Lightweight Directory Access Protocol

Introduction

LDAP is the Lightweight Directory Access Protocol, and is a protocol used to access "Directory Servers". The Directory is a special kind of database that holds information in a tree structure.

The concept is similar to your hard disk directory structure, except that in this context, the root directory is "The world" and the first level subdirectories are "countries". Lower levels of the directory structure contain entries for companies, organisations or places, while yet lower still we find directory entries for people, and perhaps equipment or documents.

To refer to a file in a subdirectory on your hard disk, you might use something like:
`/usr/local/myapp/docs`
The forwards slash marks each division in the reference, and the sequence is read from left to right.

The equivalent to the fully qualified file reference in LDAP is the "distinguished name", referred to simply as "dn". An example dn might be:
`cn=John Smith,ou=Accounts,o=My Company,c=US`
The comma marks each division in the reference, and the sequence is read from right to left. You would read this dn as:
`country = US organization = My Company organizationalUnit = Accounts commonName = John Smith`
In the same way as there are no hard rules about how you organise the directory structure of a hard disk, a directory server manager can set up any structure that is meaningful for the purpose. However, there are some conventions that are used. The message is that you can not write code to access a directory server unless you know something about its structure, any more than you can use a database without some knowledge of what is available.

Lots of information about LDAP can be found at

- » Mozilla
- » OpenLDAP Project

The Netscape SDK contains a helpful » Programmer's Guide in HTML format.

Installing/Configuring

Requirements

You will need to get and compile LDAP client libraries from either » OpenLDAP or » Bind9.net in order to compile PHP with LDAP support.

Installation

LDAP support in PHP is not enabled by default. You will need to use the *--with-ldap[=DIR]* configuration option when compiling PHP to enable LDAP support. DIR is the LDAP base install directory. To enable SASL support, be sure *--with-ldap-sasl[=DIR]* is used, and that *sasl.h* exists on the system.

Note
Note to Win32 Users In order for this extension to work, there are DLL files that must be available to the Windows system *PATH*. For information on how to do this, see the FAQ entitled " How do I add my PHP directory to the PATH on Windows ". Although copying DLL files from the PHP folder into the Windows system directory also works (because the system directory is by default in the system's *PATH*), this is not recommended. *This extension requires the following files to be in the PATH: libeay32.dll* and *ssleay32.dll* Versions before PHP 4.3.0 additionally require *libsasl.dll*.

In order to use Oracle LDAP libraries, proper Oracle environment has to be set.

Runtime Configuration

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

LDAP configuration options

Name	Default	Changeable	Changelog
ldap.max_links	"-1"	PHP_INI_SYSTEM	

For further details and definitions of the PHP_INI_* constants, see the php.ini directives.

Resource Types

Most LDAP functions operate on or return resources (e.g. ldap_connect() returns a positive LDAP link identifier required by most LDAP functions).

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.

LDAP_DEREF_NEVER (integer)

LDAP_DEREF_SEARCHING (integer)

LDAP_DEREF_FINDING (integer)

LDAP_DEREF_ALWAYS (integer)

LDAP_OPT_DEREF (integer)

LDAP_OPT_SIZELIMIT (integer)

LDAP_OPT_TIMELIMIT (integer)

LDAP_OPT_NETWORK_TIMEOUT (integer)
Option for ldap_set_option() to allow setting network timeout. (Available as of PHP 5.3.0)

LDAP_OPT_PROTOCOL_VERSION (integer)

LDAP_OPT_ERROR_NUMBER (integer)

LDAP_OPT_REFERRALS (integer)

LDAP_OPT_RESTART (integer)

LDAP_OPT_HOST_NAME (integer)

LDAP_OPT_ERROR_STRING (integer)

LDAP_OPT_MATCHED_DN (integer)

LDAP_OPT_SERVER_CONTROLS (integer)

LDAP_OPT_CLIENT_CONTROLS (integer)

LDAP_OPT_DEBUG_LEVEL (integer)

GSLC_SSL_NO_AUTH (integer)

GSLC_SSL_ONEWAY_AUTH (integer)

GSLC_SSL_TWOWAY_AUTH (integer)

Using the PHP LDAP calls

Before you can use the LDAP calls you will need to know ..

- The name or address of the directory server you will use
- The "base dn" of the server (the part of the world directory that is held on this server, which could be "o=My Company,c=US")
- Whether you need a password to access the server (many servers will provide read access for an "anonymous bind" but require a password for anything else)

The typical sequence of LDAP calls you will make in an application will follow this pattern:

```
ldap_connect() // establish connection to server | ldap_bind() // anonymous or authenticated "login" | do something like search or update the directory and display the results | ldap_close() // "logout"
```

Examples

Retrieve information for all entries where the surname starts with "S" from a directory server, displaying an extract with name and email address.

Example #1 - LDAP search example

```
<?php
// basic sequence with LDAP is connect, bind, search, interpret search
// result, close connection

echo "<h3>LDAP query test</h3>";
echo "Connecting ...";
$ds=ldap_connect("localhost");  // must be a valid LDAP server!
echo "connect result is " . $ds . "<br />";

if ($ds) {
   echo "Binding ...";
   $r=ldap_bind($ds);     // this is an "anonymous" bind, typically
                          // read-only access
   echo "Bind result is " . $r . "<br />";

   echo "Searching for (sn=S*) ...";
   // Search surname entry
   $sr=ldap_search($ds, "o=My Company, c=US", "sn=S*");
   echo "Search result is " . $sr . "<br />";

   echo "Number of entires returned is " . ldap_count_entries($ds, $sr) .
"<br />";

   echo "Getting entries ...<p>";
   $info = ldap_get_entries($ds, $sr);
   echo "Data for " . $info["count"] . " items returned:<p>";

   for ($i=0; $i<$info["count"]; $i++) {
       echo "dn is: " . $info[$i]["dn"] . "<br />";
       echo "first cn entry is: " . $info[$i]["cn"][0] . "<br />";
       echo "first email entry is: " . $info[$i]["mail"][0] . "<br /><hr
/>";
   }

   echo "Closing connection";
   ldap_close($ds);

} else {
   echo "<h4>Unable to connect to LDAP server</h4>";
}
?>
```

LDAP Functions

ldap_8859_to_t61

ldap_8859_to_t61 -- Translate 8859 characters to t61 characters

Description

string **ldap_8859_to_t61** (string `$value`)

Translate *ISO-8859* characters to *t61* characters.

This function is useful if you have to talk to a legacy *LDAPv2* server.

Parameters

`value`
: The text to be translated.

Return Values

Return the *t61* translation of *`value`*.

See Also

- ldap_t61_to_8859()

ldap_add

ldap_add -- Add entries to LDAP directory

Description

bool **ldap_add** (resource `$link_identifier`, string `$dn`, array `$entry`)

Add entries in the LDAP directory.

Parameters

`link_identifier`
An LDAP link identifier, returned by ldap_connect().

`dn`
The distinguished name of an LDAP entity.

`entry`
An array that specifies the information about the entry. The values in the entries are indexed by individual attributes. In case of multiple values for an attribute, they are indexed using integers starting with 0.

```
<?php
$entree["attribut1"] = "value";
$entree["attribut2"][0] = "value1";
$entree["attribut2"][1] = "value2";
?>
```

Return Values

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

Examples

Example #2 - Complete example with authenticated bind

```
<?php
$ds = ldap_connect("localhost");  // assuming the LDAP server is on this
host

if ($ds) {
   // bind with appropriate dn to give update access
   $r = ldap_bind($ds, "cn=root, o=My Company, c=US", "secret");
```

```
   // prepare data
   $info["cn"] = "John Jones";
   $info["sn"] = "Jones";
   $info["mail"] = "jonj@example.com";
   $info["objectclass"] = "person";

   // add data to directory
   $r = ldap_add($ds, "cn=John Jones, o=My Company, c=US", $info);

   ldap_close($ds);
} else {
   echo "Unable to connect to LDAP server";
}
?>
```

Notes

Note
This function is binary-safe.

See Also

- ldap_delete()

ldap_bind

ldap_bind -- Bind to LDAP directory

Description

bool **ldap_bind** (resource `$link_identifier` [, string `$bind_rdn` [, string `$bind_password`]])

Binds to the LDAP directory with specified RDN and password.

Parameters

`link_identifier`
An LDAP link identifier, returned by ldap_connect().

`bind_rdn`

`bind_password`

If *`bind_rdn`* and *`bind_password`* are not specified, an anonymous bind is attempted.

Return Values

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

Examples

Example #3 - Using LDAP Bind

```
<?php

// using ldap bind
$ldaprdn  = 'uname';     // ldap rdn or dn
$ldappass = 'password';  // associated password

// connect to ldap server
$ldapconn = ldap_connect("ldap.example.com")
   or die("Could not connect to LDAP server.");

if ($ldapconn) {

   // binding to ldap server
   $ldapbind = ldap_bind($ldapconn, $ldaprdn, $ldappass);
```

```
   // verify binding
   if ($ldapbind) {
       echo "LDAP bind successful...";
   } else {
       echo "LDAP bind failed...";
   }

}

?>
```

Example #4 - Using LDAP Bind Anonymously

```
<?php

//using ldap bind anonymously

// connect to ldap server
$ldapconn = ldap_connect("ldap.example.com")
   or die("Could not connect to LDAP server.");

if ($ldapconn) {

   // binding anonymously
   $ldapbind = ldap_bind($ldapconn);

   if ($ldapbind) {
       echo "LDAP bind anonymous successful...";
   } else {
       echo "LDAP bind anonymous failed...";
   }

}

?>
```

See Also

- ldap_unbind()

ldap_close

ldap_close -- Alias of ldap_unbind()

Description

This function is an alias of: ldap_unbind().

ldap_compare

ldap_compare -- Compare value of attribute found in entry specified with DN

Description

mixed **ldap_compare** (resource `$link_identifier`, string `$dn`, string `$attribute`, string `$value`)

Compare *`value`* of *`attribute`* with value of same attribute in an LDAP directory entry.

Parameters

`link_identifier`
An LDAP link identifier, returned by ldap_connect().

`dn`
The distinguished name of an LDAP entity.

`attribute`
The attribute name.

`value`
The compared value.

Return Values

Returns **TRUE** if *`value`* matches otherwise returns **FALSE**. Returns -1 on error.

Examples

The following example demonstrates how to check whether or not given password matches the one defined in DN specified entry.

Example #5 - Complete example of password check

```
<?php

$ds=ldap_connect("localhost");  // assuming the LDAP server is on this host

if ($ds) {

   // bind
   if (ldap_bind($ds)) {

       // prepare data
```

```
        $dn = "cn=Matti Meikku, ou=My Unit, o=My Company, c=FI";
        $value = "secretpassword";
        $attr = "password";

        // compare value
        $r=ldap_compare($ds, $dn, $attr, $value);

        if ($r === -1) {
            echo "Error: " . ldap_error($ds);
        } elseif ($r === true) {
            echo "Password correct.";
        } elseif ($r === false) {
            echo "Wrong guess! Password incorrect.";
        }

    } else {
        echo "Unable to bind to LDAP server.";
    }

    ldap_close($ds);

} else {
    echo "Unable to connect to LDAP server.";
}
?>
```

Notes

Warning
ldap_compare() can NOT be used to compare BINARY values!

ldap_connect

ldap_connect -- Connect to an LDAP server

Description

resource **ldap_connect** ([string `$hostname` [, int `$port`]])

Establishes a connection to a LDAP server on a specified *`hostname`* and *`port`*.

Parameters

`hostname`
: If you are using OpenLDAP 2.x.x you can specify a URL instead of the hostname. To use LDAP with SSL, compile OpenLDAP 2.x.x with SSL support, configure PHP with SSL, and set this parameter as *ldaps://hostname/*.

`port`
: The port to connect to. Not used when using URLs. Defaults to 389.

Return Values

Returns a positive LDAP link identifier on success, or **FALSE** on error. When OpenLDAP 2.x.x is used, ldap_connect() will always return a resource as it does not actually connect but just initializes the connecting parameters. The actual connect happens with the next calls to ldap_* funcs, usually with ldap_bind().

If no arguments are specified then the link identifier of the already opened link will be returned.

ChangeLog

Version	Description
4.0.4	URL and SSL support was added.

Examples

Example #6 - Example of connecting to LDAP server.

```
<?php

// LDAP variables
$ldaphost = "ldap.example.com";  // your ldap servers
$ldapport = 389;                 // your ldap server's port number

// Connecting to LDAP
$ldapconn = ldap_connect($ldaphost, $ldapport)
         or die("Could not connect to $ldaphost");

?>
```

Example #7 - Example of connecting securely to LDAP server.

```
<?php

// make sure your host is the correct one
// that you issued your secure certificate to
$ldaphost = "ldaps://ldap.example.com/";

// Connecting to LDAP
$ldapconn = ldap_connect($ldaphost)
         or die("Could not connect to {$ldaphost}");

?>
```

See Also

- ldap_bind()

ldap_count_entries

ldap_count_entries -- Count the number of entries in a search

Description

int **ldap_count_entries** (resource `$link_identifier`, resource `$result_identifier`)

Returns the number of entries stored in the result of previous search operations.

Parameters

`link_identifier`
An LDAP link identifier, returned by ldap_connect().

`result_identifier`
The internal LDAP result.

Return Values

Returns number of entries in the result or **FALSE** on error.

ldap_delete

ldap_delete -- Delete an entry from a directory

Description

bool **ldap_delete** (resource `$link_identifier`, string `$dn`)

Deletes a particular entry in LDAP directory.

Parameters

`link_identifier`
An LDAP link identifier, returned by ldap_connect().

`dn`
The distinguished name of an LDAP entity.

Return Values

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

See Also

- ldap_add()

ldap_dn2ufn

ldap_dn2ufn -- Convert DN to User Friendly Naming format

Description

string **ldap_dn2ufn** (string `$dn`)

Turns the specified *dn*, into a more user-friendly form, stripping off type names.

Parameters

dn
: The distinguished name of an LDAP entity.

Return Values

Returns the user friendly name.

ldap_err2str

ldap_err2str -- Convert LDAP error number into string error message

Description

string **ldap_err2str** (int `$errno`)

Returns the string error message explaining the error number *`errno`*. While LDAP errno numbers are standardized, different libraries return different or even localized textual error messages. Never check for a specific error message text, but always use an error number to check.

Parameters

`errno`
The error number.

Return Values

Returns the error message, as a string.

Examples

Example #8 - Enumerating all LDAP error messages

```
<?php
 for ($i=0; $i<100; $i++) {
   printf("Error $i: %s<br />\n", ldap_err2str($i));
 }
?>
```

See Also

- ldap_errno()
- ldap_error()

ldap_errno

ldap_errno -- Return the LDAP error number of the last LDAP command

Description

int **ldap_errno** (resource `$link_identifier`)

Returns the standardized error number returned by the last LDAP command. This number can be converted into a textual error message using ldap_err2str().

Parameters

`link_identifier`
An LDAP link identifier, returned by ldap_connect().

Return Values

Return the LDAP error number of the last LDAP command for this link.

Examples

Unless you lower your warning level in your *php.ini* sufficiently or prefix your LDAP commands with @ (at) characters to suppress warning output, the errors generated will also show up in your HTML output.

Example #9 - Generating and catching an error

```
<?php
// This example contains an error, which we will catch.
$ld = ldap_connect("localhost");
$bind = ldap_bind($ld);
// syntax error in filter expression (errno 87),
// must be "objectclass=*" to work.
$res =  @ldap_search($ld, "o=Myorg, c=DE", "objectclass");
if (!$res) {
   echo "LDAP-Errno: " . ldap_errno($ld) . "<br />\n";
   echo "LDAP-Error: " . ldap_error($ld) . "<br />\n";
   die("Argh!<br />\n");
}
$info = ldap_get_entries($ld, $res);
echo $info["count"] . " matching entries.<br />\n";
?>
```

See Also

- ldap_err2str()
- ldap_error()

ldap_error

ldap_error -- Return the LDAP error message of the last LDAP command

Description

string **ldap_error** (resource `$link_identifier`)

Returns the string error message explaining the error generated by the last LDAP command for the given *`link_identifier`*. While LDAP errno numbers are standardized, different libraries return different or even localized textual error messages. Never check for a specific error message text, but always use an error number to check.

Unless you lower your warning level in your *php.ini* sufficiently or prefix your LDAP commands with @ (at) characters to suppress warning output, the errors generated will also show up in your HTML output.

Parameters

`link_identifier`
An LDAP link identifier, returned by ldap_connect().

Return Values

Returns string error message.

See Also

- ldap_err2str()
- ldap_errno()

ldap_explode_dn

ldap_explode_dn -- Splits DN into its component parts

Description

array **ldap_explode_dn** (string `$dn`, int `$with_attrib`)

Splits the DN returned by ldap_get_dn() and breaks it up into its component parts. Each part is known as Relative Distinguished Name, or RDN.

Parameters

`dn`
: The distinguished name of an LDAP entity.

`with_attrib`
: Used to request if the RDNs are returned with only values or their attributes as well. To get RDNs with the attributes (i.e. in attribute=value format) set *`with_attrib`* to 0 and to get only values set it to 1.

Return Values

Returns an array of all DN components.

ldap_first_attribute

ldap_first_attribute -- Return first attribute

Description

string **ldap_first_attribute** (resource `$link_identifier`, resource `$result_entry_identifier`)

Gets the first attribute in the given entry. Remaining attributes are retrieved by calling ldap_next_attribute() successively.

Similar to reading entries, attributes are also read one by one from a particular entry.

Parameters

`link_identifier`
An LDAP link identifier, returned by ldap_connect().

`result_entry_identifier`

`ber_identifier`
`ber_identifier` is the identifier to internal memory location pointer. It is passed by reference. The same *`ber_identifier`* is passed to ldap_next_attribute(), which modifies that pointer.

Note
This parameter is no longer used as this is now handled automatically by PHP. For backwards compatibility PHP will not throw an error if this parameter is passed.

Return Values

Returns the first attribute in the entry on success and **FALSE** on error.

ChangeLog

Version	Description
5.2.4	The *`ber_identifier`* was removed. This is now handled automatically by PHP.

See Also

- ldap_next_attribute()
- ldap_get_attributes()

ldap_first_entry

ldap_first_entry -- Return first result id

Description

resource **ldap_first_entry** (resource `$link_identifier`, resource `$result_identifier`)

Returns the entry identifier for first entry in the result. This entry identifier is then supplied to ldap_next_entry() routine to get successive entries from the result.

Entries in the LDAP result are read sequentially using the ldap_first_entry() and ldap_next_entry() functions.

Parameters

`link_identifier`
An LDAP link identifier, returned by ldap_connect().

`result_identifier`

Return Values

Returns the result entry identifier for the first entry on success and **FALSE** on error.

See Also

- ldap_get_entries()

ldap_first_reference

ldap_first_reference -- Return first reference

Description

resource **ldap_first_reference** (resource `$link`, resource `$result`)

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

ldap_free_result

ldap_free_result -- Free result memory

Description

bool **ldap_free_result** (resource `$result_identifier`)

Frees up the memory allocated internally to store the result. All result memory will be automatically freed when the script terminates.

Typically all the memory allocated for the LDAP result gets freed at the end of the script. In case the script is making successive searches which return large result sets, ldap_free_result() could be called to keep the runtime memory usage by the script low.

Parameters

`result_identifier`

Return Values

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

ldap_get_attributes

ldap_get_attributes -- Get attributes from a search result entry

Description

array **ldap_get_attributes** (resource `$link_identifier`, resource `$result_entry_identifier`)

Reads attributes and values from an entry in the search result.

Having located a specific entry in the directory, you can find out what information is held for that entry by using this call. You would use this call for an application which "browses" directory entries and/or where you do not know the structure of the directory entries. In many applications you will be searching for a specific attribute such as an email address or a surname, and won't care what other data is held.

```
return_value["count"] = number of attributes in the entry
return_value[0] = first attribute
return_value[n] = nth attribute

return_value["attribute"]["count"] = number of values for attribute
return_value["attribute"][0] = first value of the attribute
return_value["attribute"][i] = (i+1)th value of the attribute
```

Parameters

`link_identifier`
An LDAP link identifier, returned by ldap_connect().

`result_entry_identifier`

Return Values

Returns a complete entry information in a multi-dimensional array on success and **FALSE** on error.

Examples

Example #10 - Show the list of attributes held for a particular directory entry

```
<?php
// $ds is the link identifier for the directory
```

```
// $sr is a valid search result from a prior call to
// one of the ldap directory search calls

$entry = ldap_first_entry($ds, $sr);

$attrs = ldap_get_attributes($ds, $entry);

echo $attrs["count"] . " attributes held for this entry:<p>";

for ($i=0; $i < $attrs["count"]; $i++) {
   echo $attrs[$i] . "<br />";
}
?>
```

See Also

- ldap_first_attribute()
- ldap_next_attribute()

ldap_get_dn

ldap_get_dn -- Get the DN of a result entry

Description

string **ldap_get_dn** (resource `$link_identifier`, resource `$result_entry_identifier`)

Finds out the DN of an entry in the result.

Parameters

`link_identifier`
An LDAP link identifier, returned by ldap_connect().

`result_entry_identifier`

Return Values

Returns the DN of the result entry and **FALSE** on error.

ldap_get_entries

ldap_get_entries -- Get all result entries

Description

array **ldap_get_entries** (resource `$link_identifier`, resource `$result_identifier`)

Reads multiple entries from the given result, and then reading the attributes and multiple values.

Parameters

`link_identifier`
An LDAP link identifier, returned by ldap_connect().

`result_identifier`

Return Values

Returns a complete result information in a multi-dimensional array on success and **FALSE** on error.

The structure of the array is as follows. The attribute index is converted to lowercase. (Attributes are case-insensitive for directory servers, but not when used as array indices.)

```
return_value["count"] = number of entries in the result
return_value[0] : refers to the details of first entry

return_value[i]["dn"] =  DN of the ith entry in the result

return_value[i]["count"] = number of attributes in ith entry
return_value[i][j] = jth attribute in the ith entry in the result

return_value[i]["attribute"]["count"] = number of values for
                                        attribute in ith entry
return_value[i]["attribute"][j] = jth value of attribute in ith entry
```

See Also

- ldap_first_entry()
- ldap_next_entry()

ldap_get_option

ldap_get_option -- Get the current value for given option

Description

bool **ldap_get_option** (resource `$link_identifier`, int `$option`, mixed `&$retval`)

Sets *`retval`* to the value of the specified option.

Parameters

`link_identifier`
An LDAP link identifier, returned by ldap_connect().

`option`
The parameter *`option`* can be one of:

Option	Type
LDAP_OPT_DEREF	integer
LDAP_OPT_SIZELIMIT	integer
LDAP_OPT_TIMELIMIT	integer
LDAP_OPT_NETWORK_TIMEOUT	integer
LDAP_OPT_PROTOCOL_VERSION	integer
LDAP_OPT_ERROR_NUMBER	integer
LDAP_OPT_REFERRALS	bool
LDAP_OPT_RESTART	bool
LDAP_OPT_HOST_NAME	string
LDAP_OPT_ERROR_STRING	string
LDAP_OPT_MATCHED_DN	string
LDAP_OPT_SERVER_CONTROLS	array
LDAP_OPT_CLIENT_CONTROLS	array

retval
This will be set to the option value.

Return Values

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

Examples

Example #11 - Check protocol version

```
<?php
// $ds is a valid link identifier for a directory server
if (ldap_get_option($ds, LDAP_OPT_PROTOCOL_VERSION, $version)) {
   echo "Using protocol version $version\n";
} else {
   echo "Unable to determine protocol version\n";
}
?>
```

Notes

Note

This function is only available when using OpenLDAP 2.x.x OR Netscape Directory SDK x.x.

See Also

- ldap_set_option()

ldap_get_values_len

ldap_get_values_len -- Get all binary values from a result entry

Description

array **ldap_get_values_len** (resource `$link_identifier`, resource `$result_entry_identifier`, string `$attribute`)

Reads all the values of the attribute in the entry in the result.

This function is used exactly like ldap_get_values() except that it handles binary data and not string data.

Parameters

`link_identifier`
An LDAP link identifier, returned by ldap_connect().

`result_entry_identifier`

`attribute`

Return Values

Returns an array of values for the attribute on success and **FALSE** on error. Individual values are accessed by integer index in the array. The first index is 0. The number of values can be found by indexing "count" in the resultant array.

See Also

- ldap_get_values()

ldap_get_values

ldap_get_values -- Get all values from a result entry

Description

array **ldap_get_values** (resource `$link_identifier`, resource `$result_entry_identifier`, string `$attribute`)

Reads all the values of the attribute in the entry in the result.

This call needs a *`result_entry_identifier`*, so needs to be preceded by one of the ldap search calls and one of the calls to get an individual entry.

You application will either be hard coded to look for certain attributes (such as "surname" or "mail") or you will have to use the ldap_get_attributes() call to work out what attributes exist for a given entry.

Parameters

`link_identifier`
An LDAP link identifier, returned by ldap_connect().

`result_entry_identifier`

`attribute`

Return Values

Returns an array of values for the attribute on success and **FALSE** on error. The number of values can be found by indexing "count" in the resultant array. Individual values are accessed by integer index in the array. The first index is 0.

LDAP allows more than one entry for an attribute, so it can, for example, store a number of email addresses for one person's directory entry all labeled with the attribute "mail"
`return_value["count"] = number of values for attribute return_value[0] = first value of attribute return_value[i] = ith value of attribute`

Examples

Example #12 - List all values of the "mail" attribute for a directory entry

```
<?php
// $ds is a valid link identifier for a directory server

// $sr is a valid search result from a prior call to
//     one of the ldap directory search calls

// $entry is a valid entry identifier from a prior call to
//        one of the calls that returns a directory entry

$values = ldap_get_values($ds, $entry, "mail");

echo $values["count"] . " email addresses for this entry.<br />";

for ($i=0; $i < $values["count"]; $i++) {
   echo $values[$i] . "<br />";
}
?>
```

See Also

- ldap_get_values_len()

ldap_list

ldap_list -- Single-level search

Description

resource **ldap_list** (resource `$link_identifier`, string `$base_dn`, string `$filter` [, array `$attributes` [, int `$attrsonly` [, int `$sizelimit` [, int `$timelimit` [, int `$deref`]]]]])

Performs the search for a specified *`filter`* on the directory with the scope **LDAP_SCOPE_ONELEVEL**.

LDAP_SCOPE_ONELEVEL means that the search should only return information that is at the level immediately below the *`base_dn`* given in the call. (Equivalent to typing "ls" and getting a list of files and folders in the current working directory.)

Parameters

`link_identifier`
An LDAP link identifier, returned by ldap_connect().

`base_dn`
The base DN for the directory.

`filter`

`attributes`
An array of the required attributes, e.g. array("mail", "sn", "cn"). Note that the "dn" is always returned irrespective of which attributes types are requested. Using this parameter is much more efficient than the default action (which is to return all attributes and their associated values). The use of this parameter should therefore be considered good practice.

`attrsonly`
Should be set to 1 if only attribute types are wanted. If set to 0 both attributes types and attribute values are fetched which is the default behaviour.

`sizelimit`
Enables you to limit the count of entries fetched. Setting this to 0 means no limit.

Note
This parameter can NOT override server-side preset sizelimit. You can set it lower though. Some directory server hosts will be configured to return no more than a preset number of entries. If this occurs, the server will indicate that it has only returned a partial results

set. This also occurs if you use this parameter to limit the count of fetched entries.

timelimit
Sets the number of seconds how long is spend on the search. Setting this to 0 means no limit.

Note
This parameter can NOT override server-side preset timelimit. You can set it lower though.

deref
Specifies how aliases should be handled during the search. It can be one of the following:

- **LDAP_DEREF_NEVER** - (default) aliases are never dereferenced.
- **LDAP_DEREF_SEARCHING** - aliases should be dereferenced during the search but not when locating the base object of the search.
- **LDAP_DEREF_FINDING** - aliases should be dereferenced when locating the base object but not during the search.
- **LDAP_DEREF_ALWAYS** - aliases should be dereferenced always.

Return Values

Returns a search result identifier or **FALSE** on error.

ChangeLog

Version	Description
4.0.5	Parallel searches support was added. See ldap_search() for details.
4.0.2	The *attrsonly*, *sizelimit*, *timelimit* and *deref* were added.

Examples

Example #13 - Produce a list of all organizational units of an organization

```
// $ds is a valid link identifier for a directory server

$basedn = "o=My Company, c=US";
$justthese = array("ou");

$sr=ldap_list($ds, $basedn, "ou=*", $justthese);

$info = ldap_get_entries($ds, $sr);

for ($i=0; $i<$info["count"]; $i++) {
   echo $info[$i]["ou"][0] ;
}
```

See Also

- ldap_search()

ldap_mod_add

ldap_mod_add -- Add attribute values to current attributes

Description

bool **ldap_mod_add** (resource `$link_identifier`, string `$dn`, array `$entry`)

Adds one or more attributes to the specified *`dn`*. It performs the modification at the attribute level as opposed to the object level. Object-level additions are done by the ldap_add() function.

Parameters

`link_identifier`
An LDAP link identifier, returned by ldap_connect().

`dn`
The distinguished name of an LDAP entity.

`entry`

Return Values

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

Notes

Note
This function is binary-safe.

See Also

- ldap_mod_del()
- ldap_mod_replace()

ldap_mod_del

ldap_mod_del -- Delete attribute values from current attributes

Description

bool **ldap_mod_del** (resource `$link_identifier`, string `$dn`, array `$entry`)

Removes one or more attributes from the specified *`dn`*. It performs the modification at the attribute level as opposed to the object level. Object-level deletions are done by the ldap_delete() function.

Parameters

`link_identifier`
An LDAP link identifier, returned by ldap_connect().

`dn`
The distinguished name of an LDAP entity.

`entry`

Return Values

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

See Also

- ldap_mod_add()
- ldap_mod_replace()

ldap_mod_replace

ldap_mod_replace -- Replace attribute values with new ones

Description

bool **ldap_mod_replace** (resource `$link_identifier`, string `$dn`, array `$entry`)

Replaces one or more attributes from the specified *`dn`*. It performs the modification at the attribute level as opposed to the object level. Object-level modifications are done by the ldap_modify() function.

Parameters

`link_identifier`
An LDAP link identifier, returned by ldap_connect().

`dn`
The distinguished name of an LDAP entity.

`entry`

Return Values

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

Notes

Note
This function is binary-safe.

See Also

- ldap_mod_del()
- ldap_mod_add()

ldap_modify

ldap_modify -- Modify an LDAP entry

Description

bool **ldap_modify** (resource `$link_identifier`, string `$dn`, array `$entry`)

Modify the existing entries in the LDAP directory. The structure of the entry is same as in ldap_add().

Parameters

`link_identifier`
An LDAP link identifier, returned by ldap_connect().

`dn`
The distinguished name of an LDAP entity.

`entry`

Return Values

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

Notes

Note
This function is binary-safe.

See Also

- ldap_rename()

ldap_next_attribute

ldap_next_attribute -- Get the next attribute in result

Description

string **ldap_next_attribute** (resource `$link_identifier`, resource `$result_entry_identifier`)

Retrieves the attributes in an entry. The first call to ldap_next_attribute() is made with the *`result_entry_identifier`* returned from ldap_first_attribute().

Parameters

`link_identifier`
An LDAP link identifier, returned by ldap_connect().

`result_entry_identifier`

`ber_identifier`
The internal state of the pointer is maintained by this parameter.

Note
This parameter is no longer used as this is now handled automatically by PHP. For backwards compatibility PHP will not throw an error if this parameter is passed.

Return Values

Returns the next attribute in an entry on success and **FALSE** on error.

ChangeLog

Version	Description
5.2.4	The *`ber_identifier`* was removed. This is now handled automatically by PHP.

See Also

- ldap_get_attributes()

ldap_next_entry

ldap_next_entry -- Get next result entry

Description

resource **ldap_next_entry** (resource `$link_identifier`, resource `$result_entry_identifier`)

Retrieve the entries stored in the result. Successive calls to the ldap_next_entry() return entries one by one till there are no more entries. The first call to ldap_next_entry() is made after the call to ldap_first_entry() with the *`result_entry_identifier`* as returned from the ldap_first_entry().

Parameters

`link_identifier`
An LDAP link identifier, returned by ldap_connect().

`result_entry_identifier`

Return Values

Returns entry identifier for the next entry in the result whose entries are being read starting with ldap_first_entry(). If there are no more entries in the result then it returns **FALSE**.

See Also

- ldap_get_entries()

ldap_next_reference

ldap_next_reference -- Get next reference

Description

resource **ldap_next_reference** (resource `$link`, resource `$entry`)

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

ldap_parse_reference

ldap_parse_reference -- Extract information from reference entry

Description

bool **ldap_parse_reference** (resource `$link`, resource `$entry`, array `&$referrals`)

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

ldap_parse_result

ldap_parse_result -- Extract information from result

Description

bool **ldap_parse_result** (resource `$link`, resource `$result`, int `&$errcode` [, string `&$matcheddn` [, string `&$errmsg` [, array `&$referrals`]]])

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

ldap_read

ldap_read -- Read an entry

Description

resource **ldap_read** (resource `$link_identifier`, string `$base_dn`, string `$filter` [, array `$attributes` [, int `$attrsonly` [, int `$sizelimit` [, int `$timelimit` [, int `$deref`]]]]])

Performs the search for a specified *`filter`* on the directory with the scope **LDAP_SCOPE_BASE**. So it is equivalent to reading an entry from the directory.

Parameters

`link_identifier`
An LDAP link identifier, returned by ldap_connect().

`base_dn`
The base DN for the directory.

`filter`
An empty filter is not allowed. If you want to retrieve absolutely all information for this entry, use a filter of *objectClass*=*. If you know which entry types are used on the directory server, you might use an appropriate filter such as *objectClass=inetOrgPerson*.

`attributes`
An array of the required attributes, e.g. array("mail", "sn", "cn"). Note that the "dn" is always returned irrespective of which attributes types are requested. Using this parameter is much more efficient than the default action (which is to return all attributes and their associated values). The use of this parameter should therefore be considered good practice.

`attrsonly`
Should be set to 1 if only attribute types are wanted. If set to 0 both attributes types and attribute values are fetched which is the default behaviour.

`sizelimit`
Enables you to limit the count of entries fetched. Setting this to 0 means no limit.

Note
This parameter can NOT override server-side preset sizelimit. You can set it lower though. Some directory server hosts will be configured to return no more than a preset number of entries. If this occurs, the server will indicate that it has only returned a partial results set. This also occurs if you use this parameter to limit the count of

fetched entries.

timelimit
: Sets the number of seconds how long is spend on the search. Setting this to 0 means no limit.

Note
This parameter can NOT override server-side preset timelimit. You can set it lower though.

deref
: Specifies how aliases should be handled during the search. It can be one of the following:

- **LDAP_DEREF_NEVER** - (default) aliases are never dereferenced.
- **LDAP_DEREF_SEARCHING** - aliases should be dereferenced during the search but not when locating the base object of the search.
- **LDAP_DEREF_FINDING** - aliases should be dereferenced when locating the base object but not during the search.
- **LDAP_DEREF_ALWAYS** - aliases should be dereferenced always.

Return Values

Returns a search result identifier or **FALSE** on error.

ChangeLog

Version	Description
4.0.5	Parallel searches support was added. See ldap_search() for details.
4.0.2	The *attrsonly*, *sizelimit*, *timelimit* and *deref* were added.

ldap_rename

ldap_rename -- Modify the name of an entry

Description

bool **ldap_rename** (resource `$link_identifier`, string `$dn`, string `$newrdn`, string `$newparent`, bool `$deleteoldrdn`)

The entry specified by *`dn`* is renamed/moved.

Parameters

`link_identifier`
: An LDAP link identifier, returned by ldap_connect().

`dn`
: The distinguished name of an LDAP entity.

`newrdn`
: The new RDN.

`newparent`
: The new parent/superior entry.

`deleteoldrdn`
: If **TRUE** the old RDN value(s) is removed, else the old RDN value(s) is retained as non-distinguished values of the entry.

Return Values

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

Notes

Note
This function currently only works with LDAPv3. You may have to use ldap_set_option() prior to binding to use LDAPv3. This function is only available when using OpenLDAP 2.x.x OR Netscape Directory SDK x.x.

See Also

- ldap_modify()

ldap_sasl_bind

ldap_sasl_bind -- Bind to LDAP directory using SASL

Description

bool **ldap_sasl_bind** (resource `$link` [, string `$binddn` [, string `$password` [, string `$sasl_mech` [, string `$sasl_realm` [, string `$sasl_authz_id` [, string `$props`]]]]]])

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

Return Values

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

Notes

Note
Requirement ldap_sasl_bind() requires SASL support (*sasl.h*). Be sure *--with-ldap-sasl* is used when configuring PHP otherwise this function will be undefined.

ldap_search

ldap_search -- Search LDAP tree

Description

resource **ldap_search** (resource `$link_identifier`, string `$base_dn`, string `$filter` [, array `$attributes` [, int `$attrsonly` [, int `$sizelimit` [, int `$timelimit` [, int `$deref`]]]]])

Performs the search for a specified filter on the directory with the scope of **LDAP_SCOPE_SUBTREE**. This is equivalent to searching the entire directory.

From 4.0.5 on it's also possible to do parallel searches. To do this you use an array of link identifiers, rather than a single identifier, as the first argument. If you don't want the same base DN and the same filter for all the searches, you can also use an array of base DNs and/or an array of filters. Those arrays must be of the same size as the link identifier array since the first entries of the arrays are used for one search, the second entries are used for another, and so on. When doing parallel searches an array of search result identifiers is returned, except in case of error, then the entry corresponding to the search will be **FALSE** . This is very much like the value normally returned, except that a result identifier is always returned when a search was made. There are some rare cases where the normal search returns **FALSE** while the parallel search returns an identifier.

Parameters

`link_identifier`
An LDAP link identifier, returned by ldap_connect().

`base_dn`
The base DN for the directory.

`filter`
The search filter can be simple or advanced, using boolean operators in the format described in the LDAP documentation (see the » Netscape Directory SDK for full information on filters).

`attributes`
An array of the required attributes, e.g. array("mail", "sn", "cn"). Note that the "dn" is always returned irrespective of which attributes types are requested. Using this parameter is much more efficient than the default action (which is to return all attributes and their associated values). The use of this parameter should therefore be considered good practice.

`attrsonly`
Should be set to 1 if only attribute types are wanted. If set to 0 both attributes types and attribute values are fetched which is the default behaviour.

sizelimit
Enables you to limit the count of entries fetched. Setting this to 0 means no limit.

Note
This parameter can NOT override server-side preset sizelimit. You can set it lower though. Some directory server hosts will be configured to return no more than a preset number of entries. If this occurs, the server will indicate that it has only returned a partial results set. This also occurs if you use this parameter to limit the count of fetched entries.

timelimit
Sets the number of seconds how long is spend on the search. Setting this to 0 means no limit.

Note
This parameter can NOT override server-side preset timelimit. You can set it lower though.

deref
Specifies how aliases should be handled during the search. It can be one of the following:

- **LDAP_DEREF_NEVER** - (default) aliases are never dereferenced.
- **LDAP_DEREF_SEARCHING** - aliases should be dereferenced during the search but not when locating the base object of the search.
- **LDAP_DEREF_FINDING** - aliases should be dereferenced when locating the base object but not during the search.
- **LDAP_DEREF_ALWAYS** - aliases should be dereferenced always.

Return Values

Returns a search result identifier or **FALSE** on error.

ChangeLog

Version	Description

4.0.5	Parallel searches support was added.
4.0.2	The *attrsonly*, *sizelimit*, *timelimit* and *deref* were added.

Examples

The example below retrieves the organizational unit, surname, given name and email address for all people in "My Company" where the surname or given name contains the substring $person. This example uses a boolean filter to tell the server to look for information in more than one attribute.

Example #14 - LDAP search

```
<?php
// $ds is a valid link identifier for a directory server

// $person is all or part of a person's name, eg "Jo"

$dn = "o=My Company, c=US";
$filter="(|(sn=$person*)(givenname=$person*))";
$justthese = array("ou", "sn", "givenname", "mail");

$sr=ldap_search($ds, $dn, $filter, $justthese);

$info = ldap_get_entries($ds, $sr);

echo $info["count"]." entries returned\n";
?>
```

ldap_set_option

ldap_set_option -- Set the value of the given option

Description

bool **ldap_set_option** (resource `$link_identifier`, int `$option`, mixed `$newval`)

Sets the value of the specified option to be *`newval`*.

Parameters

`link_identifier`
An LDAP link identifier, returned by ldap_connect().

`option`
The parameter *`option`* can be one of:

Option	Type
LDAP_OPT_DEREF	integer
LDAP_OPT_SIZELIMIT	integer
LDAP_OPT_TIMELIMIT	integer
LDAP_OPT_NETWORK_TIMEOUT	integer
LDAP_OPT_PROTOCOL_VERSION	integer
LDAP_OPT_ERROR_NUMBER	integer
LDAP_OPT_REFERRALS	bool
LDAP_OPT_RESTART	bool
LDAP_OPT_HOST_NAME	string
LDAP_OPT_ERROR_STRING	string
LDAP_OPT_MATCHED_DN	string
LDAP_OPT_SERVER_CONTROLS	array
LDAP_OPT_CLIENT_CONTROLS	array

LDAP_OPT_SERVER_CONTROLS and **LDAP_OPT_CLIENT_CONTROLS** require a list of controls, this means that the value must be an array of controls. A control consists of an

oid identifying the control, an optional *value*, and an optional flag for *criticality*. In PHP a control is given by an array containing an element with the key *oid* and string value, and two optional elements. The optional elements are key *value* with string value and key *iscritical* with boolean value. *iscritical* defaults to **FALSE** if not supplied. See » draft-ietf-ldapext-ldap-c-api-xx.txt for details. See also the second example below.

newval
The new value for the specified *option*.

Return Values

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

Examples

Example #15 - Set protocol version

```
<?php
// $ds is a valid link identifier for a directory server
if (ldap_set_option($ds, LDAP_OPT_PROTOCOL_VERSION, 3)) {
   echo "Using LDAPv3";
} else {
   echo "Failed to set protocol version to 3";
}
?>
```

Example #16 - Set server controls

```
<?php
// $ds is a valid link identifier for a directory server
// control with no value
$ctrl1 = array("oid" => "1.2.752.58.10.1", "iscritical" => true);
// iscritical defaults to FALSE
$ctrl2 = array("oid" => "1.2.752.58.1.10", "value" => "magic");
// try to set both controls
if (!ldap_set_option($ds, LDAP_OPT_SERVER_CONTROLS, array($ctrl1, $ctrl2))) {
   echo "Failed to set server controls";
}
?>
```

Notes

Note

This function is only available when using OpenLDAP 2.x.x OR Netscape Directory SDK x.x.

See Also

- ldap_get_option()

ldap_set_rebind_proc

ldap_set_rebind_proc -- Set a callback function to do re-binds on referral chasing

Description

bool **ldap_set_rebind_proc** (resource `$link`, callback `$callback`)

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

ldap_sort

ldap_sort -- Sort LDAP result entries

Description

bool **ldap_sort** (resource `$link`, resource `$result`, string `$sortfilter`)

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

ldap_start_tls

ldap_start_tls -- Start TLS

Description

bool **ldap_start_tls** (resource `$link`)

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

ldap_t61_to_8859

ldap_t61_to_8859 -- Translate t61 characters to 8859 characters

Description

string **ldap_t61_to_8859** (string `$value`)

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

ldap_unbind

ldap_unbind -- Unbind from LDAP directory

Description

bool **ldap_unbind** (resource `$link_identifier`)

Unbinds from the LDAP directory.

Parameters

`link_identifier`
An LDAP link identifier, returned by ldap_connect().

Return Values

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

See Also

- ldap_bind()