NAME

ldap_str2syntax, ldap_syntax2str, ldap_syntax2name, ldap_syntax_free, ldap_str2matchingrule, ldap_matchingrule2str, ldap_matchingrule2name, ldap_matchingrule_free, ldap_str2attributetype, ldap_attributetype2str, ldap_attributetype2name, ldap_attributetype_free, ldap_str2objectclass, ldap_objectclass2str, ldap_objectclass2name, ldap_objectclass_free, ldap_scherr2str - Schema definition handling routines

LIBRARY

OpenLDAP LDAP (libldap, -lldap)

SYNOPSIS

#include <ldap.h>
#include <ldap_schema.h>

LDAPSyntax * ldap_str2syntax(s, code, errp, flags)
const char * s;
int * code;
const char ** errp;
const int flags;

char * ldap_syntax2str(syn)
const LDAPSyntax * syn;

const char * ldap_syntax2name(syn)
LDAPSyntax * syn;

ldap_syntax_free(syn)
LDAPSyntax * syn;

LDAPMatchingRule * ldap_str2matchingrule(s, code, errp, flags)
const char * s;
int * code;
const char ** errp;
const int flags;

char * ldap_matchingrule2str(mr);
const LDAPMatchingRule * mr;

const char * ldap_matchingrule2name(mr)
LDAPMatchingRule * mr;

ldap_matchingrule_free(mr)
LDAPMatchingRule * mr;

LDAPAttributeType * ldap_str2attributetype(s, code, errp, flags)
const char * s;
int * code;
const char ** errp;
const int flags;

char * ldap_attributetype2str(at)
const LDAPAttributeType * at;

const char * ldap_attributetype2name(at)
LDAPAttributeType * at;

ldap_attributetype_free(at)
LDAPAttributeType * at;

LDAPObjectClass * ldap_str2objectclass(s, code, errp, flags)
const char * s;
int * code;
const char ** errp;
const int flags;

char * ldap_objectclass2str(oc)
const LDAPObjectClass * oc;

const char * ldap_objectclass2name(oc)
LDAPObjectClass * oc;

ldap_objectclass_free(oc)
LDAPObjectClass * oc;

char * ldap_scherr2str(code)
int code;

DESCRIPTION

These routines are used to parse schema definitions in the syntax defined in RFC 4512 into structs and handle these structs. These routines handle four kinds of definitions: syntaxes, matching rules, attribute types and object classes. For each definition kind, four routines are provided.

ldap_str2xxx() takes a definition in RFC 4512 format in argument *s* as a NUL-terminated string and returns, if possible, a pointer to a newly allocated struct of the appropriate kind. The caller is responsible for freeing the struct by calling **ldap_xxx_free()** when not needed any longer. The routine returns NULL if some problem happened. In this case, the integer pointed at by argument *code* will receive an error code (see below the description of **ldap_scherr2str()** for an explanation of the values) and a pointer to a NUL-terminated string will be placed where requested by argument *errp* , indicating where in argument *s* the error happened, so it must not be freed by the caller. Argument *flags* is a bit mask of parsing options controlling the relaxation of the syntax recognized. The following values are defined:

LDAP_SCHEMA_ALLOW_NONE
strict parsing according to RFC 4512.

LDAP_SCHEMA_ALLOW_NO_OID
permit definitions that do not contain an initial OID.

LDAP_SCHEMA_ALLOW_QUOTED
permit quotes around some items that should not have them.

LDAP_SCHEMA_ALLOW_DESCR
permit a **descr** instead of a numeric OID in places where the syntax expect the latter.

LDAP_SCHEMA_ALLOW_DESCR_PREFIX
permit that the initial numeric OID contains a prefix in **descr** format.

LDAP_SCHEMA_ALLOW_ALL
be very liberal, include all options.

The structures returned are as follows:

```
typedef struct ldap_schema_extension_item {
        char *lsei_name;              /* Extension name */
        char **lsei_values;           /* Extension values */
} LDAPSchemaExtensionItem;

typedef struct ldap_syntax {
        char *syn_oid;                /* OID */
        char **syn_names;             /* Names */
        char *syn_desc;               /* Description */
        LDAPSchemaExtensionItem **syn_extensions; /* Extension */
} LDAPSyntax;

typedef struct ldap_matchingrule {
        char *mr_oid;              /* OID */
        char **mr_names;           /* Names */
        char *mr_desc;             /* Description */
        int  mr_obsolete;          /* Is obsolete? */
        char *mr_syntax_oid;       /* Syntax of asserted values */
        LDAPSchemaExtensionItem **mr_extensions; /* Extensions */
} LDAPMatchingRule;

typedef struct ldap_attributetype {
        char *at_oid;              /* OID */
        char **at_names;           /* Names */
        char *at_desc;             /* Description */
        int  at_obsolete;          /* Is obsolete? */
        char *at_sup_oid;          /* OID of superior type */
        char *at_equality_oid;     /* OID of equality matching rule */
        char *at_ordering_oid;     /* OID of ordering matching rule */
        char *at_substr_oid;       /* OID of substrings matching rule */
        char *at_syntax_oid;       /* OID of syntax of values */
        int  at_syntax_len;        /* Suggested minimum maximum length */
        int  at_single_value;      /* Is single-valued?  */
        int  at_collective;        /* Is collective? */
        int  at_no_user_mod;       /* Are changes forbidden through LDAP? */
        int  at_usage;             /* Usage, see below */
        LDAPSchemaExtensionItem **at_extensions; /* Extensions */
} LDAPAttributeType;

typedef struct ldap_objectclass {
        char *oc_oid;              /* OID */
        char **oc_names;           /* Names */
        char *oc_desc;             /* Description */
        int  oc_obsolete;          /* Is obsolete? */
        char **oc_sup_oids;        /* OIDs of superior classes */
        int  oc_kind;              /* Kind, see below */
        char **oc_at_oids_must;    /* OIDs of required attribute types */
        char **oc_at_oids_may;     /* OIDs of optional attribute types */
        LDAPSchemaExtensionItem **oc_extensions; /* Extensions */
} LDAPObjectClass;
```

Some integer fields (those described with a question mark) have a truth value, for these fields the possible values are:

LDAP_SCHEMA_NO
The answer to the question is no.

LDAP_SCHEMA_YES
The answer to the question is yes.

For attribute types, the following usages are possible:

LDAP_SCHEMA_USER_APPLICATIONS
the attribute type is non-operational.

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 provide the entry containing this attribute type.

LDAP_SCHEMA_DISTRIBUTED_OPERATION
the attribute type is operational and is pertinent to replication, shadowing or other distributed directory aspect. TBC.

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 provide the entry.

Object classes can be of three kinds:

LDAP_SCHEMA_ABSTRACT
the object class is abstract, i.e. there cannot be entries of this class alone.

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.

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.

Routines **ldap_xxx2name()** return a canonical name for the definition.

Routines **ldap_xxx2str()** return a string representation in the format described by RFC 4512 of the struct passed in the argument. The string is a newly allocated string that must be freed by the caller. These routines may return NULL if no memory can be allocated for the string.

ldap_scherr2str() returns a NUL-terminated string with a text description of the error found. This is a pointer to a static area, so it must not be freed by the caller. The argument *code* comes from one of the parsing routines and can adopt the following values:

LDAP_SCHERR_OUTOFMEM
Out of memory.

LDAP_SCHERR_UNEXPTOKEN
Unexpected token.

LDAP_SCHERR_NOLEFTPAREN
Missing opening parenthesis.

LDAP_SCHERR_NORIGHTPAREN
Missing closing parenthesis.

LDAP_SCHERR_NODIGIT
Expecting digit.

LDAP_SCHERR_BADNAME
Expecting a name.

LDAP_SCHERR_BADDESC
Bad description.

LDAP_SCHERR_BADSUP
Bad superiors.

LDAP_SCHERR_DUPOPT
Duplicate option.

LDAP_SCHERR_EMPTY
Unexpected end of data.

SEE ALSO

ldap(3)

ACKNOWLEDGEMENTS

OpenLDAP Software is developed and maintained by The OpenLDAP Project <http://www.openldap.org/>. **OpenLDAP Software** is derived from the University of Michigan LDAP 3.3 Release.