Network Working Group
Request for Comments: 3928
Category: Standards Track

R. Megginson, Ed.
Netscape Communications Corp.
M. Smith
Pearl Crescent, LLC
O. Natkovich
Yahoo
J. Parham
Microsoft Corporation

October 2004

Lightweight Directory Access Protocol (LDAP) Client Update Protocol (LCUP)

Status of this Memo

Copyright Notice

Abstract

This document defines the Lightweight Directory Access Protocol (LDAP) Client Update Protocol (LCUP). The protocol is intended to allow an LDAP client to synchronize with the content of a directory information tree (DIT) stored by an LDAP server and to be notified about the changes to that content.

Table of Contents

1. Overview

The LCUP protocol is intended to allow LDAP clients to synchronize with the content stored by LDAP servers.

The problem areas addressed by the protocol include:

- Mobile clients that maintain a local read-only copy of the directory data. While off-line, the client uses the local copy of the data. When the client connects to the network, it synchronizes with the current directory content and can optionally receive notification about the changes that occur while it is on-line. For example, a mail client can maintain a local copy of the corporate address book that it synchronizes with the master copy whenever the client is connected to the corporate network.

- Applications intending to synchronize heterogeneous data stores. A meta directory application, for instance, would periodically retrieve a list of modified entries from the directory, construct the changes and apply them to a foreign data store.

- Clients that need to take certain actions when a directory entry is modified. For instance, an electronic mail repository may want to perform a "create mailbox" task when a new person entry is added to an LDAP directory and a "delete mailbox" task when a person entry is removed.

The problem areas not being considered:

- Directory server to directory server synchronization. The IETF is developing a LDAP replication protocol, called LDUP [RFC3384], which is specifically designed to address this problem area.

There are currently several protocols in use for LDAP client server synchronization. While each protocol addresses the needs of a particular group of clients (e.g., on-line clients or off-line clients), none satisfies the requirements of all clients in the target group. For instance, a mobile client that was off-line and wants to become up to date with the server and stay up to date while connected can't be easily supported by any of the existing protocols.

LCUP is designed such that the server does not need to maintain state information specific to individual clients. The server may need to maintain additional state information about attribute modifications, deleted entries, and moved/renamed entries. The clients are responsible for storing the information about how up to date they are with respect to the server's content. LCUP design avoids the need for LCUP-specific update agreements to be made between client and server prior to LCUP use. The client decides when and from where to retrieve the changes. LCUP design requires clients to initiate the update session and "pull" the changes from server.

LCUP operations are subject to administrative and access control policies enforced by the server.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14, RFC 2119 [RFC2119].

2. Applicability

LCUP will work best if the following conditions are met:

1) The server stores some degree of historical state or change information to reduce the amount of wire traffic required for incremental synchronizations. The optimal balance between server state and wire traffic varies amongst implementations and usage scenarios, and is therefore left in the hands of implementers.

2) The client cannot be assumed to understand the physical information model (virtual attributes, operational attributes, subentries, etc.) implemented by the server. Optimizations would be possible if such assumptions could be made.

3) Meta data changes and renames and deletions of large subtrees are very infrequent. LCUP makes these assumptions in order to reduce client complexity required to deal with these special operations, though when they do occur they may result in a large number of incremental update messages or a full resync.

3. Specification of Protocol Elements

The following sections define the new elements required to use this protocol.

3.1. ASN.1 Considerations

Protocol elements are described using ASN.1 [X.680]. The term "BER-encoded" means the element is to be encoded using the Basic Encoding Rules [X.690] under the restrictions detailed in Section 5.1 of [RFC2251]. All ASN.1 in this document uses implicit tags.

3.2. Universally Unique Identifiers

Distinguished names can change, so are therefore unreliable as identifiers. A Universally Unique Identifier (or UUID for short) MUST be used to uniquely identify entries used with LCUP. The UUID is part of the Sync Update control value (see below) returned with each search result. The server SHOULD provide the UUID as a single valued operational attribute of the entry (e.g., "entryUUID"). We RECOMMEND that the server provides a way to do efficient (i.e., indexed) searches for values of UUID, e.g., by using a search filter like (entryUUID=<some UUID value>) to quickly search for and retrieve an entry based on its UUID. Servers SHOULD use a UUID format as specified in [UUID]. The UUID used by LCUP is a value of the following ASN.1 type:

```
LCUPUUID ::= OCTET STRING
```

3.3. LCUP Scheme and LCUP Cookie

The LCUP protocol uses a cookie to hold the state of the client's data with respect to the server's data. Each cookie format is uniquely identified by its scheme. The LCUP Scheme is a value of the following ASN.1 type:

```
LCUPScheme ::= LDAPOID
```

This is the OID which identifies the format of the LCUP Cookie value. The scheme OID, as all object identifiers, MUST be unique for a given cookie scheme. The cookie value may be opaque or it may be exposed to LCUP clients. For cookie schemes that expose their value, the preferred form of documentation is an RFC. It is expected that there will be one or more standards track cookie schemes where the value format is exposed and described in detail.

The LCUP Cookie is a value of the following ASN.1 type:

```
LCUPCookie ::= OCTET STRING
```

This is the actual data describing the state of the client's data. This value may be opaque, or its value may have some well-known format, depending on the scheme.

Further uses of the LCUP Cookie value are described below.

3.4. LCUP Context

A part of the DIT which is enabled for LCUP is referred to as an LCUP Context. A server may support one or more LCUP Contexts. For example, a server with two naming contexts may support LCUP in one naming context but not the other, or support different LCUP cookie schemes in each naming context. Each LCUP Context MAY use a different cookie scheme. An LCUP search will not cross an LCUP Context boundary, but will instead return a SearchResultReference message, with the LDAP URL specifying the same host and port as currently being searched, and with the baseDN set to the baseDN of the new LCUP Context. The client is then responsible for issuing another search using the new baseDN, and possibly a different cookie if that LCUP Context uses a different cookie. The client is responsible for maintaining a mapping of the LDAP URL to its corresponding cookie.

3.5. Additional LDAP Result Codes defined by LCUP

Implementations of this specification SHALL recognize the following additional resultCode values. The LDAP result code names and numbers defined in the following table have been assigned by IANA per RFC 3383 [RFC3383].

lcupResourcesExhausted	(113)	the server is running out of resources
lcupSecurityViolation	(114)	the client is suspected of malicious actions
lcupInvalidData	(115)	invalid scheme or cookie was supplied by the client

lcupUnsupportedScheme	(116)	The cookie scheme is a valid OID but is not supported by this server
lcupReloadRequired	(117)	indicates that client data needs to be reinitialized. This reason is returned if the server does not contain sufficient information to synchronize the client or if the server's data was reloaded since the last synchronization session

The uses of these codes are described below.

3.6. Sync Request Control

The Sync Request Control is an LDAP Control [RFC2251, Section 4.1.2] where the controlType is the object identifier 1.3.6.1.1.7.1 and the controlValue, an OCTET STRING, contains a BER-encoded syncRequestControlValue.

```
syncRequestControlValue ::= SEQUENCE {
   updateType           ENUMERATED {
                           syncOnly       (0),
                           syncAndPersist (1),
                           persistOnly    (2) },
   sendCookieInterval   [0] INTEGER    OPTIONAL,
   scheme               [1] LCUPScheme OPTIONAL,
   cookie               [2] LCUPCookie OPTIONAL
  }
```

sendCookieInterval - the server SHOULD send the cookie back in the Sync Update control value (defined below) for every sendCookieInterval number of SearchResultEntry and SearchResultReference PDUs returned to the client. For example, if the value is 5, the server SHOULD send the cookie back in the Sync Update control value for every 5 search results returned to the client. If this value is absent, zero or less than zero, the server chooses the interval.

The Sync Request Control is only applicable to the searchRequest message. Use of this control is described below.

3.7. Sync Update Control

The Sync Update Control is an LDAP Control [RFC2251, Section 4.1.2] where the controlType is the object identifier 1.3.6.1.1.7.2 and the controlValue, an OCTET STRING, contains a BER-encoded syncUpdateControlValue.

```
syncUpdateControlValue ::= SEQUENCE {
   stateUpdate   BOOLEAN,
   entryUUID     [0] LCUPUUID OPTIONAL, -- REQUIRED for entries --
   UUIDAttribute [1] AttributeType OPTIONAL,
   entryLeftSet  [2] BOOLEAN,
   persistPhase  [3] BOOLEAN,
   scheme        [4] LCUPScheme OPTIONAL,
   cookie        [5] LCUPCookie OPTIONAL
}
```

The field UUIDAttribute contains the name or OID of the attribute that the client should use to perform searches for entries based on the UUID. The client should be able to use it in an equality search filter, e.g., "(<uuid attribute>=<entry UUID value>)" and should be able to use it in the attribute list of the search request to return its value. The UUIDAttribute field may be omitted if the server does not support searching on the UUID values.

The Sync Update Control is only applicable to SearchResultEntry and SearchResultReference messages. Although entryUUID is OPTIONAL, it MUST be used with SearchResultEntry messages. Use of this control is described below.

3.8. Sync Done Control

The Sync Done Control is an LDAP Control [RFC2251, Section 4.1.2] where the controlType is the object identifier 1.3.6.1.1.7.3 and the controlValue contains a BER-encoded syncDoneValue.

```
syncDoneValue ::= SEQUENCE {
   scheme      [0] LCUPScheme OPTIONAL,
   cookie      [1] LCUPCookie OPTIONAL
}
```

The Sync Done Control is only applicable to SearchResultDone message. Use of this control is described below.

4. Protocol Usage and Flow

4.1. LCUP Search Requests

A client initiates a synchronization or persistent search session with a server by attaching a Sync Request control to an LDAP searchRequest message. The search specification determines the part of the directory information tree (DIT) the client wishes to synchronize with, the set of attributes it is interested in and the amount of data the client is willing to receive. The Sync Request control contains the client's request specification.

If there is an error condition, the server MUST immediately return a SearchResultDone message with the resultCode set to an error code. This table maps a condition to its corresponding behavior and resultCode.

Condition	Behavior or resultCode
Sync Request Control is not supported	Server behaves as [RFC2251, Section 4.1.2] - specifically, if the criticality of the control is FALSE, the server will process the request as a normal search request
Scheme is not supported	lcupUnsupportedScheme
A control value field is invalid (e.g., illegal updateType, or the scheme is not a valid OID, or the cookie is invalid)	lcupInvalidData
Server is running out of resources	lcupResourcesExhausted
Server suspects client of malicious behavior (frequent connects/disconnects, etc.)	lcupSecurityViolation
The server cannot bring the client up to date (server data has been reloaded, or other changes prevent convergence)	lcupReloadRequired

4.1.1. Initial Synchronization and Full Resync

For an initial synchronization or full resync, the fields of the Sync Request control MUST be specified as follows:

```
updateType         - MUST be set to syncOnly or syncAndPersist
sendCookieInterval - MAY be set
scheme             - MAY be set - if set, the server MUST use this
                     specified scheme or return lcupUnsupportedScheme
                     (see above) - if not set, the server MAY use any
                     scheme it supports.
cookie             - MUST NOT be set
```

If the request was successful, the client will receive results as described in the section "LCUP Search Responses" below.

4.1.2. Incremental or Update Synchronization

For an incremental or update synchronization, the fields of the Sync Request control MUST be specified as follows:

```
updateType         - MUST be set to syncOnly or syncAndPersist
sendCookieInterval - MAY be set
scheme             - MUST be set
cookie             - MUST be set
```

The client SHOULD always use the latest cookie it received from the server.

If the request was successful, the client will receive results as described in the section "LCUP Search Responses" below.

4.1.3. Persistent Only

For persistent only search request, the fields of the Sync Request MUST be specified as follows:

```
updateType          - MUST be set to persistOnly
sendCookieInterval  - MAY be set
scheme              - MAY be set - if set, the server MUST use this
                      specified scheme or return
                      lcupUnsupportedScheme (see above) - if not set,
                      the server MAY use any scheme it supports.
cookie              - MAY be set, but the server MUST ignore it
```

If the request was successful, the client will receive results as described in the section "LCUP Search Responses" below.

4.2. LCUP Search Responses

In response to the client's LCUP request, the server returns zero or more SearchResultEntry or SearchResultReference PDUs that fit the client's specification, followed by a SearchResultDone PDU. The behavior is as specified in [RFC2251 Section 4.5]. Each SearchResultEntry or SearchResultReference PDU also contains a Sync Update control that describes the LCUP state of the returned entry. The SearchResultDone PDU contains a Sync Done control. The following sections specify behaviors in addition to [RFC2251 Section 4.5].

4.2.1 Sync Update Informational Responses

The server may use the Sync Update control to return information not related to a particular entry. It MAY do this at any time to return a cookie to the client, or to inform the client that the sync phase of a syncAndPersist search is complete and the persist phase has begun. It MAY do this during the persist phase even though no entry has changed that would have normally triggered a response. In order to do this, it is REQUIRED to return the following:

- A SearchResultEntry PDU with the objectName field set to the DN of the baseObject of the search request and with an empty attribute list.

- A Sync Update control value with the fields set to the following:

```
stateUpdate   - MUST be set to TRUE
entryUUID     - SHOULD be set to the UUID of the baseObject of the
                search request
entryLeftSet  - MUST be set to FALSE
persistPhase  - MUST be FALSE if the search is in the sync phase of a
                request, and MUST be TRUE if the search is in the
                persist phase
UUIDAttribute - SHOULD only be set if this is either the first result
                returned or if the attribute has changed
scheme        - MUST be set if the cookie is set and the cookie
                format has changed; otherwise, it MAY be omitted
cookie        - SHOULD be set
```

If the server merely wants to return a cookie to the client, it should return as above with the cookie field set.

During a syncAndPersist request, the server MUST return (as above) immediately after the last entry of the sync phase has been sent and before the first entry of the persist phase has been sent. In this case, the persistPhase field MUST be set to TRUE. This allows the client to know that the sync phase is complete and the persist phase is starting.

4.2.2 Cookie Return Frequency

The cookie field of the Sync Update control value MAY be set in any returned result, during both the sync phase and the persist phase. The server should return the cookie to the client often enough for the client to resync in a reasonable period of time in case the search is disconnected or otherwise terminated. The sendCookieInterval field in the Sync Request control is a suggestion

to the server of how often to return the cookie in the Sync Update control. The server SHOULD respect this value.

The scheme field of the Sync Update control value MUST be set if the cookie is set and the cookie format has changed; otherwise, it MAY be omitted.

Some clients may have unreliable connections, for example, a wireless device or a WAN connection. These clients may want to insure that the cookie is returned often in the Sync Update control value, so that if they have to reconnect, they do not have to process many redundant entries. These clients should set the sendCookieInterval in the Sync Request control value to a low number, perhaps even 1. Some clients may have a limited bandwidth connection, and may not want to receive the cookie very often, or even at all (however, the cookie is always sent back in the Sync Done control value upon successful completion). These clients should set the sendCookieInterval in the Sync Request control value to a high number.

A reasonable behavior of the server is to return the cookie only when data in the LCUP context has changed, even if the client has specified a frequent sendCookieInterval. If nothing has changed, the server can probably save some bandwidth by not returning the cookie.

4.2.3. Definition of an Entry That Has Entered the Result Set

An entry SHALL BE considered to have entered the client's search result set if one of the following conditions is met:

- During the sync phase for an incremental sync operation, the entry is present in the search result set but was not present before; this can be due to the entry being added via an LDAP Add operation, or by the entry being moved into the result set by an LDAP Modify DN operation, or by some modification to the entry that causes it to enter the result set (e.g., adding an attribute value that matches the clients search filter), or by some meta-data change that causes the entry to enter the result set (e.g., relaxing of some access control that permits the entry to be visible to the client).

- During the persist phase for a persistent search operation, the entry enters the search result set; this can be due to the entry being added via an LDAP Add operation, or by the entry being moved into the result set by an LDAP Modify DN operation, or by some modification to the entry that causes it to enter the result set (e.g., adding an attribute value that matches the clients search filter), or by some meta-data change that causes the entry to

enter the result set (e.g., relaxing of some access control that permits the entry to be visible to the client).

4.2.4. Definition of an Entry That Has Changed

An entry SHALL BE considered to be changed if one or more of the attributes in the attribute list in the search request have been modified. For example, if the search request listed the attributes "cn sn uid", and there is an entry in the client's search result set with the "cn" attribute that has been modified, the entry is considered to be modified. The modification may be due to an LDAP Modify operation or by some change to the meta-data for the entry (e.g., virtual attributes) that causes some change to the value of the specified attributes.

The converse of this is that an entry SHALL NOT BE considered to be changed if none of the attributes in the attribute list of the search request are modified attributes of the entry. For example, if the search request listed the attributes "cn sn uid", and there is an entry in the client's search result set with the "foo" attribute that has been modified, and none of the "cn" or "sn" or "uid" attributes have been modified, the entry is NOT considered to be changed.

4.2.5. Definition of an Entry That Has Left the Result Set

An entry SHALL BE considered to have left the client's search result set if one of the following conditions is met:

- During the sync phase for an incremental sync operation, the entry is not present in the search result set but was present before; this can be due to the entry being deleted via an LDAP Delete operation, or by the entry leaving the result set via an LDAP Modify DN operation, or by some modification to the entry that causes it to leave the result set (e.g., changing/removing an attribute value so that it no longer matches the client's search filter), or by some meta-data change that causes the entry to leave the result set (e.g., adding of some access control that denies the entry to be visible to the client).

- During the persist phase for a persistent search operation, the entry leaves the search result set; this can be due to the entry being deleted via an LDAP Delete operation, or by the entry leaving the result set via an LDAP Modify DN operation, or by some modification to the entry that causes it to leave the result set (e.g., changing/removing an attribute value so that it no longer matches the client's search filter), or by some meta-data change

that causes the entry to leave the result set (e.g., adding of some access control that denies the entry to be visible to the client).

4.2.6. Results For Entries Present in the Result Set

An entry SHOULD be returned as present under the following conditions:

- The request is an initial synchronization or full resync request and the entry is present in the client's search result set
- The request is an incremental synchronization and the entry has changed or entered the result set since the last sync
- The search is in the persist phase and the entry enters the result set or changes

For a SearchResultEntry return, the fields of the Sync Update control value MUST be set as follows:

```
stateUpdate   - MUST be set to FALSE
entryUUID     - MUST be set to the UUID of the entry
entryLeftSet  - MUST be set to FALSE
persistPhase  - MUST be set to FALSE if during the sync phase or TRUE
                if during the persist phase
UUIDAttribute - SHOULD only be set if this is either the first result
                returned or if the attribute has changed
scheme        - as above
cookie        - as above
```

The searchResultReference return will look the same, except that the entryUUID is not required. If it is specified, it MUST contain the UUID of the DSE holding the reference knowledge.

4.2.7. Results For Entries That Have Left the Result Set

An entry SHOULD be returned as having left the result set under the following conditions:

- The request is an incremental synchronization during the sync phase and the entry has left the result set
- The search is in the persist phase and the entry has left the result set

- The entry has left the result set as a result of an LDAP Delete or LDAP Modify DN operation against the entry itself (i.e., not as a result of an operation against its parent or ancestor)

For a SearchResultEntry return where the entry has left the result set, the fields of the Sync Update control value MUST be set as follows:

```
stateUpdate   - MUST be set to FALSE
entryUUID     - MUST be set to the UUID of the entry that left the
                result set
entryLeftSet  - MUST be set to TRUE
persistPhase  - MUST be set to FALSE if during the sync phase or TRUE
                if during the persist phase
UUIDAttribute - SHOULD only be set if this is either the first result
                returned or if the attribute has changed
scheme        - as above
cookie        - as above
```

The searchResultReference return will look the same, except that the entryUUID is not required. If it is specified, it MUST contain the UUID of the DSE holding the reference knowledge.

Some server implementations keep track of deleted entries using a tombstone - a hidden entry that keeps track of the state, but not all of the data, of an entry that has been deleted. In this case, the tombstone may not contain all of the original attributes of the entry, and therefore it may be impossible for the server to determine if an entry should be removed from the result set based on the attributes in the client's search request. Servers SHOULD keep enough information about the attributes in the deleted entries to determine if an entry should be removed from the result set. Since this may not be possible, the server MAY return an entry as having left the result set even if it is not or never was in the client's result set. Clients MUST ignore these notifications.

4.3. Responses Requiring Special Consideration

The following sections describe special handling that may be required when returning results.

4.3.1. Returning Results During the Persistent Phase

During the persistent phase, the server SHOULD return the changed entries to the client as quickly as possible.

4.3.2. No Mixing of Sync Phase with Persist Phase

During a sync phase, the server MUST NOT return any entries with the persistPhase flag set to TRUE, and during the persist phase, all entries returned MUST have the persistPhase flag set to TRUE. The server MUST NOT mix and match sync phase entries with persist phase entries. If there are any sync phase entries to return, they MUST be returned before any persist phase entries are returned.

4.3.3. Returning Updated Results During the Sync Phase

There may be updates to the entries in the result set of a sync phase search during the actual search operation. If the DSA is under a heavy update load, and it attempts to send all of those updated entries to the client in addition to the other updates it was already planning to send for the sync phase, the server may never get to the end of the sync phase. Therefore, it is left up to the discretion of the server implementation to decide when the client is "in sync" - that is, when to end a syncOnly request, or when to send the Sync Update Informational Response between the sync phase and the persist phase of a syncAndPersist request. The server MAY send the same entry multiple times during the sync phase if the entry changes during the sync phase.

A reasonable behavior is for the server to generate a cookie based on the server state at the time the client initiated the LCUP request, and only send entries up to that point during the sync phase. Entries updated after that point will be returned only during the persist phase of a syncAndPersist request, or only upon an incremental synchronization.

4.3.4. Operational Attributes and Administrative Entries

An operational attribute SHOULD be returned if it is specified in the attributes list and would normally be returned as subject to the constraints of [RFC2251 Section 4.5]. If the server does not support syncing of operational attributes, the server MUST return a SearchResultDone message with a resultCode of unwillingToPerform.

LDAP Subentries [RFC3672] SHOULD be returned if they would normally be returned by the search request. If the server does not support syncing of LDAP Subentries, and the server can determine from the search request that the client has requested LDAP Subentries to be returned (e.g., search control or search filter), the server MUST return a SearchResultDone message with a resultCode of unwillingToPerform. Otherwise, the server MAY simply omit returning LDAP Subentries.

4.3.5. Virtual Attributes

An entry may have attributes whose presence in the entry, or presence of values of the attribute, is generated on the fly, possibly by some mechanism outside of the entry, elsewhere in the DIT. An example of this is collective attributes [RFC3671]. These attributes shall be referred to in this document as virtual attributes.

LCUP treats these attributes the same way as normal, non-virtual attributes. A virtual attribute SHOULD be returned if it is specified in the attributes list and would normally be returned as subject to the constraints of [RFC2251 Section 4.5]. If the server does not support syncing of virtual attributes, the server MUST return a SearchResultDone message with a resultCode of unwillingToPerform.

One consequence of this is that if you change the definition of a virtual attribute such that it makes the value of that attribute change in many entries in the client's search scope, this means that a server may have to return many entries to the client as a result of that one change. It is not anticipated that this will be a frequent occurrence, and the server has the option to simply force the client to resync if necessary.

It is also possible that a future LDAP control will allow the client to request only virtual or only non-virtual attributes.

4.3.6. Modify DN and Delete Operations Applied to Subtrees

There is a special case where a Modify DN or a Delete operation is applied to the base entry of a subtree, and either that base entry or entries in the subtree are within the scope of an LCUP search request. In this case, all of the entries in the subtree are implicitly renamed or removed.

In either of these cases, the server MUST do one of the following:

- treat all of these entries as having been renamed or removed and return each entry to the client as such

- decide that this would be prohibitively expensive, and force the client to resync

If the search base object has been renamed, and the client has received a noSuchObject as the result of a search request, the client MAY use the entryUUID and UUIDAttribute to locate the new DN that is the result of the modify DN operation.

4.3.7. Convergence Guarantees

If at any time during an LCUP search, either during the sync phase or the persist phase, the server determines that it cannot guarantee that it can bring the client's copy of the data to eventual convergence, it SHOULD immediately terminate the LCUP search request and return a SearchResultDone message with a resultCode of lcupReloadRequired. This can also happen at the beginning of an incremental synchronization request, if the client presents a cookie that is out of date or otherwise unable to be processed. The client should then issue an initial synchronization request.

This can happen, for example, if the data on the server is reloaded, or if there has been some change to the meta-data that makes it impossible for the server to determine if a particular entry should or should not be part of the search result set, or if the meta-data change makes it too resource intensive for the server to calculate the proper result set.

The server can also return lcupReloadRequired if it determines that it would be more efficient for the client to perform a reload, for example, if too many entries have changed and a simple reload would be much faster.

4.4. LCUP Search Termination

4.4.1. Server Initiated Termination

When the server has successfully finished processing the client's request, it attaches a Sync Done control to the SearchResultDone message and sends it to the client. However, if the SearchResultDone message contains a resultCode that is not success or canceled, the Sync Done control MAY be omitted. Although the LCUP cookie is OPTIONAL in the Sync Done control value, it MUST be set if the SearchResultDone resultCode is success or canceled. The server SHOULD also set the cookie if the resultCode is lcupResourcesExhausted, timeLimitExceeded, sizeLimitExceeded, or adminLimitExceeded. This allows the client to more easily resync later. If some error occurred, either an LDAP search error (e.g., insufficientAccessRights) or an LCUP error (e.g., lcupUnsupportedScheme), the cookie MAY be omitted. If the cookie is set, the scheme MUST be set also if the cookie format has changed, otherwise, it MAY be omitted.

If server resources become tight, the server can terminate one or more search operations by sending a SearchResultDone message to the client(s) with a resultCode of lcupResourcesExhausted. The server SHOULD attach a Sync Done control with the cookie set. A server side

policy is used to decide which searches to terminate. This can also be used as a security mechanism to disconnect clients that are suspected of malicious actions, but if the server can infer that the client is malicious, the server SHOULD return lcupSecurityViolation instead.

4.4.2. Client Initiated Termination

If the client needs to terminate the synchronization process and it wishes to obtain the cookie that represents the current state of its data, it issues an LDAP Cancel operation [RFC3909]. The server responds immediately with a LDAP Cancel response [RFC3909]. The server MAY send any pending SearchResultEntry or SearchResultReference PDUs if the server cannot easily abort or remove those search results from its outgoing queue. The server SHOULD send as few of these remaining messages as possible. Finally, the server sends the message SearchResultDone with the Sync Done control attached. If the search was successful up to that point, the resultCode field of the SearchResultDone message MUST be canceled [RFC3909], and the cookie MUST be set in the Sync Done control. If there is an error condition, the server MAY return as described in section 4.4.1 above, or MAY return as described in [RFC3909].

If the client is not interested in the state information, it can simply abandon the search operation or disconnect from the server.

4.5. Size and Time Limits

The server SHALL support size and time limits as specified in [RFC2251, Section 5]. The server SHOULD ensure that if the operation is terminated due to these conditions, the cookie is sent back to the client.

4.6. Operations on the Same Connection

It is permissible for the client to issue other LDAP operations on the connection used by the protocol. Since each LDAP request/response carries a message id there will be no ambiguity about which PDU belongs to which operation. By sharing the connection among multiple operations, the server will be able to conserve its resources.

4.7. Interactions with Other Controls

LCUP defines neither restrictions nor guarantees about the ability to use the controls defined in this document in conjunction with other LDAP controls, except for the following: A server MAY ignore non-critical controls supplied with the LCUP control. A server MAY

ignore an LCUP defined control if it is non-critical and it is supplied with other critical controls. If a server receives a critical LCUP control with another critical control, and the server does not support both controls at the same time, the server SHOULD return unavailableCriticalExtension.

It is up to the server implementation to determine if the server supports controls such as the Sort or VLV or similar controls that change the order of the entries sent to the client. But note that it may be difficult or impossible for a server to perform an incremental synchronization in the presence of such controls, since the cookie will typically be based off a change number, or Change Sequence Number (CSN), or timestamp, or some criteria other than an alphabetical order.

4.8. Replication Considerations

Use of an LCUP cookie with multiple DSAs in a replicated environment is not defined by LCUP. An implementation of LCUP may support continuation of an LCUP session with another DSA holding a replica of the LCUP context. Clients MAY submit cookies returned by one DSA to a different DSA; it is up to the server to determine if a cookie is one they recognize or not and to return an appropriate result code if not.

5. Client Side Considerations

5.1. Using Cookies with Different Search Criteria

The cookie received from the server after a synchronization session SHOULD only be used with the same search specification as the search that generated the cookie. Some servers MAY allow the cookie to be used with a more restrictive search specification than the search that generated the cookie. If the server does not support the cookie, it MUST return lcupInvalidCookie. This is because the client can end up with an incomplete data store otherwise. A more restrictive search specification is one that would generate a subset of the data produced by the original search specification.

5.2. Renaming the Base Object

Because an LCUP client specifies the area of the tree with which it wishes to synchronize through the standard LDAP search specification, the client can be returned noSuchObject error if the root of the synchronization area was renamed between the synchronization sessions or during a synchronization session. If this condition occurs, the client can attempt to locate the root by using the root's UUID saved in client's local data store. It then can repeat the synchronization

request using the new search base. In general, a client can detect that an entry was renamed and apply the changes received to the right entry by using the UUID rather than DN based addressing.

5.3. Use of Persistent Searches With Respect to Resources

Each active persistent operation requires that an open TCP connection be maintained between an LDAP client and an LDAP server that might not otherwise be kept open. Therefore, client implementors are encouraged to avoid using persistent operations for non-essential tasks and to close idle LDAP connections as soon as practical. The server may close connections if server resources become tight.

5.4. Continuation References to Other LCUP Contexts

The client MAY receive a continuation reference (SearchResultReference [RFC2251 SECTION 4.5.3]) if the search request spans multiple parts of the DIT, some of which may require a different LCUP cookie, some of which may not even be managed by LCUP. The client SHOULD maintain a cache of the LDAP URLs returned in the continuation references and the cookies associated with them. The client is responsible for performing another LCUP search to follow the references, and SHOULD use the cookie corresponding to the LDAP URL for that reference (if it has a cookie).

5.5. Referral Handling

The client may receive a referral (Referral [RFC2251 SECTION 4.1.11]) when the search base is a subordinate reference, and this will end the operation.

5.6. Multiple Copies of Same Entry During Sync Phase

The server MAY send the same entry multiple times during a sync phase if the entry changes during the sync phase. The client SHOULD use the last sent copy of the entry as the current one.

5.7. Handling Server Out of Resources Condition

If the client receives an lcupResourcesExhausted or lcupSecurityViolation resultCode, the client SHOULD wait at least 5 seconds before attempting another operation. It is RECOMMENDED that the client use an exponential backoff strategy, but different clients may want to use different backoff strategies.

6. Server Implementation Considerations

6.1. Server Support for UUIDs

Servers MUST support UUIDs. UUIDs are required in the Sync Update control. Additionally, server implementers SHOULD make the UUID values for the entries available as an attribute of the entry, and provide indexing or other mechanisms to allow clients to search for an entry using the UUID attribute in the search filter. The syncUpdate control provides a field UUIDAttribute to allow the server to let the client know the name or OID of the attribute to use to search for an entry by UUID.

6.2. Example of Using an RUV as the Cookie Value

By design, the protocol supports multiple cookie schemes. This is to allow different implementations the flexibility of storing any information applicable to their environment. A reasonable implementation for an LDUP compliant server would be to use the Replica Update Vector (RUV). For each master, RUV contains the largest CSN seen from this master. In addition, RUV implemented by some directory servers (not yet in LDUP) contains replica generation - an opaque string that identifies the replica's data store. The replica generation value changes whenever the replica's data is reloaded. Replica generation is intended to signal the replication/synchronization peers that the replica's data was reloaded and that all other replicas need to be reinitialized. RUV satisfies the three most important properties of the cookie: (1) it uniquely identifies the state of client's data, (2) it can be used to synchronize with multiple servers, and (3) it can be used to detect that the server's data was reloaded. If RUV is used as the cookie, entries last modified by a particular master must be sent to the client in the order of their last modified CSN. This ordering guarantees that the RUV can be updated after each entry is sent.

6.3. Cookie Support Issues

6.3.1. Support for Multiple Cookie Schemes

A server may support one or more LCUP cookie schemes. It is expected that schemes will be published along with their OIDs as RFCs. The server's DIT may be partitioned into different sections which may have different cookies associated with them. For example, some servers may use some sort of replication mechanism to support LCUP. If so, the DIT may be partitioned into multiple replicas. A client may send an LCUP search request that spans multiple replicas. Some parts of the DIT spanned by the search request scope may support LCUP and some may not. The server MUST send a SearchResultReference

[RFC2251, SECTION 4.5.3] when the LCUP Context for a returned entry changes. The server SHOULD send all references to other LCUP Contexts in the search scope first, in order to allow the clients to process these searches in parallel. The LDAP URL(s) returned MUST contain the DN(s) of the base of another section of the DIT (however the server implementation has partitioned the DIT). The client will then issue another LCUP search using the LDAP URL returned. Each section of the DIT MAY require a different cookie value, so the client SHOULD maintain a cache, mapping the different LDAP URL values to different cookies. If the cookie changes, the scheme may change as well, but the cookie scheme MUST be the same within a given LCUP Context.

6.3.2. Information Contained in the Cookie

The cookie must contain enough information to allow the server to determine whether the cookie can be safely used with the search specification it is attached to. As discussed earlier in the document, the cookie SHOULD only be used with the search specification that is equal to the one for which the cookie was generated, but some servers MAY support using a cookie with a search specification that is more restrictive than the one used to generate the cookie.

6.4. Persist Phase Response Time

The specification makes no guarantees about how soon a server should send notification of a changed entry to the client during the persist phase. This is intentional as any specific maximum delay would be impossible to meet in a distributed directory service implementation. Server implementers are encouraged to minimize the delay before sending notifications to ensure that clients' needs for timeliness of change notification are met.

6.5. Scaling Considerations

Implementers of servers that support the mechanism described in this document should ensure that their implementation scales well as the number of active persistent operations and the number of changes made in the directory increases. Server implementers are also encouraged to support a large number of client connections if they need to support large numbers of persistent operations.

6.6. Alias Dereferencing

LCUP design does not consider issues associated with alias dereferencing in search. Clients MUST specify derefAliases as either neverDerefAliases or derefFindingBaseObj. Servers are to return protocolError if the client specifies either derefInSearching or derefAlways.

7. Synchronizing Heterogeneous Data Stores

Clients, like a meta directory join engine, synchronizing multiple writable data stores, will only work correctly if each piece of information comes from a single authoritative data source. In a replicated environment, an LCUP Context should employ the same conflict resolution scheme across all its replicas. This is because different systems have different notions of time and different update resolution procedures. As a result, a change applied on one system can be discarded by the other, thus preventing the data stores from converging.

8. IANA Considerations

This document lists several values that have been registered by the IANA. The following LDAP result codes have been assigned by IANA as described in section 3.6 of [RFC3383]:

lcupResourcesExhausted	113
lcupSecurityViolation	114
lcupInvalidData	115
lcupUnsupportedScheme	116
lcupReloadRequired	117

The three controls defined in this document have been registered as LDAP Protocol Mechanisms as described in section 3.2 of [RFC3383]. One OID, 1.3.6.1.1.7, has been assigned by IANA as described in section 3.1 of [RFC3383]. The OIDs for the controls defined in this document are derived as follows from the one assigned by IANA:

LCUP Sync Request Control	1.3.6.1.1.7.1
LCUP Sync Update Control	1.3.6.1.1.7.2
LCUP Sync Done Control	1.3.6.1.1.7.3

9. Security Considerations

In some situations, it may be important to prevent general exposure of information about changes that occur in an LDAP server. Therefore, servers that implement the mechanism described in this document SHOULD provide a means to enforce access control on the entries

returned and MAY also provide specific access control mechanisms to control the use of the controls and extended operations defined in this document.

As with normal LDAP search requests, a malicious client can initiate a large number of persistent search requests in an attempt to consume all available server resources and deny service to legitimate clients. The protocol provides the means to stop malicious clients by disconnecting them from the server. The servers that implement the mechanism SHOULD provide the means to detect the malicious clients. In addition, the servers SHOULD provide the means to limit the number of resources that can be consumed by a single client.

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2251] Wahl, M., Howes, T., and S. Kille, "Lightweight Directory Access Protocol (v3)", RFC 2251, December 1997.

[RFC3383] Zeilenga, K., "Internet Assigned Numbers Authority (IANA) Considerations for Lightweight Directory Access Protocol (LDAP)", BCP 64, RFC 3383, September 2002.

[RFC3909] Zeilenga, K., "Lightweight Directory Access Protocol (LDAP) Cancel Operation", RFC 3909, October 2004.

[X.680] ITU-T, "Abstract Syntax Notation One (ASN.1) - Specification of Basic Notation", X.680, 1994.

[X.690] ITU-T, "Specification of ASN.1 encoding rules: Basic, Canonical, and Distinguished Encoding Rules", X.690, 1994.

[UUID] International Organization for Standardization (ISO), "Information technology - Open Systems Interconnection - Remote Procedure Call", ISO/IEC 11578:1996.

10.2. Informative References

[RFC3384] Stokes, E., Weiser, R., Moats, R., and R. Huber, "Lightweight Directory Access Protocol (version 3) Replication Requirements", RFC 3384, October 2002.

[RFC3671] Zeilenga, K., "Collective Attributes in the Lightweight Directory Access Protocol (LDAP)", RFC 3671, December 2003.

[RFC3672] Zeilenga, K. and S. Legg, "Subentries in the Lightweight Directory Access Protocol (LDAP)", RFC 3672, December 2003.

11. Acknowledgments

The LCUP protocol is based in part on the Persistent Search Change Notification Mechanism defined by Mark Smith, Gordon Good, Tim Howes, and Rob Weltman, the LDAPv3 Triggered Search Control defined by Mark Wahl, and the LDAP Control for Directory Synchronization defined by Michael Armijo. The members of the IETF LDUP working group made significant contributions to this document.

Appendix - Features Left Out of LCUP

There are several features present in other protocols or considered useful by clients that are currently not included in the protocol primarily because they are difficult to implement on the server. These features are briefly discussed in this section.

Triggered Search Change Type

This feature is present in the Triggered Search specification. A flag is attached to each entry returned to the client indicating the reason why this entry is returned. The possible reasons from the document are:

- notChange: the entry existed in the directory and matched the search at the time the operation is being performed,
- enteredSet: the entry entered the result,
- leftSet: the entry left the result,
- modified: the entry was part of the result set, was modified or renamed, and still is in the result set.

The leftSet feature is particularly useful because it indicates to the client that an entry is no longer within the client's search specification and the client can remove the associated data from its data store. Ironically, this feature is the hardest to implement on the server because the server does not keep track of the client's state and has no easy way of telling which entries moved out of scope between synchronization sessions with the client. A compromise could be reached by only providing this feature for the operations that occur while the client is connected to the server. This is easier to accomplish because the decision about the change type can be made based only on the change without need for any historical information. This, however, would add complexity to the protocol.

Persistent Search Change Type

This feature is present in the Persistent Search specification. Persistent search has the notion of changeTypes. The client specifies which type of updates will cause entries to be returned, and optionally whether the server tags each returned entry with the type of change that caused that entry to be returned.

For LCUP, the intention is full synchronization, not partial. Each entry returned by an LCUP search will have some change associated with it that may concern the client. The client may have to have a

local index of entries by DN or UUID to determine if the entry has been added or just modified. It is easy for clients to determine if the entry has been deleted because the entryLeftSet value of the Sync Update control will be TRUE.

Sending Changes

Some earlier synchronization protocols sent the client(s) only the modified attributes of the entry rather than the entire entry. While this approach can significantly reduce the amount of data returned to the client, it has several disadvantages. First, unless a separate mechanism (like the change type described above) is used to notify the client about entries moving into the search scope, sending only the changes can result in the client having an incomplete version of the data. Let's consider an example. An attribute of an entry is modified. As a result of the change, the entry enters the scope of the client's search. If only the changes are sent, the client would never see the initial data of the entry. Second, this feature is hard to implement since the server might not contain sufficient information to construct the changes based solely on the server's state and the client's cookie. On the other hand, this feature can be easily implemented by the client assuming that the client has the previous version of the data and can perform value by value comparisons.

Data Size Limits

Some earlier synchronization protocols allowed clients to control the amount of data sent to them in the search response. This feature was intended to allow clients with limited resources to process synchronization data in batches. However, an LDAP search operation already provides the means for the client to specify the size limit by setting the sizeLimit field in the SearchRequest to the maximum number of entries the client is willing to receive. While the granularity is not the same, the assumption is that regular LDAP clients that can deal with the limitations of the LDAP protocol will implement LCUP.

Data Ordering

Some earlier synchronization protocols allowed a client to specify that parent entries should be sent before the children for add operations and children entries sent before their parents during delete operations. This ordering helps clients to maintain a hierarchical view of the data in their data store. While possibly useful, this feature is relatively hard to implement and is expensive to perform.

Authors' Addresses

Rich Megginson
Netscape Communications Corp., an America Online company.
360 W. Caribbean Drive
Sunnyvale, CA 94089
USA

Phone: +1 505 797-7762
EMail: rmegginson0224@aol.com

Olga Natkovich
Yahoo, Inc.
701 First Ave.
Sunnyvale, CA 94089
USA

Phone: +1 408 349-6153
EMail: olgan@yahoo-inc.com

Mark Smith
Pearl Crescent, LLC
447 Marlpool Drive
Saline, MI 48176
USA

Phone: +1 734 944-2856
EMail: mcs@pearlcrescent.com

Jeff Parham
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399
USA

Phone: +1 425 882-8080
EMail: jeffparh@microsoft.com

Full Copyright Statement

Intellectual Property

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.