

---

# **libldap Documentation**

*Release 0.1*

**Yves Legrandgerard**

**Nov 16, 2017**



---

## Contents

---

<b>1</b>	<b>libldap OpenLDAP C library interface module</b>	<b>3</b>
1.1	Functions . . . . .	3
1.1.1	Schema parsing functions . . . . .	4
1.2	Constants . . . . .	7
1.2.1	General . . . . .	7
1.2.2	Options . . . . .	8
1.3	Exceptions . . . . .	9
<b>2</b>	<b>LDAPObject classes</b>	<b>11</b>
<b>3</b>	<b>LDAPMod(s) classes</b>	<b>17</b>
<b>4</b>	<b>LDAPControl(s) classes</b>	<b>19</b>
<b>5</b>	<b>Indices and tables</b>	<b>21</b>
	<b>Python Module Index</b>	<b>23</b>



Contents:



---

## libldap OpenLDAP C library interface module

---

Module *libldap* is a Python3 wrapper for OpenLDAP (Lightweight Directory Access Protocol) C library.

### 1.1 Functions

Following functions are defined by module *libldap*:

`libldap.ldap_initialize(uri [, version=LDAP_VERSION3])`

Creates and initializes a new connection object (*LDAPObject*) to access a LDAP server and returns this object.

#### Parameters

- **uri** (*str*) – LDAP URI (Uniform Resource Identifier). It has the following form: *ldap[is]://host[:port][/dn]*. This parameter is identical to that of the underlying function `ldap_initialize()` of the library `openLDAP` except the optional *dn*. When this *dn* is provided, each parameter of a method of an instance of a *LDAPObject* which is a DN, is automatically completed by *dn* unless it already ends with *dn*. For example, in the code below:

```
>>> l = ldap_initialize('ldap://host.test/dc=example,dc=test')
>>> l.simple_bind_s(user='cn=admin', password='secret')
```

parameter *user* is rewritten to *'cn=admin,dc=example,dc=test'* before passed to the underlying OpenLDAP library C function

- **version** (*LDAP\_VERSION2* or *LDAP\_VERSION3*) – version of LDAP protocol

**Returns** a new *LDAPObject*

**Raises** *LDAPError*, *TypeError* or *ValueError*

**See also:**

`ldap_open(3)`

`libldap.ldap_get_option(option)`

This routine is used to retrieve global options. See *Options* for available options.

**Parameters** `option` (*int*) – global option to retrieve

**Returns** option value

**Return type** `int`

**Raises** `LDAPError`

For example, to get the peer certificate checking strategy:

```
>>> ldap_get_option(LDAP_OPT_PROTOCOL_VERSION)
2
```

**See also:**

`ldap_get_option(3)`

`libldap.ldap_set_option(option, optval)`

This routine permits to set global options. See *Options* for available options.

**Parameters**

- `option` (*int*) – global option to set
- `optval` (*int*) – option value

**Returns** `None`

**Raises** `LDAPError`

For example, to set the peer certificate checking strategy:

```
>>> ldap_set_option(LDAP_OPT_X_TLS_REQUIRE_CERT, LDAP_OPT_X_TLS_NEVER)
```

**See also:**

`ldap_set_option(3)`

`libldap.ldap_is_valid_dn(dn[, flags=LDAP_DN_FORMAT_LDAPV3])`

checks DN syntax

**Parameters**

- `dn` (*str*) – DN to check
- `flags` (`LDAP_DN_FORMAT_LDAPV3`, `LDAP_DN_FORMAT_LDAPV2` or `LDAP_DN_FORMAT_DCE`) – defines what DN syntax is expected (according to **RFC 4514**, **RFC 1779** and **DCE**, respectively). Parameter *flags* can also be ORed to the flag `LDAP_DN_PEDANTIC`

**Returns** `True` if DN is valid, `False` otherwise

**See also:**

`ldap_str2dn(3)`

### 1.1.1 Schema parsing functions

These functions are used to parse schema definitions in the syntax defined in RFC 4512 into Python dictionaries

`libldap.ldap_str2syntax(string[, flags])`



**Parameters**

- **string** (*str*) – the string to parse
- **flags** (*int*) – *flags* is a bit mask of parsing options controlling the relaxation of the syntax recognized. Default is `LDAP_SCHEMA_ALLOW_NONE`, see section *Flags* for more details

**Returns** {'oid': <str>, 'names': <list\_of\_strs>, 'desc': <str>|None, 'extensions': (<str>, <list\_of\_strs>)|None}

**Raises** `LDAPError`, `TypeError`

The returned value is a Python dictionary corresponding to the C-structure `LDAPSyntax` of the OpenLDAP library

**See also:**

`ldap_schema` (3)

`libldap.ldap_str2matchingrule` (*string* [, *flags* ])

**Parameters**

- **string** (*str*) – the string to parse
- **flags** (*int*) – *flags* is a bit mask of parsing options controlling the relaxation of the syntax recognized. Default is `LDAP_SCHEMA_ALLOW_NONE`, see section *Flags* for more details

**Returns** {'oid': <str>, 'names': <list\_of\_strs>, 'desc': <str>|None, 'obsolete': <bool>, 'syntax\_oid': <str>|None, 'extensions': (<str>, <list\_of\_strs>)|None}

**Raises** `LDAPError`, `TypeError`

The returned value is a Python dictionary corresponding to the C-structure `LDAPMatchingRule` of the OpenLDAP library

**See also:**

`ldap_schema` (3)

`libldap.ldap_str2matchingruleuse` (*string* [, *flags* ])

**Parameters**

- **string** (*str*) – the string to parse
- **flags** (*int*) – *flags* is a bit mask of parsing options controlling the relaxation of the syntax recognized. Default is `LDAP_SCHEMA_ALLOW_NONE`, see section *Flags* for more details

**Returns** {'oid': <str>, 'names': <list\_of\_strs>, 'desc': <str>|None, 'obsolete': <bool>, 'applies\_oids': <list\_of\_strs>, 'extensions': (<str>, <list\_of\_strs>)|None}

**Raises** `LDAPError`, `TypeError`

The returned value is a Python dictionary corresponding to the C-structure `LDAPMatchingRuleUse` of the OpenLDAP library

**See also:**

`ldap_schema` (3)

`libldap.ldap_str2attributetype` (*string* [, *flags* ])

**Parameters**

- **string** (*str*) – the string to parse

- **flags** (*int*) – *flags* is a bit mask of parsing options controlling the relaxation of the syntax recognized. Default is `LDAP_SCHEMA_ALLOW_NONE`, see section *Flags* for more details

**Returns** {'oid': <str>, 'names': <list\_of\_strs>, 'desc': <str>|None, 'obsolete': <bool>, 'sup\_oid': <str>|None, 'equality\_oid': <str>|None, 'ordering\_oid': <str>|None, 'substr\_oid': <str>|None, 'syntax\_oid': <str>|None, 'syntax\_len': <int>, 'single\_value': <bool>, 'collective': <bool>, 'no\_user\_mod': <bool>, 'usage': <int>, 'extensions': (<str>, <list\_of\_strs>)|None}

**Raises** `LDAPError`, `TypeError`

The returned value is a Python dictionary corresponding to the C-structure `LDAPAttributeType` of the OpenLDAP library. For possible values of the field *usage* see *Attribute types*

**See also:**

`ldap_schema(3)`

`libldap.ldap_str2objectclass(string[, flags])`

**Parameters**

- **string** (*str*) – the string to parse
- **flags** (*int*) – *flags* is a bit mask of parsing options controlling the relaxation of the objectclass recognized. Default is `LDAP_SCHEMA_ALLOW_NONE`, see section *Flags* for more details

**Returns** {'oid': <str>, 'names': <list\_of\_strs>, 'desc': <str>|None, 'obsolete': <bool>, 'sup\_oids': <list\_of\_strs>, 'kind': <int>, 'oids\_must': <list\_of\_strs>, 'oids\_may': <list\_of\_strs>, 'extensions': (<str>, <list\_of\_strs>)|None}

**Raises** `LDAPError`, `TypeError`

The returned value is a Python dictionary corresponding to the C-structure `LDAPObjectClass` of the OpenLDAP library. For possible values of the field *kind* see *Object classes*

**See also:**

`ldap_schema(3)`

## Examples

```
>>> ldap_str2syntax("( 1.3.6.1.4.1.1466.115.121.1.4 DESC 'Audio' X-NOT-HUMAN-READABLE
↳ 'TRUE' )")
{'extensions': [('X-NOT-HUMAN-READABLE', ['TRUE'])], 'oid': '1.3.6.1.4.1.1466.115.121.
↳ 1.4', 'desc': 'Audio', 'names': []}
>>> ldap_str2matchingrule("( 1.3.6.1.1.16.3 NAME 'UIDOrderingMatch' SYNTAX 1.3.6.1.1.
↳ 16.1 )")
{'names': ['UIDOrderingMatch'], 'desc': None, 'syntax_oid': '1.3.6.1.1.16.1', 'oid':
↳ '1.3.6.1.1.16.3', 'obsolete': False, 'extensions': None}
```

## 1.2 Constants

### 1.2.1 General

libldap.LDAP\_VERSION2

libldap.LDAP\_VERSION3

libldap.LDAP\_NO\_LIMIT

libldap.LDAP\_AUTH\_SIMPLE

### Modify constants

libldap.LDAP\_MOD\_ADD

libldap.LDAP\_MOD\_DELETE

libldap.LDAP\_MOD\_REPLACE

### Scope constants

libldap.LDAP\_SCOPE\_BASE

search the object itself

libldap.LDAP\_SCOPE\_ONELEVEL

search the object's immediate children

libldap.LDAP\_SCOPE\_SUBTREE

search the object and all its descendants

libldap.LDAP\_SCOPE\_CHILDREN

search all of the descendants

### Schema constants

#### See also:

*ldap\_schema(3)*

libldap.LDAP\_SCHEMA\_BASE

The base DN used to retrieve an LDAP server schema. It is usually the string: 'cn=Subschema'

### Flags

libldap.LDAP\_SCHEMA\_ALLOW\_NONE

Strict parsing according to RFC 4512

libldap.LDAP\_SCHEMA\_ALLOW\_NO\_OID

Permit definitions that do not contain an initial OID

libldap.LDAP\_SCHEMA\_ALLOW\_QUOTED

Permit quotes around some items that should not have them

libldap.LDAP\_SCHEMA\_ALLOW\_DESCR

Permit a descr instead of a numeric OID in places where the syntax expect the latter

libldap.**LDAP\_SCHEMA\_ALLOW\_DESCR\_PREFIX**  
permit that the initial numeric OID contains a prefix in descr format

libldap.**LDAP\_SCHEMA\_ALLOW\_ALL**  
Be very liberal, include all options

## Attribute types

libldap.**LDAP\_SCHEMA\_USER\_APPLICATIONS**  
The attribute type is non-operational

libldap.**LDAP\_SCHEMA\_DIRECTORY\_OPERATION**  
The attribute type is operational and is pertinent to the directory itself, i.e. it has the same value on all servers that master the entry containing this attribute type

libldap.**LDAP\_SCHEMA\_DISTRIBUTED\_OPERATION**  
The attribute type is operational and is pertinent to replication, shadowing or other distributed directory aspect

libldap.**LDAP\_SCHEMA\_DSA\_OPERATION**  
The attribute type is operational and is pertinent to the directory server itself, i.e. it may have different values for the same entry when retrieved from different servers that master the entry

## Object classes

libldap.**LDAP\_SCHEMA\_ABSTRACT**  
The object class is abstract, i.e. there cannot be entries of this class alone

libldap.**LDAP\_SCHEMA\_STRUCTURAL**  
The object class is structural, i.e. it describes the main role of the entry. On some servers, once the entry is created the set of structural object classes assigned cannot be changed: none of those present can be removed and none other can be added

libldap.**LDAP\_SCHEMA\_AUXILIARY**  
The object class is auxiliary, i.e. it is intended to go with other, structural, object classes. These can be added or removed at any time if attribute types are added or removed at the same time as needed by the set of object classes resulting from the operation

## DN Constants

libldap.**LDAP\_DN\_FORMAT\_LDAPV3**

libldap.**LDAP\_DN\_FORMAT\_LDAPV2**

libldap.**LDAP\_DN\_FORMAT\_DCE**

libldap.**LDAP\_DN\_PEDANTIC**  
does not allow extra spaces in the DN

### See also:

*ldap\_str2dn(3)*

## 1.2.2 Options

libldap.**LDAP\_OPT\_PROTOCOL\_VERSION**

## TLS options

libldap.LDAP\_OPT\_X\_TLS\_REQUIRE\_CERT

libldap.LDAP\_OPT\_X\_TLS\_NEVER

libldap.LDAP\_OPT\_X\_TLS\_HARD

libldap.LDAP\_OPT\_X\_TLS\_DEMAND

libldap.LDAP\_OPT\_X\_TLS\_ALLOW

libldap.LDAP\_OPT\_X\_TLS\_TRY

## 1.3 Exceptions

The module *libldap* defines only one exception:

**exception** libldap.LDAPError

This exception is in particular thrown when a call to a function of the OpenLDAP library fails. In this case, the error message associated with this exception is the string returned by `ldap_err2string()` (see *ldap\_error(3)* for more details)



---

## LDAPObject classes

---

### class `LDAPObject`

Instances of the class `LDAPObject` are created either by calling the function `ldap_initialize()`, or by calling the `LDAP` class constructor. The two methods are strictly equivalent. More precisely,

```
>>> l = ldap_initialize('ldap://host.test')
```

is equivalent to:

```
>>> l = LDAP('ldap://host.test')
```

The connection is automatically unbound and closed when the LDAP object is deleted.

### class `LDAP(uri[, version=LDAP_VERSION3])`

An instance of the class `LDAPObject` has the following attributes:

#### **uri**

LDAP URI (Uniform Resource Identifier): `ldap[is]://host[:port]`

#### **scheme**

URI scheme: `ldap`, `ldapi` or `ldaps`

#### **host**

LDAP host to contact

#### **ip**

IPv4/v6 address of LDAP host to contact

#### **port**

port on host (usually 389 or 636)

#### **dn**

To learn how `dn` attribute is used, refer to the documentation of the function `ldap_initialize()`. This attribute can be modified at any time:

```
>>> l = ldap_initialize('ldap://host.test')
>>> print(l.dn)
```

```
None
>>> l.dn = 'dc=example,dc=test'
>>> print(l.dn)
dc=example,dc=test
>>> l.dn = None
```

Methods of the class *LDAPObject* are:

**simple\_bind\_s** (*[user, password]*)

Just after an *LDAPObject* is created, it must be bound. If parameters *user* and *password* are not present, an *anonymous* bind is done

**Parameters**

- **user** (*str*) – DN to bind as
- **password** (*str*) – userPassword associated with the entry

**Returns** None

**Raises** LDAPError

**See also:**

*ldap\_simple\_bind\_s(3)*

**bind\_s** (*[user, password, method=LDAP\_AUTH\_SIMPLE]*)

Identical to method *simple\_bind\_s()* except for the extra *method* parameter selecting the authentication method to use. Only method LDAP\_AUTH\_SIMPLE is currently available

**Parameters**

- **user** (*str*) – DN to bind as
- **password** (*str*) – userPassword associated with the entry
- **method** (*int*) – authentication method to use

**Returns** None

**Raises** LDAPError

**See also:**

*ldap\_bind\_s(3)*

**unbind\_s** ()

Unbind from the directory, terminate the current association, and free the resources previously allocated. Further invocation of methods on the object will yield exception LDAPError

**Returns** None

**Raises** LDAPError

**See also:**

*ldap\_unbind\_s(3)*

**start\_tls\_s** ()

Initiates TLS processing on an LDAP session

**Returns** None

**Raises** LDAPError



**See also:**`ldap_start_tls_s(3)`**get\_option** (*option*)

This routine is used to retrieve options from an *LDAPObject*. See *Options* for available options.

**Parameters** *option* (*int*) – global option to retrieve

**Returns** option value

**Return type** int

**Raises** LDAPError

**See also:**`ldap_get_option(3)`**set\_option** (*option*, *optval*)

This routine permits to set options for an *LDAPObject*. See *Options* for available options.

**Parameters**

- **option** (*int*) – option to set
- **optval** (*int*) – option value

**Returns** None

**Raises** LDAPError

**See also:**`ldap_set_option(3)`**add\_ext\_s** (*dn*, *mods*[, *serverctrls*[, *clientctrls* ] ])

Performs an LDAP add operation

**Parameters**

- **dn** (*str*) – the DN of the entry to add
- **mods** – a list of *LDAPMod* objects. Attribute mode of each *LDAPMod* object must be `LDAP_MOD_ADD`
- **serverctrls** (*LDAPControls*) – specifies server control(s). See section *Control methods*
- **clientctrls** (*LDAPControls*) – specifies client control(s). See section *Control methods*

**Returns** None

**Raises** LDAPError, TypeError

**See also:**`ldap_add_ext_s(3)`**delete\_ext\_s** (*dn*[, *serverctrls*[, *clientctrls* ] ])

Performs an LDAP delete operation

**Parameters**

- **dn** (*str*) – the DN of the entry to be deleted
- **serverctrls** (*LDAPControls*) – specifies server control(s). See section *Control methods*

- **clientctrls** (*LDAPControls*) – specifies client control(s). See section *Control methods*

**Returns** None

**Raises** LDAPError

**See also:**

*ldap\_delete\_ext\_s(3)*

**modify\_ext\_s** (*dn*, *mods*[, *serverctrls*[, *clientctrls*]])

Performs an LDAP modify operation

**Parameters**

- **dn** (*str*) – the DN of the entry to modify
- **mods** – a list of *LDAPMod* objects. All modifications are performed in the order in which they are listed
- **serverctrls** (*LDAPControls*) – specifies server control(s). See section *Control methods*
- **clientctrls** (*LDAPControls*) – specifies client control(s). See section *Control methods*

**Returns** None

**Raises** LDAPError, TypeError

```
>>> l = ldap_initialize('ldap://host.test/dc=example,dc=test')
>>> l.start_tls_s()
>>> l.simple_bind_s(user='cn=admin', password='secret')
>>> lma = LDAPMod(LDAP_MOD_ADD, 'mailalias', ['bob@example.test'])
>>> lmr = LDAPMod(LDAP_MOD_REPLACE, 'givenName', ['Robert'])
>>> l.modify_ext_s('uid=bob,ou=users', [lma, lmr])
```

**See also:**

*ldap\_modify\_ext\_s(3)*

**search\_ext\_s** ([*base*[, *scope*[, *filter*[, *attrs*[, *attrsonly*[, *serverctrls*[, *clientctrls*[, *limit*[, *timeout*]]]]]]]])

Performs a LDAP search operation

**Parameters**

- **base** (*str*) – DN of the entry at which to start the search. If parameter *base* is not present, attribute *dn* is used if it's not None otherwise exception *LDAPError* is raised
- **scope** (*int*) – scope of the search. Default is *LDAP\_SCOPE\_SUBTREE* (search the object and all its descendants). For other possible values, see *scope constants*
- **filter** (*str*) – filter to apply in the search. Default is '(objectClass=\*)'
- **attrs** (*list of str(s)*) – a list of attribute descriptions to return from matching entries. If parameter *attrs* is not present, all attributes are returned
- **attrsonly** (*bool*) – if True, only attribute descriptions are returned (attribute values are then empty lists). Default is False
- **serverctrls** (*LDAPControls*) – specifies server control(s). See section *Control methods*

- **clientctrls** (*LDAPControls*) – specifies client control(s). See section *Control methods*
- **limit** (*int*) – size limit of the answer. Default is LDAP\_NO\_LIMIT
- **timeout** (*int*) – timeout in seconds to wait server answer. 0 means no timeout, this is the default

**Returns** a (possibly empty) list of results of the form: *[(dn, entry), ...]*. Each item of the list is 2-tuple where *dn* is a string containing the DN of the entry, and *entry* is a dictionary containing the attributes associated with the entry: *{attr: [value, ...], ...}*. For each entry in the dictionary, the key *attr* (string) is the attribute description and the corresponding value is the list of the associated values (strings)

**Raises** LDAPError, TypeError

A simple example:

```
>>> l = ldap_initialize('ldap://host.test/dc=example,dc=test')
>>> l.start_tls_s()
>>> l.simple_bind_s()
>>> l.search_ext_s(attrs=['uid'])
[(('uid=alice',ou=users,dc=example,dc=test', {'uid': ['alice']}), ('uid=bob,
↪ou=users,dc=example,dc=test', {'uid': ['bob']}))]
```

**See also:**

*ldap\_search\_ext\_s(3)*

**get\_schema()**

retrieves LDAP schema from server

**Returns** [(‘cn=Subschema’, entry)]

**Raises** LDAPError, TypeError

More precisely, this function first executes the following statement:

```
>>> schema = self.search_ext_s(LDAP_SCHEMA_BASE, scope=LDAP_SCOPE_BASE,
↪attrs=['+'])
```

The variable *schema* has the following form: *[(‘cn=Subschema’, entry)]*. The function *get\_schema()*, before returning *schema*, performs the following treatment: fields *ldapSyntaxes*, *matchingRules*, *matchingRuleUse*, *attributeTypes* and *objectClasses* of dictionary *entry* are respectively parsed with *ldap\_str2syntax()*, *ldap\_str2matchingrule()*, *ldap\_str2matchingruleuse()*, *ldap\_str2attributetype()* and *ldap\_str2objectclass()*. See section *Schema parsing functions* for more details

**modrdn2\_s(dn, newrdn[, deleteoldrdn=False])**  
performs an LDAP modify RDN operation

**Parameters**

- **dn** (*str*) – the DN of the entry whose RDN is to be changed
- **newrdn** (*str*) – the new RDN
- **deleteoldrdn** (*bool*) – if True, the old RDN values are deleted from the entry

**Returns** None

**Raises** LDAPError

See also:

`ldap_modrdn2_s(3)`

## Control methods

`create_sort_control(keylist[, iscritical=False])`

builds a sort control

### Parameters

- **keylist** (*str*) – sort string. For example, if *keylist* is `'sn -givenName'` the search results are sorted first by surname and then by given name, with the given name being sorted in reverse (descending order) as specified by the prefixed minus sign (-)
- **iscritical** (*bool*) – the *iscritical* parameter is `True` non-zero for a critical control, `False` otherwise. Default is `False`

**Returns** a new `LDAPControl` object

**Raises** `LDAPError`, `TypeError`

`create_assertion_control(filter[, iscritical=False])`

builds an assertion control

### Parameters

- **filter** (*str*) – control value (LDAP filter). See [RFC 4528](#)
- **iscritical** (*bool*) – the *iscritical* parameter is `True` non-zero for a critical control, `False` otherwise. Default is `False`

**Returns** a new `LDAPControl` object

**Raises** `LDAPError`, `TypeError`

---

## LDAPMod(s) classes

---

**class** `LDAPMod` (*mode*, *attr*[, *values=None* ])

This class is just a Python wrapper for the corresponding C structure `LDAPMod` described in *ldap\_modify\_ext\_s(3)*

**Parameters**

- **mode** (`LDAP_MOD_ADD`, `LDAP_MOD_DELETE` or `LDAP_MOD_REPLACE`) – type of modification to perform
- **attr** (*str*) – the attribute to modify
- **values** – a list of values (strings) to add, delete, or replace respectively or `None` if the the entire attribute is to be deleted when parameter *mode* is `LDAP_MOD_DELETE`

**Returns** a new `LDAPMod` object

**Raises** `TypeError`, `ValueError`

An instance of the class `LDAPMod` has the following attributes:

**mode**

type of modification to perform (`LDAP_MOD_ADD`, `LDAP_MOD_DELETE` or `LDAP_MOD_REPLACE`)

**attr**

attribute to modify

**values**

a list of values to add, delete, or replace respectively or `None`

Some examples:

```
>>> lma = LDAPMod(LDAP_MOD_ADD, 'uid', ['bob'])
>>> lmd = LDAPMod(LDAP_MOD_DELETE, 'uid')
```

**See also:**

*ldap\_add\_ext\_s(3)*, *ldap\_modify\_ext\_s(3)*

**class LDAPMods** (*mode*, *\*\*attrs*)

This class is just a utility for regrouping classes *LDAPMod* with the same mode. It is a subclass of Python class *list*

**Parameters**

- **mode** (LDAP\_MOD\_ADD, LDAP\_MOD\_DELETE or LDAP\_MOD\_REPLACE) – type of modification to perform
- **attrs** (*dict*) – attributes to modify

**Returns** a list of *LDAPMod* objects

**Raises** LDAPError, TypeError or ValueError

So instead of writing,

```
>>> lma = LDAPMod(LDAP_MOD_ADD, 'uid', ['bob'])
>>> lmb = LDAPMod(LDAP_MOD_ADD, 'givenName', ['Robert'])
>>> l.add_ext_s('ou=users', [lma, lmb])
```

it is often shorter to write:

```
>>> lm = LDAPMods(LDAP_MOD_ADD, uid=['bob'], givenName=['Robert'])
>>> l.add_ext_s('ou=users', lm)
```

or

```
>>> d = {'uid': ['bob'], 'givenName': ['Robert']}
>>> l.add_ext_s('ou=users', LDAPMods(LDAP_MOD_ADD, **d))
```

---

## LDAPControl(s) classes

---

**class LDAPControl**

This class is just a Python wrapper for the corresponding C structure `LDAPControl` (see `ldap_controls(3)`)

**Warning:** `LDAPControl` object cannot be created directly. You have to use instead `create_*_control()` methods of a `LDAP` object instance

**class LDAPControls** (`<LDAPControl>`[, `<LDAPControl>` ... ])

Classes `LDAPControls` are the type of parameters `serverctrls` and `clientctrls` used to specify server and client controls respectively in `*_ext[_s]()` methods of `LDAP` objects. For example:

```
>>> ca = l.create_assertion_control('ou=users', True)
>>> cs = l.create_sort_control('sn -givenName')
>>> ctrls = LDAPControls(ca, cs)
>>> l.search_ext_s(serverctrls=ctrls)
```

**See also:**

`ldap_controls(3)`





## CHAPTER 5

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`



|

libldap (*Posix*), 3



**A**

add\_ext\_s() (LDAP method), 13  
 attr (LDAPMod attribute), 17

**B**

bind\_s() (LDAP method), 12

**C**

create\_assertion\_control() (LDAP method), 16  
 create\_sort\_control() (LDAP method), 16

**D**

delete\_ext\_s() (LDAP method), 13  
 dn (LDAP attribute), 11

**G**

get\_option() (LDAP method), 13  
 get\_schema() (LDAP method), 15

**H**

host (LDAP attribute), 11

**I**

ip (LDAP attribute), 11

**L**

LDAP (built-in class), 11  
 LDAP\_AUTH\_SIMPLE (in module libldap), 7  
 LDAP\_DN\_FORMAT\_DCE (in module libldap), 8  
 LDAP\_DN\_FORMAT\_LDAPV2 (in module libldap), 8  
 LDAP\_DN\_FORMAT\_LDAPV3 (in module libldap), 8  
 LDAP\_DN\_PEDANTIC (in module libldap), 8  
 ldap\_get\_option() (in module libldap), 3  
 ldap\_initialize() (in module libldap), 3  
 ldap\_is\_valid\_dn() (in module libldap), 4  
 LDAP\_MOD\_ADD (in module libldap), 7  
 LDAP\_MOD\_DELETE (in module libldap), 7  
 LDAP\_MOD\_REPLACE (in module libldap), 7

LDAP\_NO\_LIMIT (in module libldap), 7  
 LDAP\_OPT\_PROTOCOL\_VERSION (in module libldap), 8  
 LDAP\_OPT\_X\_TLS\_ALLOW (in module libldap), 9  
 LDAP\_OPT\_X\_TLS\_DEMAND (in module libldap), 9  
 LDAP\_OPT\_X\_TLS\_HARD (in module libldap), 9  
 LDAP\_OPT\_X\_TLS\_NEVER (in module libldap), 9  
 LDAP\_OPT\_X\_TLS\_REQUIRE\_CERT (in module libldap), 9  
 LDAP\_OPT\_X\_TLS\_TRY (in module libldap), 9  
 LDAP\_SCHEMA\_ABSTRACT (in module libldap), 8  
 LDAP\_SCHEMA\_ALLOW\_ALL (in module libldap), 8  
 LDAP\_SCHEMA\_ALLOW\_DESCR (in module libldap), 7  
 LDAP\_SCHEMA\_ALLOW\_DESCR\_PREFIX (in module libldap), 7  
 LDAP\_SCHEMA\_ALLOW\_NO\_OID (in module libldap), 7  
 LDAP\_SCHEMA\_ALLOW\_NONE (in module libldap), 7  
 LDAP\_SCHEMA\_ALLOW\_QUOTED (in module libldap), 7  
 LDAP\_SCHEMA\_AUXILIARY (in module libldap), 8  
 LDAP\_SCHEMA\_BASE (in module libldap), 7  
 LDAP\_SCHEMA\_DIRECTORY\_OPERATION (in module libldap), 8  
 LDAP\_SCHEMA\_DISTRIBUTED\_OPERATION (in module libldap), 8  
 LDAP\_SCHEMA\_DSA\_OPERATION (in module libldap), 8  
 LDAP\_SCHEMA\_STRUCTURAL (in module libldap), 8  
 LDAP\_SCHEMA\_USER\_APPLICATIONS (in module libldap), 8  
 LDAP\_SCOPE\_BASE (in module libldap), 7  
 LDAP\_SCOPE\_CHILDREN (in module libldap), 7  
 LDAP\_SCOPE\_ONELEVEL (in module libldap), 7  
 LDAP\_SCOPE\_SUBTREE (in module libldap), 7  
 ldap\_set\_option() (in module libldap), 4  
 ldap\_str2attributetype() (in module libldap), 5

ldap\_str2matchingrule() (in module libldap), 5  
ldap\_str2matchingruleuse() (in module libldap), 5  
ldap\_str2objectclass() (in module libldap), 6  
ldap\_str2syntax() (in module libldap), 4  
LDAP\_VERSION2 (in module libldap), 7  
LDAP\_VERSION3 (in module libldap), 7  
LDAPControl (built-in class), 19  
LDAPControls (built-in class), 19  
LDAPError, 9  
LDAPMod (built-in class), 17  
LDAPMods (built-in class), 17  
LDAPObject (built-in class), 11  
libldap (module), 3

## M

mode (LDAPMod attribute), 17  
modify\_ext\_s() (LDAP method), 14  
modrdn2\_s() (LDAP method), 15

## P

port (LDAP attribute), 11

## R

RFC  
    RFC 1779, 4  
    RFC 4514, 4  
    RFC 4528, 16

## S

scheme (LDAP attribute), 11  
search\_ext\_s() (LDAP method), 14  
set\_option() (LDAP method), 13  
simple\_bind\_s() (LDAP method), 12  
start\_tls\_s() (LDAP method), 12

## U

unbind\_s() (LDAP method), 12  
uri (LDAP attribute), 11

## V

values (LDAPMod attribute), 17