NAME

ldap_sync_init, ldap_sync_init_refresh_only, ldap_sync_init_refresh_and_persist, ldap_sync_poll - LDAP sync routines

LIBRARY

OpenLDAP LDAP (libldap, -lldap)

SYNOPSIS

#include <ldap.h>

int ldap_sync_init(ldap_sync_t ****ls*, int** *mode***);**

int ldap_sync_init_refresh_only(ldap_sync_t ****ls*);**

int ldap_sync_init_refresh_and_persist(ldap_sync_t ****ls*);**

int ldap_sync_poll(ldap_sync_t ****ls*);**

ldap_sync_t * ldap_sync_initialize(ldap_sync_t ****ls*);**

void ldap_sync_destroy(ldap_sync_t ****ls*, int** *freeit***);**

typedef int (**ldap_sync_search_entry_f***)(ldap_sync_t ****ls***,**
LDAPMessage ****msg*, struct berval ****entryUUID***,**
ldap_sync_refresh_t *phase***);**

typedef int (**ldap_sync_search_reference_f***)(ldap_sync_t ****ls***,**
LDAPMessage ****msg*);**

typedef int (**ldap_sync_intermediate_f***)(ldap_sync_t ****ls***,**
LDAPMessage ****msg*, BerVarray** *syncUUIDs***,**
ldap_sync_refresh_t *phase***);**

typedef int (**ldap_sync_search_result_f***)(ldap_sync_t ****ls***,**
LDAPMessage ****msg*, int** *refreshDeletes***);**

DESCRIPTION

These routines provide an interface to the LDAP Content Synchronization operation (RFC 4533). They require an **ldap_sync_t** structure to be set up with parameters required for various phases of the operation; this includes setting some handlers for special events. All handlers take a pointer to the

ldap_sync_t structure as the first argument, and a pointer to the **LDAPMessage** structure as received from the server by the client library, plus, occasionally, other specific arguments.

The members of the **ldap_sync_t** structure are:

**char ****ls_base*
The search base; by default, the **BASE** option in **ldap.conf**(5).

int *ls_scope*
The search scope (one of **LDAP_SCOPE_BASE**, **LDAP_SCOPE_ONELEVEL**, **LDAP_SCOPE_SUBORDINATE** or **LDAP_SCOPE_SUBTREE**; see **ldap.h** for details).

**char ****ls_filter*
The filter (RFC 4515); by default, **(objectClass=*)**.

**char *****ls_attrs*
The requested attributes; by default **NULL**, indicating all user attributes.

int *ls_timelimit*
The requested time limit (in seconds); by default **0**, to indicate no limit.

int *ls_sizelimit*
The requested size limit (in entries); by default **0**, to indicate no limit.

int *ls_timeout*
The desired timeout during polling with **ldap_sync_poll**(3). A value of **-1** means that polling is blocking, so **ldap_sync_poll**(3) will not return until a message is received; a value of **0** means that polling returns immediately, no matter if any response is available or not; a positive value represents the timeout the **ldap_sync_poll**(3) function will wait for response before returning, unless a message is received; in that case, **ldap_sync_poll**(3) returns as soon as the message is available.

ldap_sync_search_entry_f *ls_search_entry*
A function that is called whenever an entry is returned. The **msg** argument is the **LDAPMessage** that contains the searchResultEntry; it can be parsed using the regular client API routines, like **ldap_get_dn**(3), **ldap_first_attribute**(3), and so on. The **entryUUID** argument contains the entryUUID of the entry. The **phase** argument indicates the type of operation: one of **LDAP_SYNC_CAPI_PRESENT**, **LDAP_SYNC_CAPI_ADD**, **LDAP_SYNC_CAPI_MODIFY**, **LDAP_SYNC_CAPI_DELETE**; in case of **LDAP_SYNC_CAPI_PRESENT** or **LDAP_SYNC_CAPI_DELETE**, only the DN is contained in the *LDAPMessage*; in case of

LDAP_SYNC_CAPI_MODIFY, the whole entry is contained in the *LDAPMessage*, and the application is responsible of determining the differences between the new view of the entry provided by the caller and the data already known.

ldap_sync_search_reference_f *ls_search_reference*

A function that is called whenever a search reference is returned. The **msg** argument is the **LDAPMessage** that contains the searchResultReference; it can be parsed using the regular client API routines, like **ldap_parse_reference**(3).

ldap_sync_intermediate_f *ls_intermediate*

A function that is called whenever something relevant occurs during the refresh phase of the search, which is marked by an *intermediateResponse* message type. The **msg** argument is the **LDAPMessage** that contains the intermediate response; it can be parsed using the regular client API routines, like **ldap_parse_intermediate**(3). The **syncUUIDs** argument contains an array of UUIDs of the entries that depends on the value of the **phase** argument. In case of **LDAP_SYNC_CAPI_PRESENTS**, the "present" phase is being entered; this means that the following sequence of results will consist in entries in "present" sync state. In case of **LDAP_SYNC_CAPI_DELETES**, the "deletes" phase is being entered; this means that the following sequence of results will consist in entries in "delete" sync state. In case of **LDAP_SYNC_CAPI_PRESENTS_IDSET**, the message contains a set of UUIDs of entries that are present; it replaces a "presents" phase. In case of **LDAP_SYNC_CAPI_DELETES_IDSET**, the message contains a set of UUIDs of entries that have been deleted; it replaces a "deletes" phase. In case of **LDAP_SYNC_CAPI_DONE,** a "presents" phase with "refreshDone" set to "TRUE" has been returned to indicate that the refresh phase of refreshAndPersist is over, and the client should start polling. Except for the **LDAP_SYNC_CAPI_PRESENTS_IDSET** and **LDAP_SYNC_CAPI_DELETES_IDSET** cases, **syncUUIDs** is NULL.

ldap_sync_search_result_f *ls_search_result*

A function that is called whenever a searchResultDone is returned. In refreshAndPersist this can only occur when the server decides that the search must be interrupted. The **msg** argument is the **LDAPMessage** that contains the response; it can be parsed using the regular client API routines, like **ldap_parse_result**(3). The **refreshDeletes** argument is not relevant in this case; it should always be -1.

**void ***ls_private*

A pointer to private data. The client may register here a pointer to data the handlers above may need.

**LDAP ***ls_ld*

A pointer to a LDAP structure that is used to connect to the server. It is the responsibility of the

client to initialize the structure and to provide appropriate authentication and security in place.

GENERAL USE

A **ldap_sync_t** structure is initialized by calling **ldap_sync_initialize(3).** This simply clears out the contents of an already existing **ldap_sync_t** structure, and sets appropriate values for some members. After that, the caller is responsible for setting up the connection (member **ls_ld**), eventually setting up transport security (TLS), for binding and any other initialization. The caller must also fill all the documented search-related fields of the **ldap_sync_t** structure.

At the end of a session, the structure can be cleaned up by calling **ldap_sync_destroy**(3), which takes care of freeing all data assuming it was allocated by **ldap_mem***(3) routines. Otherwise, the caller should take care of destroying and zeroing out the documented search-related fields, and call **ldap_sync_destroy**(3) to free undocumented members set by the API.

REFRESH ONLY

The **refreshOnly** functionality is obtained by periodically calling **ldap_sync_init**(3) with mode set to **LDAP_SYNC_REFRESH_ONLY**, or, which is equivalent, by directly calling **ldap_sync_init_refresh_only**(3). The state of the search, and the consistency of the search parameters, is preserved across calls by passing the **ldap_sync_t** structure as left by the previous call.

REFRESH AND PERSIST

The **refreshAndPersist** functionality is obtained by calling **ldap_sync_init**(3) with mode set to **LDAP_SYNC_REFRESH_AND_PERSIST**, or, which is equivalent, by directly calling **ldap_sync_init_refresh_and_persist**(3) and, after a successful return, by repeatedly polling with **ldap_sync_poll**(3) according to the desired pattern.

A client may insert a call to **ldap_sync_poll**(3) into an external loop to check if any modification was returned; in this case, it might be appropriate to set **ls_timeout** to 0, or to set it to a finite, small value. Otherwise, if the client's main purpose consists in waiting for responses, a timeout of -1 is most suitable, so that the function only returns after some data has been received and handled.

ERRORS

All routines return any LDAP error resulting from a lower-level error in the API calls they are based on, or LDAP_SUCCESS in case of success. **ldap_sync_poll**(3) may return **LDAP_SYNC_REFRESH_REQUIRED** if a full refresh is requested by the server. In this case, it is appropriate to call **ldap_sync_init**(3) again, passing the same **ldap_sync_t** structure as resulted from any

previous call.

NOTES

SEE ALSO

ldap(3), **ldap_search_ext**(3), **ldap_result**(3)**; RFC 4533** (http://www.rfc-editor.org),

AUTHOR

Designed and implemented by Pierangelo Masarati, based on RFC 4533 and loosely inspired by syncrepl code in **slapd**(8).

ACKNOWLEDGEMENTS

Initially developed by **SysNet s.n.c.** **OpenLDAP** is developed and maintained by The OpenLDAP Project (http://www.openldap.org/). **OpenLDAP** is derived from University of Michigan LDAP 3.3 Release.