer values limit and usage for the given mailbox. The value of limit represents the total amount of space allowed for this mailbox. The usage value represents the mailboxes current level of capacity. Will return **FALSE** in the case of failure.

As of PHP 4.3, the function more properly reflects the functionality as dictated by the » RFC2087. The array return value has changed to support an unlimited number of returned resources (i.e. messages, or sub-folders) with each named resource receiving an individual array key. Each key value then contains an another array with the usage and limit values within it.

For backwards compatibility reasons, the original access methods are still available for use, although it is suggested to update.

Examples

Example #6 - imap_get_quota() example

```
<?php
$mbox = imap_open("{imap.example.org}", "mailadmin", "password",
OP_HALFOPEN)
     or die("can't connect: " . imap_last_error());
```

```
$quota_value = imap_get_quota($mbox, "user.kalowsky");
if (is_array($quota_value)) {
   echo "Usage level is: " . $quota_value['usage'];
   echo "Limit level is: " . $quota_value['limit'];
}

imap_close($mbox);
?>
```

Example #7 - imap_get_quota() 4.3 or greater example

```
<?php
$mbox = imap_open("{imap.example.org}", "mailadmin", "password",
OP_HALFOPEN)
     or die("can't connect: " . imap_last_error());

$quota_values = imap_get_quota($mbox, "user.kalowsky");
if (is_array($quota_values)) {
  $storage = $quota_values['STORAGE'];
  echo "STORAGE usage level is: " .  $storage['usage'];
  echo "STORAGE limit level is: " .  $storage['limit'];

  $message = $quota_values['MESSAGE'];
  echo "MESSAGE usage level is: " .  $message['usage'];
  echo "MESSAGE limit is: " .  $message['limit'];

  /* ...  */
}

imap_close($mbox);
?>
```

Notes

This function is currently only available to users of the c-client2000 or greater library.

The given *imap_stream* must be opened as the mail administrator, other wise this function will fail.

See Also

- imap_open()
- imap_set_quota()
- imap_get_quotaroot()

imap_get_quotaroot

imap_get_quotaroot -- Retrieve the quota settings per user

Description

array **imap_get_quotaroot** (resource `$imap_stream`, string `$quota_root`)

Retrieve the quota settings per user. The limit value represents the total amount of space allowed for this user's total mailbox usage. The usage value represents the user's current total mailbox capacity.

Parameters

`imap_stream`
An IMAP stream returned by imap_open().

`quota_root`
`quota_root` should normally be in the form of which mailbox (i.e. INBOX).

Return Values

Returns an array of integer values pertaining to the specified user mailbox. All values contain a key based upon the resource name, and a corresponding array with the usage and limit values within.

This function will return **FALSE** in the case of call failure, and an array of information about the connection upon an un-parsable response from the server.

Examples

Example #8 - imap_get_quotaroot() example

```
<?php
$mbox = imap_open("{imap.example.org}", "kalowsky", "password", OP_HALFOPEN)
     or die("can't connect: " . imap_last_error());

$quota = imap_get_quotaroot($mbox, "INBOX");
if (is_array($quota)) {
  $storage = $quota_values['STORAGE'];
  echo "STORAGE usage level is: " .  $storage['usage'];
  echo "STORAGE limit level is: " .  $storage['limit'];

  $message = $quota_values['MESSAGE'];
  echo "MESSAGE usage level is: " .  $message['usage'];
  echo "MESSAGE limit level is: " .  $message['limit'];
```

```
  /* ...  */

}

imap_close($mbox);
?>
```

Notes

This function is currently only available to users of the c-client2000 or greater library.

The *`imap_stream`* should be opened as the user whose mailbox you wish to check.

See Also

- imap_open()
- imap_set_quota()
- imap_get_quota()

imap_getacl

imap_getacl -- Gets the ACL for a given mailbox

Description

array **imap_getacl** (resource `$imap_stream`, string `$mailbox`)

Gets the ACL for a given mailbox.

Parameters

`imap_stream`
An IMAP stream returned by imap_open().

`mailbox`
The mailbox name, see imap_open() for more information

Return Values

Returns an associative array of "folder" => "acl" pairs.

Examples

Example #9 - imap_getacl() example

```
<?php

print_r(imap_getacl($conn_id, 'user.joecool'));

?>
```

The above example will output something similar to:

```
Array
(
   [asubfolder] => lrswipcda
   [anothersubfolder] => lrswipcda
)
```

Notes

This function is currently only available to users of the c-client2000 or greater library.

See Also

- imap_setacl()

imap_getmailboxes

imap_getmailboxes -- Read the list of mailboxes, returning detailed information on each one

Description

array **imap_getmailboxes** (resource `$imap_stream`, string `$ref`, string `$pattern`)

Gets information on the mailboxes.

Parameters

`imap_stream`
An IMAP stream returned by imap_open().

`ref`
`ref` should normally be just the server specification as described in imap_open()

`pattern`
Specifies where in the mailbox hierarchy to start searching. There are two special characters you can pass as part of the *`pattern`*: '*' and '%'. '*' means to return all mailboxes. If you pass *`pattern`* as '*', you will get a list of the entire mailbox hierarchy. '%' means to return the current level only. '%' as the *`pattern`* parameter will return only the top level mailboxes; '~/mail/%' on UW_IMAPD will return every mailbox in the ~/mail directory, but none in subfolders of that directory.

Return Values

Returns an array of objects containing mailbox information. Each object has the attributes *`name`*, specifying the full name of the mailbox; *`delimiter`*, which is the hierarchy delimiter for the part of the hierarchy this mailbox is in; and *`attributes`*. *`Attributes`* is a bitmask that can be tested against:

- **LATT_NOINFERIORS** - This mailbox has no "children" (there are no mailboxes below this one).
- **LATT_NOSELECT** - This is only a container, not a mailbox - you cannot open it.
- **LATT_MARKED** - This mailbox is marked. Only used by UW-IMAPD.
- **LATT_UNMARKED** - This mailbox is not marked. Only used by UW-IMAPD.

Examples

Example #10 - imap_getmailboxes() example

```
<?php
$mbox = imap_open("{imap.example.org}", "username", "password", OP_HALFOPEN)
     or die("can't connect: " . imap_last_error());

$list = imap_getmailboxes($mbox, "{imap.example.org}", "*");
if (is_array($list)) {
   foreach ($list as $key => $val) {
       echo "($key) ";
       echo imap_utf7_decode($val->name) . ",";
       echo "'" . $val->delimiter . "',";
       echo $val->attributes . "<br />\n";
   }
} else {
   echo "imap_getmailboxes failed: " . imap_last_error() . "\n";
}

imap_close($mbox);
?>
```

See Also

- imap_getsubscribed()

imap_getsubscribed

imap_getsubscribed -- List all the subscribed mailboxes

Description

array **imap_getsubscribed** (resource `$imap_stream`, string `$ref`, string `$pattern`)

Gets information on the subscribeds mailboxes.

Identical to imap_getmailboxes(), except that it only returns mailboxes that the user is subscribed to.

Parameters

`imap_stream`
An IMAP stream returned by imap_open().

`ref`
`ref` should normally be just the server specification as described in imap_open()

`pattern`
Specifies where in the mailbox hierarchy to start searching. There are two special characters you can pass as part of the *`pattern`*: '*' and '%'. '*' means to return all mailboxes. If you pass *`pattern`* as '*', you will get a list of the entire mailbox hierarchy. '%' means to return the current level only. '%' as the *`pattern`* parameter will return only the top level mailboxes; '~/mail/%' on UW_IMAPD will return every mailbox in the ~/mail directory, but none in subfolders of that directory.

Return Values

Returns an array of objects containing mailbox information. Each object has the attributes *`name`*, specifying the full name of the mailbox; *`delimiter`*, which is the hierarchy delimiter for the part of the hierarchy this mailbox is in; and *`attributes`*. *`Attributes`* is a bitmask that can be tested against:

- **LATT_NOINFERIORS** - This mailbox has no "children" (there are no mailboxes below this one).
- **LATT_NOSELECT** - This is only a container, not a mailbox - you cannot open it.
- **LATT_MARKED** - This mailbox is marked. Only used by UW-IMAPD.
- **LATT_UNMARKED** - This mailbox is not marked. Only used by UW-IMAPD.

imap_header

imap_header -- Alias of imap_headerinfo()

Description

This function is an alias of: imap_headerinfo().

imap_headerinfo

imap_headerinfo -- Read the header of the message

Description

object **imap_headerinfo** (resource `$imap_stream`, int `$msg_number` [, int `$fromlength` [, int `$subjectlength` [, string `$defaulthost`]]])

Gets information about the given message number by reading its headers.

Parameters

`imap_stream`
An IMAP stream returned by imap_open().

`msg_number`
The message number

`fromlength`
Number of characters for the *fetchfrom* property. Must be greater than or equal to zero.

`subjectlength`
Number of characters for the *fetchsubject* property Must be greater than or equal to zero.

`defaulthost`

Return Values

Returns the information in an object with following properties:

- toaddress - full to: line, up to 1024 characters
- to - an array of objects from the To: line, with the following properties: *personal*, *adl*, *mailbox*, and *host*
- fromaddress - full from: line, up to 1024 characters
- from - an array of objects from the From: line, with the following properties: *personal*, *adl*, *mailbox*, and *host*
- ccaddress - full cc: line, up to 1024 characters
- cc - an array of objects from the Cc: line, with the following properties: *personal*, *adl*, *mailbox*, and *host*
- bccaddress - full bcc: line, up to 1024 characters

- bcc - an array of objects from the Bcc: line, with the following properties: *personal*, *adl*, *mailbox*, and *host*
- reply_toaddress - full Reply-To: line, up to 1024 characters
- reply_to - an array of objects from the Reply-To: line, with the following properties: *personal*, *adl*, *mailbox*, and *host*
- senderaddress - full sender: line, up to 1024 characters
- sender - an array of objects from the Sender: line, with the following properties: *personal*, *adl*, *mailbox*, and *host*
- return_pathaddress - full Return-Path: line, up to 1024 characters
- return_path - an array of objects from the Return-Path: line, with the following properties: *personal*, *adl*, *mailbox*, and *host*
- remail -
- date - The message date as found in its headers
- Date - Same as date
- subject - The message subject
- Subject - Same a subject
- in_reply_to -
- message_id -
- newsgroups -
- followup_to -
- references -
- Recent - *R* if recent and seen, *N* if recent and not seen, ' ' if not recent.
- Unseen - *U* if not seen AND not recent, ' ' if seen OR not seen and recent
- Flagged - *F* if flagged, ' ' if not flagged
- Answered - *A* if answered, ' ' if unanswered
- Deleted - *D* if deleted, ' ' if not deleted
- Draft - *X* if draft, ' ' if not draft
- Msgno - The message number
- MailDate -
- Size - The message size
- udate - mail message date in Unix time
- fetchfrom - from line formatted to fit `fromlength` characters
- fetchsubject - subject line formatted to fit `subjectlength` characters

See Also

- imap_fetch_overview()

imap_headers

imap_headers -- Returns headers for all messages in a mailbox

Description

array **imap_headers** (resource `$imap_stream`)

Returns headers for all messages in a mailbox.

Parameters

`imap_stream`
An IMAP stream returned by imap_open().

Return Values

Returns an array of string formatted with header info. One element per mail message.

imap_last_error

imap_last_error -- Gets the last IMAP error that occurred during this page request

Description

string **imap_last_error** (void)

Gets the full text of the last IMAP error message that occurred on the current page. The error stack is untouched; calling imap_last_error() subsequently, with no intervening errors, will return the same error.

Return Values

Returns the full text of the last IMAP error message that occurred on the current page. Returns **FALSE** if no error messages are available.

See Also

- imap_errors()

imap_list

imap_list -- Read the list of mailboxes

Description

array **imap_list** (resource `$imap_stream`, string `$ref`, string `$pattern`)

Read the list of mailboxes.

Parameters

`imap_stream`
An IMAP stream returned by imap_open().

`ref`
`ref` should normally be just the server specification as described in imap_open().

`pattern`
Specifies where in the mailbox hierarchy to start searching. There are two special characters you can pass as part of the *`pattern`*: '*' and '%'. '*' means to return all mailboxes. If you pass *`pattern`* as '*', you will get a list of the entire mailbox hierarchy. '%' means to return the current level only. '%' as the *`pattern`* parameter will return only the top level mailboxes; '~/mail/%' on UW_IMAPD will return every mailbox in the ~/mail directory, but none in subfolders of that directory.

Return Values

Returns an array containing the names of the mailboxes.

Examples

Example #11 - imap_list() example

```
<?php
$mbox = imap_open("{imap.example.org}", "username", "password", OP_HALFOPEN)
     or die("can't connect: " . imap_last_error());

$list = imap_list($mbox, "{imap.example.org}", "*");
if (is_array($list)) {
   foreach ($list as $val) {
       echo imap_utf7_decode($val) . "\n";
   }
} else {
   echo "imap_list failed: " . imap_last_error() . "\n";
}
```

```
imap_close($mbox);
?>
```

See Also

- imap_getmailboxes()
- imap_lsub()

imap_listmailbox

imap_listmailbox -- Alias of imap_list()

Description

This function is an alias of: imap_list().

imap_listscan

imap_listscan -- Returns the list of mailboxes that matches the given text

Description

array **imap_listscan** (resource `$imap_stream`, string `$ref`, string `$pattern`, string `$content`)

Returns an array containing the names of the mailboxes that have *`content`* in the text of the mailbox.

This function is similar to imap_listmailbox(), but it will additionally check for the presence of the string *`content`* inside the mailbox data.

Parameters

`imap_stream`
: An IMAP stream returned by imap_open().

`ref`
: *`ref`* should normally be just the server specification as described in imap_open()

`pattern`
: Specifies where in the mailbox hierarchy to start searching. There are two special characters you can pass as part of the *`pattern`*: '*' and '%'. '*' means to return all mailboxes. If you pass *`pattern`* as '*', you will get a list of the entire mailbox hierarchy. '%' means to return the current level only. '%' as the *`pattern`* parameter will return only the top level mailboxes; '~/mail/%' on UW_IMAPD will return every mailbox in the ~/mail directory, but none in subfolders of that directory

`content`
: The searched string

Return Values

Returns an array containing the names of the mailboxes that have *`content`* in the text of the mailbox.

See Also

- imap_listmailbox()
- imap_search()

imap_listsubscribed

imap_listsubscribed -- Alias of imap_lsub()

Description

This function is an alias of: imap_lsub().

imap_lsub

imap_lsub -- List all the subscribed mailboxes

Description

array **imap_lsub** (resource `$imap_stream`, string `$ref`, string `$pattern`)

Gets an array of all the mailboxes that you have subscribed.

Parameters

`imap_stream`
An IMAP stream returned by imap_open().

`ref`
`ref` should normally be just the server specification as described in imap_open()

`pattern`
Specifies where in the mailbox hierarchy to start searching. There are two special characters you can pass as part of the *`pattern`*: '*' and '%'. '*' means to return all mailboxes. If you pass *`pattern`* as '*', you will get a list of the entire mailbox hierarchy. '%' means to return the current level only. '%' as the *`pattern`* parameter will return only the top level mailboxes; '~/mail/%' on UW_IMAPD will return every mailbox in the ~/mail directory, but none in subfolders of that directory.

Return Values

Returns an array of all the subscribed mailboxes.

See Also

- imap_list()
- imap_getmailboxes()

imap_mail_compose

imap_mail_compose -- Create a MIME message based on given envelope and body sections

Description

string **imap_mail_compose** (array `$envelope`, array `$body`)

Create a MIME message based on the given *`envelope`* and *`body`* sections.

Parameters

`envelope`
: An associative array of headers fields

`body`
: An indexed array of bodies A body is an associative array which can consist of the following keys: "type", "encoding", "subtype", "description" and "contents.data"

Return Values

Returns the MIME message.

Examples

Example #12 - imap_mail_compose() example

```
<?php

$envelope["from"]= "joe@example.com";
$envelope["to"]  = "foo@example.com";
$envelope["cc"]  = "bar@example.com";

$part1["type"] = TYPEMULTIPART;
$part1["subtype"] = "mixed";

$filename = "/tmp/imap.c.gz";
$fp = fopen($filename, "r");
$contents = fread($fp, filesize($filename));
fclose($fp);

$part2["type"] = TYPEAPPLICATION;
$part2["encoding"] = ENCBINARY;
$part2["subtype"] = "octet-stream";
$part2["description"] = basename($filename);
$part2["contents.data"] = $contents;
```

```
$part3["type"] = TYPETEXT;
$part3["subtype"] = "plain";
$part3["description"] = "description3";
$part3["contents.data"] = "contents.data3\n\n\n\t";

$body[1] = $part1;
$body[2] = $part2;
$body[3] = $part3;

echo nl2br(imap_mail_compose($envelope, $body));

?>
```

imap_mail_copy

imap_mail_copy -- Copy specified messages to a mailbox

Description

bool **imap_mail_copy** (resource `$imap_stream`, string `$msglist`, string `$mailbox` [, int `$options`])

Copies mail messages specified by *`msglist`* to specified mailbox.

Parameters

`imap_stream`
An IMAP stream returned by imap_open().

`msglist`
`msglist` is a range not just message numbers (as described in » RFC2060).

`mailbox`
The mailbox name, see imap_open() for more information

`options`
`options` is a bitmask of one or more of

- **CP_UID** - the sequence numbers contain UIDS
- **CP_MOVE** - Delete the messages from the current mailbox after copying

Return Values

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

See Also

- imap_mail_move()

imap_mail_move

imap_mail_move -- Move specified messages to a mailbox

Description

bool **imap_mail_move** (resource `$imap_stream`, string `$msglist`, string `$mailbox` [, int `$options`])

Moves mail messages specified by *`msglist`* to the specified *`mailbox`*.

Parameters

`imap_stream`
An IMAP stream returned by imap_open().

`msglist`
`msglist` is a range not just message numbers (as described in » RFC2060).

`mailbox`
The mailbox name, see imap_open() for more information

`options`
`options` is a bitmask and may contain the single option:

- **CP_UID** - the sequence numbers contain UIDS

Return Values

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

See Also

- imap_mail_copy()

imap_mail

imap_mail -- Send an email message

Description

bool **imap_mail** (string `$to`, string `$subject`, string `$message` [, string `$additional_headers` [, string `$cc` [, string `$bcc` [, string `$rpath`]]]])

This function allows sending of emails with correct handling of Cc and Bcc receivers.

The parameters *`to`*, *`cc`* and *`bcc`* are all strings and are all parsed as » RFC822 address lists.

Parameters

`to`
The receiver

`subject`
The mail subject

`message`
The mail body

`additional_headers`
As string with additional headers to be set on the mail

`cc`

`bcc`
The receivers specified in *`bcc`* will get the mail, but are excluded from the headers.

`rpath`
Use this parameter to specify return path upon mail delivery failure. This is useful when using PHP as a mail client for multiple users.

Return Values

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

See Also

- mail()

imap_mailboxmsginfo

imap_mailboxmsginfo -- Get information about the current mailbox

Description

object **imap_mailboxmsginfo** (resource `$imap_stream`)

Checks the current mailbox status on the server. It is similar to imap_status(), but will additionally sum up the size of all messages in the mailbox, which will take some additional time to execute.

Parameters

`imap_stream`
An IMAP stream returned by imap_open().

Return Values

Returns the information in an object with following properties:

Mailbox properties

Date	date of last change
Driver	driver
Mailbox	name of the mailbox
Nmsgs	number of messages
Recent	number of recent messages
Unread	number of unread messages
Deleted	number of deleted messages
Size	mailbox size

Returns **FALSE** on failure.

Examples

Example #13 - imap_mailboxmsginfo() example

```
<?php

$mbox = imap_open("{imap.example.org}INBOX", "username", "password")
     or die("can't connect: " . imap_last_error());

$check = imap_mailboxmsginfo($mbox);

if ($check) {
   echo "Date: "     . $check->Date    . "<br />\n" ;
   echo "Driver: "   . $check->Driver  . "<br />\n" ;
   echo "Mailbox: "  . $check->Mailbox . "<br />\n" ;
   echo "Messages: " . $check->Nmsgs   . "<br />\n" ;
   echo "Recent: "   . $check->Recent  . "<br />\n" ;
   echo "Unread: "   . $check->Unread  . "<br />\n" ;
   echo "Deleted: "  . $check->Deleted . "<br />\n" ;
   echo "Size: "     . $check->Size    . "<br />\n" ;
} else {
   echo "imap_check() failed: " . imap_last_error() . "<br />\n";
}

imap_close($mbox);

?>
```

imap_mime_header_decode

imap_mime_header_decode -- Decode MIME header elements

Description

array **imap_mime_header_decode** (string `$text`)

Decodes MIME message header extensions that are non ASCII text (see » RFC2047).

Parameters

`text`
The MIME text

Return Values

The decoded elements are returned in an array of objects, where each object has two properties, *charset* and *text*.

If the element hasn't been encoded, and in other words is in plain US-ASCII, the *charset* property of that element is set to *default*.

Examples

Example #14 - imap_mime_header_decode() example

```
<?php
$text = "=?ISO-8859-1?Q?Keld_J=F8rn_Simonsen?= <keld@example.com>";

$elements = imap_mime_header_decode($text);
for ($i=0; $i<count($elements); $i++) {
   echo "Charset: {$elements[$i]->charset}\n";
   echo "Text: {$elements[$i]->text}\n\n";
}
?>
```

The above example will output:

```
Charset: ISO-8859-1
Text: Keld Jørn Simonsen

Charset: default
Text:  <keld@example.com>
```

In the above example we would have two elements, whereas the first element had previously been encoded with ISO-8859-1, and the second element would be plain US-ASCII.

See Also

- imap_utf8()

imap_msgno

imap_msgno -- Gets the message sequence number for the given UID

Description

int **imap_msgno** (resource `$imap_stream`, int `$uid`)

Returns the message sequence number for the given *`uid`*.

This function is the inverse of imap_uid().

Parameters

`imap_stream`
: An IMAP stream returned by imap_open().

`uid`
: The message UID

Return Values

Returns the message sequence number for the given *`uid`*.

See Also

- imap_uid()

imap_num_msg

imap_num_msg -- Gets the number of messages in the current mailbox

Description

int **imap_num_msg** (resource `$imap_stream`)

Gets the number of messages in the current mailbox.

Parameters

`imap_stream`
An IMAP stream returned by imap_open().

Return Values

Return the number of messages in the current mailbox, as an integer.

See Also

- imap_num_recent()
- imap_status()

imap_num_recent

imap_num_recent -- Gets the number of recent messages in current mailbox

Description

int **imap_num_recent** (resource `$imap_stream`)

Gets the number of recent messages in the current mailbox.

Parameters

`imap_stream`
An IMAP stream returned by imap_open().

Return Values

Returns the number of recent messages in the current mailbox, as an integer.

See Also

- imap_num_msg()
- imap_status()

imap_open

imap_open -- Open an IMAP stream to a mailbox

Description

resource **imap_open** (string `$mailbox`, string `$username`, string `$password` [, int `$options` [, int `$n_retries`]])

Opens an IMAP stream to a *`mailbox`*.

This function can also be used to open streams to POP3 and NNTP servers, but some functions and features are only available on IMAP servers.

Parameters

`mailbox`
: A mailbox name consists of a server and a mailbox path on this server. The special name *INBOX* stands for the current users personal mailbox. Mailbox names that contain international characters besides those in the printable ASCII space have to be encoded width imap_utf7_encode(). The server part, which is enclosed in '{' and '}', consists of the servers name or ip address, an optional port (prefixed by ':'), and an optional protocol specification (prefixed by '/'). The server part is mandatory in all mailbox parameters. All names which start with *{* are remote names, and are in the form *"{" remote_system_name [":" port] [flags] "}" [mailbox_name]* where:

- *remote_system_name* - Internet domain name or bracketed IP address of server.
- *port* - optional TCP port number, default is the default port for that service
- *flags* - optional flags, see following table.
- *mailbox_name* - remote mailbox name, default is INBOX

Optional flags for names

Flag	Description
/service= service	mailbox access service, default is "imap"
/user= user	remote user name for login on the server
/authuser= user	remote authentication user; if specified this is the user name whose password is used (e.g. administrator)
/anonymous	remote access as anonymous user

/debug	record protocol telemetry in application's debug log
/secure	do not transmit a plaintext password over the network
/imap, */imap2*, */imap2bis*, */imap4*, */imap4rev1*	equivalent to /service=imap
/pop3	equivalent to /service=pop3
/nntp	equivalent to /service=nntp
/norsh	do not use rsh or ssh to establish a preauthenticated IMAP session
/ssl	use the Secure Socket Layer to encrypt the session
/validate-cert	validate certificates from TLS/SSL server (this is the default behavior)
/novalidate-cert	do not validate certificates from TLS/SSL server, needed if server uses self-signed certificates
/tls	force use of start-TLS to encrypt the session, and reject connection to servers that do not support it
/notls	do not do start-TLS to encrypt the session, even with servers that support it
/readonly	request read-only mailbox open (IMAP only; ignored on NNTP, and an error with SMTP and POP3)

`username`
The user name

`password`
The password associated with the `username`

`options`
The `options` are a bit mask with one or more of the following:

- **OP_READONLY** - Open mailbox read-only
- **OP_ANONYMOUS** - Don't use or update a*.newsrc* for news (NNTP only)
- **OP_HALFOPEN** - For IMAP and NNTP names, open a connection but don't open a

mailbox.

- **CL_EXPUNGE** - Expunge mailbox automatically upon mailbox close (see also imap_delete() and imap_expunge())
- **OP_DEBUG** - Debug protocol negotiations
- **OP_SHORTCACHE** - Short (elt-only) caching
- **OP_SILENT** - Don't pass up events (internal use)
- **OP_PROTOTYPE** - Return driver prototype
- **OP_SECURE** - Don't do non-secure authentication

n_retries
Number of maximum connect attempts

Return Values

Returns an IMAP stream on success or **FALSE** on error.

ChangeLog

Version	Description
5.2.0	*n_retries* added

Examples

Example #15 - Different use of imap_open()

```
<?php
// To connect to an IMAP server running on port 143 on the local machine,
// do the following:
$mbox = imap_open("{localhost:143}INBOX", "user_id", "password");

// To connect to a POP3 server on port 110 on the local server, use:
$mbox = imap_open ("{localhost:110/pop3}INBOX", "user_id", "password");

// To connect to an SSL IMAP or POP3 server, add /ssl after the protocol
// specification:
$mbox = imap_open ("{localhost:993/imap/ssl}INBOX", "user_id", "password");

// To connect to an SSL IMAP or POP3 server with a self-signed certificate,
// add /ssl/novalidate-cert after the protocol specification:
$mbox = imap_open ("{localhost:995/pop3/ssl/novalidate-cert}", "user_id",
"password");
```

```
// To connect to an NNTP server on port 119 on the local server, use:
$nntp = imap_open ("{localhost:119/nntp}comp.test", "", "");
// To connect to a remote server replace "localhost" with the name or the
// IP address of the server you want to connect to.
?>
```

Example #16 - imap_open() example

```
<?php
$mbox = imap_open("{imap.example.org:143}", "username", "password");

echo "<h1>Mailboxes</h1>\n";
$folders = imap_listmailbox($mbox, "{imap.example.org:143}", "*");

if ($folders == false) {
   echo "Call failed<br />\n";
} else {
   foreach ($folders as $val) {
       echo $val . "<br />\n";
   }
}

echo "<h1>Headers in INBOX</h1>\n";
$headers = imap_headers($mbox);

if ($headers == false) {
   echo "Call failed<br />\n";
} else {
   foreach ($headers as $val) {
       echo $val . "<br />\n";
   }
}

imap_close($mbox);
?>
```

See Also

- imap_close()

imap_ping

imap_ping -- Check if the IMAP stream is still active

Description

bool **imap_ping** (resource `$imap_stream`)

imap_ping() pings the stream to see if it's still active. It may discover new mail; this is the preferred method for a periodic "new mail check" as well as a "keep alive" for servers which have inactivity timeout.

Parameters

`imap_stream`
An IMAP stream returned by imap_open().

Return Values

Returns **TRUE** if the stream is still alive, **FALSE** otherwise.

Examples

Example #17 - imap_ping() Example

```
<?php

$imap = imap_open("{imap.example.org}", "mailadmin", "password");

// after some sleeping
if (!imap_ping($imap)) {
   // do some stuff to reconnect
}

?>
```

imap_qprint

imap_qprint -- Convert a quoted-printable string to an 8 bit string

Description

string **imap_qprint** (string `$string`)

Convert a quoted-printable string to an 8 bit string according to » RFC2045, section 6.7.

Parameters

`string`
A quoted-printable string

Return Values

Returns an 8 bits string.

See Also

- imap_8bit()

imap_renamemailbox

imap_renamemailbox -- Rename an old mailbox to new mailbox

Description

bool **imap_renamemailbox** (resource `$imap_stream`, string `$old_mbox`, string `$new_mbox`)

This function renames on old mailbox to new mailbox (see imap_open() for the format of *`mbox`* names).

Parameters

`imap_stream`
An IMAP stream returned by imap_open().

`old_mbox`
The old mailbox name, see imap_open() for more information

`new_mbox`
The new mailbox name, see imap_open() for more information

Return Values

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

See Also

- imap_createmailbox()
- imap_deletemailbox()

imap_reopen

imap_reopen -- Reopen IMAP stream to new mailbox

Description

bool **imap_reopen** (resource `$imap_stream`, string `$mailbox` [, int `$options` [, int `$n_retries`]])

Reopens the specified stream to a new *`mailbox`* on an IMAP or NNTP server.

Parameters

`imap_stream`
An IMAP stream returned by imap_open().

`mailbox`
The mailbox name, see imap_open() for more information

`options`
The *`options`* are a bit mask with one or more of the following:

- **OP_READONLY** - Open mailbox read-only
- **OP_ANONYMOUS** - Don't use or update a *.newsrc* for news (NNTP only)
- **OP_HALFOPEN** - For IMAP and NNTP names, open a connection but don't open a mailbox.
- **OP_EXPUNGE** - Silently expunge recycle stream
- **CL_EXPUNGE** - Expunge mailbox automatically upon mailbox close (see also imap_delete() and imap_expunge())

`n_retries`
Number of maximum connect attempts

Return Values

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

ChangeLog

Version	Description

5.2.0	*n_retries* added

Examples

Example #18 - imap_reopen() example

```
<?php
$mbox = imap_open("{imap.example.org:143}INBOX", "username", "password") or
die(implode(", ", imap_errors()));
// ...
imap_reopen($mbox, "{imap.example.org:143}INBOX.Sent") or die(implode(", ",
imap_errors()));
// ..
?>
```

imap_rfc822_parse_adrlist

imap_rfc822_parse_adrlist -- Parses an address string

Description

array **imap_rfc822_parse_adrlist** (string `$address`, string `$default_host`)

Parses the address string as defined in » RFC2822 and for each address.

Parameters

`address`
A string containing addresses

`default_host`
The default host name

Return Values

Returns an array of objects. The objects properties are:

- mailbox - the mailbox name (username)
- host - the host name
- personal - the personal name
- adl - at domain source route

Examples

Example #19 - imap_rfc822_parse_adrlist() **example**

```
<?php

$address_string = "Joe Doe <doe@example.com>, postmaster@example.com, root";
$address_array  = imap_rfc822_parse_adrlist($address_string, "example.com");
if (!is_array($address_array) || count($address_array) < 1) {
   die("something is wrong\n");
}

foreach ($address_array as $id => $val) {
   echo "# $id\n";
```

```
   echo "  mailbox : " . $val->mailbox . "\n";
   echo "  host    : " . $val->host . "\n";
   echo "  personal: " . $val->personal . "\n";
   echo "  adl     : " . $val->adl . "\n";
}
?>
```

The above example will output:

```
# 0
 mailbox : doe
 host    : example.com
 personal: Joe Doe
 adl     :
# 1
 mailbox : postmaster
 host    : example.com
 personal:
 adl     :
# 2
 mailbox : root
 host    : example.com
 personal:
 adl     :
```

See Also

- imap_rfc822_parse_headers()

imap_rfc822_parse_headers

imap_rfc822_parse_headers -- Parse mail headers from a string

Description

object **imap_rfc822_parse_headers** (string `$headers` [, string `$defaulthost`])

Gets an object of various header elements, similar to imap_header().

Parameters

`headers`
The parsed headers data

`defaulthost`
The default host name

Return Values

Returns an object similar to the one returned by imap_header(), except for the flags and other properties that come from the IMAP server.

See Also

- imap_rfc822_parse_adrlist()

imap_rfc822_write_address

imap_rfc822_write_address -- Returns a properly formatted email address given the mailbox, host, and personal info

Description

string **imap_rfc822_write_address** (string `$mailbox`, string `$host`, string `$personal`)

Returns a properly formatted email address as defined in » RFC2822 given the needed information.

Parameters

`mailbox`
The mailbox name, see imap_open() for more information

`host`
The email host part

`personal`
The name of the account owner

Return Values

Returns a string properly formatted email address as defined in » RFC2822.

Examples

Example #20 - imap_rfc822_write_address() example

```
<?php
echo imap_rfc822_write_address("hartmut", "example.com", "Hartmut
Holzgraefe");
?>
```

The above example will output:

```
Hartmut Holzgraefe <hartmut@example.com>
```

imap_savebody

imap_savebody -- Save a specific body section to a file

Description

bool **imap_savebody** (resource `$imap_stream`, mixed `$file`, int `$msg_number` [, string `$part_number` [, int `$options`]])

Saves a part or the whole body of the specified message.

Parameters

`imap_stream`
An IMAP stream returned by imap_open().

`file`
The path to the saved file as a string, or a valid file descriptor returned by fopen().

`msg_number`
The message number

`part_number`
The part number. It is a string of integers delimited by period which index into a body part list as per the IMAP4 specification

`options`
A bitmask with one or more of the following:

- **FT_UID** - The *`msg_number`* is a UID
- **FT_PEEK** - Do not set the \Seen flag if not already set
- **FT_INTERNAL** - The return string is in internal format, will not canonicalize to CRLF.

Return Values

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

See Also

- imap_fetchbody()

imap_scanmailbox

imap_scanmailbox -- Alias of imap_listscan()

Description

This function is an alias of: imap_listscan().

imap_search

imap_search -- This function returns an array of messages matching the given search criteria

Description

array **imap_search** (resource `$imap_stream`, string `$criteria` [, int `$options` [, string `$charset`]])

This function performs a search on the mailbox currently opened in the given imap stream.

For example, to match all unanswered messages sent by Mom, you'd use: "UNANSWERED FROM mom". Searches appear to be case insensitive. This list of criteria is from a reading of the UW c-client source code and may be incomplete or inaccurate (see also » RFC2060, section 6.4.4).

Parameters

`imap_stream`
An IMAP stream returned by imap_open().

`criteria`
A string, delimited by spaces, in which the following keywords are allowed. Any multi-word arguments (e.g. FROM "joey smith") must be quoted.

- ALL - return all messages matching the rest of the criteria
- ANSWERED - match messages with the \\ANSWERED flag set
- BCC "string" - match messages with "string" in the Bcc: field
- BEFORE "date" - match messages with Date: before "date"
- BODY "string" - match messages with "string" in the body of the message
- CC "string" - match messages with "string" in the Cc: field
- DELETED - match deleted messages
- FLAGGED - match messages with the \\FLAGGED (sometimes referred to as Important or Urgent) flag set
- FROM "string" - match messages with "string" in the From: field
- KEYWORD "string" - match messages with "string" as a keyword
- NEW - match new messages
- OLD - match old messages
- ON "date" - match messages with Date: matching "date"
- RECENT - match messages with the \\RECENT flag set

- SEEN - match messages that have been read (the \\SEEN flag is set)
- SINCE "date" - match messages with Date: after "date"
- SUBJECT "string" - match messages with "string" in the Subject:
- TEXT "string" - match messages with text "string"
- TO "string" - match messages with "string" in the To:
- UNANSWERED - match messages that have not been answered
- UNDELETED - match messages that are not deleted
- UNFLAGGED - match messages that are not flagged
- UNKEYWORD "string" - match messages that do not have the keyword "string"
- UNSEEN - match messages which have not been read yet

`options`
: Valid values for *`options`* are **SE_UID**, which causes the returned array to contain UIDs instead of messages sequence numbers.

`charset`

Return Values

Returns an array of message numbers or UIDs.

Return **FALSE** if it does not understand the search *`criteria`* or no messages have been found.

ChangeLog

Version	Description
4.3.3	The *`charset`* parameter was added

See Also

- imap_listscan()

imap_set_quota

imap_set_quota -- Sets a quota for a given mailbox

Description

bool **imap_set_quota** (resource `$imap_stream`, string `$quota_root`, int `$quota_limit`)

Sets an upper limit quota on a per mailbox basis.

Parameters

`imap_stream`
An IMAP stream returned by imap_open().

`quota_root`
The mailbox to have a quota set. This should follow the IMAP standard format for a mailbox: *user.name*.

`quota_limit`
The maximum size (in KB) for the *`quota_root`*

Return Values

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

Examples

Example #21 - imap_set_quota() example

```
<?php
$mbox = imap_open("{imap.example.org:143}", "mailadmin", "password");

if (!imap_set_quota($mbox, "user.kalowsky", 3000)) {
   echo "Error in setting quota\n";
   return;
}

imap_close($mbox);
?>
```

Notes

This function is currently only available to users of the c-client2000 or greater library.

The given *imap_stream* must be opened as the mail administrator, other wise this function will fail.

See Also

- imap_open()
- imap_get_quota()

imap_setacl

imap_setacl -- Sets the ACL for a giving mailbox

Description

bool **imap_setacl** (resource `$imap_stream`, string `$mailbox`, string `$id`, string `$rights`)

Sets the ACL for a giving mailbox.

Parameters

`imap_stream`
An IMAP stream returned by imap_open().

`mailbox`
The mailbox name, see imap_open() for more information

`id`
The user to give the rights to.

`rights`
The rights to give to the user. Passing an empty string will delete acl.

Return Values

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

Notes

This function is currently only available to users of the c-client2000 or greater library.

See Also

- imap_getacl()

imap_setflag_full

imap_setflag_full -- Sets flags on messages

Description

bool **imap_setflag_full** (resource `$imap_stream`, string `$sequence`, string `$flag` [, int `$options`])

Causes a store to add the specified *`flag`* to the flags set for the messages in the specified *`sequence`*.

Parameters

`imap_stream`
An IMAP stream returned by imap_open().

`sequence`
A sequence of message numbers. You can enumerate desired messages with the *X,Y* syntax, or retrieve all messages within an interval with the *X:Y* syntax

`flag`
The flags which you can set are *Seen*, *Answered*, *Flagged*, *Deleted*, and *Draft* as defined by » RFC2060.

`options`
A bit mask that may contain the single option:

- **ST_UID** - The sequence argument contains UIDs instead of sequence numbers

Return Values

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

Examples

Example #22 - imap_setflag_full() example

```
<?php
$mbox = imap_open("{imap.example.org:143}", "username", "password")
    or die("can't connect: " . imap_last_error());

$status = imap_setflag_full($mbox, "2,5", "\\Seen \\Flagged");
```

```
echo gettype($status) . "\n";
echo $status . "\n";

imap_close($mbox);
?>
```

See Also

- imap_clearflag_full()

imap_sort

imap_sort -- Gets and sort messages

Description

array **imap_sort** (resource `$imap_stream`, int `$criteria`, int `$reverse` [, int `$options` [, string `$search_criteria` [, string `$charset`]]])

Gets and sorts message numbers by the given parameters.

Parameters

`imap_stream`
An IMAP stream returned by imap_open().

`criteria`
Criteria can be one (and only one) of the following:

- **SORTDATE** - message Date
- **SORTARRIVAL** - arrival date
- **SORTFROM** - mailbox in first From address
- **SORTSUBJECT** - message subject
- **SORTTO** - mailbox in first To address
- **SORTCC** - mailbox in first cc address
- **SORTSIZE** - size of message in octets

`reverse`
Set this to 1 for reverse sorting

`options`
The *`options`* are a bitmask of one or more of the following:

- **SE_UID** - Return UIDs instead of sequence numbers
- **SE_NOPREFETCH** - Don't prefetch searched messages

`search_criteria`

`charset`

Return Values

Returns an array of message numbers sorted by the given parameters.

ChangeLog

Version	Description
4.3.3	The `charset` parameter was added

imap_status

imap_status -- Returns status information on a mailbox

Description

object **imap_status** (resource `$imap_stream`, string `$mailbox`, int `$options`)

Gets status information about the given *`mailbox`*.

Parameters

`imap_stream`
An IMAP stream returned by imap_open().

`mailbox`
The mailbox name, see imap_open() for more information

`options`
Valid flags are:

- **SA_MESSAGES** - set status->messages to the number of messages in the mailbox
- **SA_RECENT** - set status->recent to the number of recent messages in the mailbox
- **SA_UNSEEN** - set status->unseen to the number of unseen (new) messages in the mailbox
- **SA_UIDNEXT** - set status->uidnext to the next uid to be used in the mailbox
- **SA_UIDVALIDITY** - set status->uidvalidity to a constant that changes when uids for the mailbox may no longer be valid
- **SA_ALL** - set all of the above

Return Values

This function returns an object containing status information. The object has the following properties: *messages*, *recent*, *unseen*, *uidnext*, and *uidvalidity*.

flags is also set, which contains a bitmask which can be checked against any of the above constants.

Examples

Example #23 - imap_status() example

```
<?php
$mbox = imap_open("{imap.example.com}", "username", "password", OP_HALFOPEN)
     or die("can't connect: " . imap_last_error());

$status = imap_status($mbox, "{imap.example.org}INBOX", SA_ALL);
if ($status) {
 echo "Messages:   " . $status->messages    . "<br />\n";
 echo "Recent:     " . $status->recent      . "<br />\n";
 echo "Unseen:     " . $status->unseen      . "<br />\n";
 echo "UIDnext:    " . $status->uidnext     . "<br />\n";
 echo "UIDvalidity:" . $status->uidvalidity . "<br />\n";
} else {
 echo "imap_status failed: " . imap_last_error() . "\n";
}

imap_close($mbox);
?>
```

imap_subscribe

imap_subscribe -- Subscribe to a mailbox

Description

bool **imap_subscribe** (resource `$imap_stream`, string `$mailbox`)

Subscribe to a new mailbox.

Parameters

`imap_stream`
An IMAP stream returned by imap_open().

`mailbox`
The mailbox name, see imap_open() for more information

Return Values

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

See Also

- imap_unsubscribe()

imap_thread

imap_thread -- Returns a tree of threaded message

Description

array **imap_thread** (resource `$imap_stream` [, int `$options`])

Gets a tree of a threaded message.

Parameters

`imap_stream`
An IMAP stream returned by imap_open().

`options`

Return Values

imap_thread() returns an associative array containing a tree of messages threaded by *REFERENCES*, or **FALSE** on error.

Every message in the current mailbox will be represented by three entries in the resulting array:

- *$thread["XX.num"]* - current message number
- *$thread["XX.next"]*
- *$thread["XX.branch"]*

Examples

Example #24 - imap_thread() Example

```
<?php

// Here we're outputting the threads of a newsgroup, in HTML

$nntp = imap_open('{news.example.com:119/nntp}some.newsgroup', '', '');
$threads = imap_thread($nntp);

foreach ($threads as $key => $val) {
 $tree = explode('.', $key);
```

```
 if ($tree[1] == 'num') {
   $header = imap_headerinfo($nntp, $val);
   echo "<ul>\n\t<li>" . $header->fromaddress . "\n";
 } elseif ($tree[1] == 'branch') {
   echo "\t</li>\n</ul>\n";
 }
}

imap_close($nntp);

?>
```

imap_timeout

imap_timeout -- Set or fetch imap timeout

Description

mixed **imap_timeout** (int `$timeout_type` [, int `$timeout`])

Sets or fetchs the imap timeout.

Parameters

`timeout_type`
One of the following: **IMAP_OPENTIMEOUT**, **IMAP_READTIMEOUT**, **IMAP_WRITETIMEOUT**, or **IMAP_CLOSETIMEOUT**.

`timeout`
The timeout, in seconds.

Return Values

If the *`timeout`* parameter is set, this function returns **TRUE** on success and **FALSE** on failure.

If *`timeout`* is not provided or evaluates to -1, the current timeout value of *`timeout_type`* is returned as an integer.

Examples

Example #25 - imap_timeout() example

```
<?php

echo "The current read timeout is " . imap_timeout(IMAP_READTIMEOUT) . "\n";

?>
```

imap_uid

imap_uid -- This function returns the UID for the given message sequence number

Description

int **imap_uid** (resource `$imap_stream`, int `$msg_number`)

This function returns the UID for the given message sequence number. An UID is a unique identifier that will not change over time while a message sequence number may change whenever the content of the mailbox changes.

This function is the inverse of imap_msgno().

Parameters

`imap_stream`
An IMAP stream returned by imap_open().

`msg_number`
The message number.

Return Values

The UID of the given message.

Notes

Note
This function is not supported by POP3 mailboxes.

See Also

- imap_msgno()

imap_undelete

imap_undelete -- Unmark the message which is marked deleted

Description

bool **imap_undelete** (resource `$imap_stream`, int `$msg_number` [, int `$flags`])

Removes the deletion flag for a specified message, which is set by imap_delete() or imap_mail_move().

Parameters

`imap_stream`
An IMAP stream returned by imap_open().

`msg_number`
The message number

`flags`

Return Values

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

See Also

- imap_delete()
- imap_mail_move()

imap_unsubscribe

imap_unsubscribe -- Unsubscribe from a mailbox

Description

bool **imap_unsubscribe** (string `$imap_stream`, string `$mailbox`)

Unsubscribe from the specified *`mailbox`*.

Parameters

`imap_stream`
: An IMAP stream returned by imap_open().

`mailbox`
: The mailbox name, see imap_open() for more information

Return Values

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

See Also

- imap_subscribe()

imap_utf7_decode

imap_utf7_decode -- Decodes a modified UTF-7 encoded string

Description

string **imap_utf7_decode** (string `$text`)

Decodes modified UTF-7 *`text`* into ISO-8859-1 string.

This function is needed to decode mailbox names that contain certain characters which are not in range of printable ASCII characters.

Parameters

`text`
: A modified UTF-7 encoding string, as defined in » RFC 2060, section 5.1.3 (original UTF-7 was defined in » RFC1642).

Return Values

Returns a string that is encoded in ISO-8859-1 and consists of the same sequence of characters in *`text`*, or **FALSE** if *`text`* contains invalid modified UTF-7 sequence or *`text`* contains a character that is not part of ISO-8859-1 character set.

See Also

- imap_utf7_encode()

imap_utf7_encode

imap_utf7_encode -- Converts ISO-8859-1 string to modified UTF-7 text

Description

string **imap_utf7_encode** (string `$data`)

Converts *`data`* to modified UTF-7 text.

This is needed to encode mailbox names that contain certain characters which are not in range of printable ASCII characters.

Parameters

`data`
An ISO-8859-1 string.

Return Values

Returns *`data`* encoded with the modified UTF-7 encoding as defined in » RFC 2060, section 5.1.3 (original UTF-7 was defined in » RFC1642).

See Also

- imap_utf7_decode()

imap_utf8

imap_utf8 -- Converts MIME-encoded text to UTF-8

Description

string **imap_utf8** (string `$mime_encoded_text`)

Converts the given *`mime_encoded_text`* to UTF-8.

Parameters

`mime_encoded_text`
: A MIME encoded string. MIME encoding method and the UTF-8 specification are described in » RFC2047 and » RFC2044 respectively.

Return Values

Returns an UTF-8 encoded string.

See Also

- imap_mime_header_decode()