libldap Documentation

Release 1.2

Yves Legrandgerard

Nov 16, 2017

Contents

Contents:

CHAPTER 1

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 retreive global options. See *Options* for available options.

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

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 'UUIDOrderingMatch' SYNTAX 1.3.6.1.1.
↪16.1 )")
{'names': ['UUIDOrderingMatch'], '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

SASL constants

libldap.**LDAP_SASL_AUTOMATIC**
use defaults if available, prompt otherwise

libldap.**LDAP_SASL_INTERACTIVE**
always prompt

libldap.**LDAP_SASL_QUIET**
never prompt

libldap.**LDAP_SASL_SIMPLE**
select simple authentication

Schema constants

See also:

ldap_schema(3)

libldap.**LDAP_SCHEMA_BASE**
The base DN used to retreive 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**

SASL options

libldap.**LDAP_OPT_X_SASL_MECH**
to get the SASL mechanism

libldap.**LDAP_OPT_X_SASL_MECHLIST**
to get the list of the available SASL mechanisms. For example:

```
>>> ldap_get_option(LDAP_OPT_X_SASL_MECHLIST)
('ANONYMOUS', 'LOGIN', 'PLAIN', 'CRAM-MD5', 'NTLM', 'EXTERNAL', 'DIGEST-MD5')
```

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)

CHAPTER 2

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 stricly 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)

sasl_bind_s ([*mech*[, *dn*[, *password*]]])

Performs a SASL bind

Parameters

- **mech** (*str*) – a SASL mechanism. For example: 'DIGEST-MD5', 'GSSAPI',... Default is LDAP_SASL_SIMPLE
- **dn** (*str*) – the DN to bind as. If not provided, *sasl_bind_s()* will prompt for it.
- **password** (*str*) – the password associated to entry *dn*. If not provided, *sasl_bind_s()* will prompt for it.

Returns None

Raises LDAPError, TypeError

```
>>> l = ldap_initialize('ldap://host.test')
>>> l.start_tls_s()
>>> l.sasl_bind_s(dn='uid=testsasl,ou=users,dc=example,dc=test')
```

```
Enter password:
>>>
```

See also:

ldap_sasl_bind_s(3)

sasl_interactive_bind_s([*mechs*[, *flags*[, *user*[, *password*]]]])
Performs a (interactive) SASL bind

Parameters

- **mechs** – a list or a tuple of candidate mechanisms to use. For example: `('LOGIN', 'PLAIN', 'DIGEST-MD5')`
- **flags** (*int*) – controls the interaction used to retrieve any necessary SASL authentication parameters. Default `flags` is `LDAP_SASL_INTERACTIVE` if `user` or `password` is not provided and `LDAP_SASL_QUIET` otherwise. See *SASL constants* for available flags
- **user** (*str*) – the user to authenticate. If not provided, *sasl_interactive_bind_s()* will prompt for it.
- **password** (*str*) – the password for the provided user. If not given, *sasl_interactive_bind_s()* will prompt for it.

Returns `None`

Raises `LDAPError`, `TypeError`

```
>>> l = ldap_initialize('ldap://host.test')
>>> l.start_tls_s()
>>> l.sasl_interactive_bind_s(user='testsasl')
SASL/DIGEST-MD5 authentication started
Enter user's password:
SASL username: testsasl
SASL SSF: 128
SASL data security layer installed.
>>>
```

Another example:

```
>>> l.sasl_interactive_bind_s(mechs=('DIGEST-MD5',), flags=LDAP_SASL_QUIET,␣
↪user='testsasl')
Enter user's password:
>>>
```

See also:

ldap_sasl_interactive_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 retreive options from an *LDAPObject*. See *Options* for available options.

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

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()
retreives 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`

CHAPTER 3

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))
```

CHAPTER 4

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

Python Module Index

l

Index

M

P

R

S

U

V