LDAP Replication Architecture

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or made obsolete by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This draft, file name draft-ietf-ldup-model-04.txt, is intended to be become a Proposed Standard RFC, to be published by the IETF Working Group LDUP. Distribution of this document is unlimited. Comments should be sent to the LDUP Replication mailing list <ldup@imc.org> or to the authors.

This Internet-Draft expires on 11 June 2001
1. Abstract

This architectural document outlines a suite of schema and protocol extensions to LDAPv3 that enables the robust, reliable, server-to-server exchange of directory content and changes.

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 RFC 2119 [RFC2119]. The sections below reiterate these definitions and include some additional ones.
2. Table of Contents

1. ABSTRACT ........................................................ 2

2. TABLE OF CONTENTS ............................................. 3

3. INTRODUCTION ................................................ 6

3.1 SCOPE ........................................................ 6

3.2 DOCUMENT OBJECTIVES ....................................... 7

3.3 DOCUMENT NON-OBJECTIVES ................................. 7

3.4 EXISTING IMPLEMENTATIONS ................................. 8

3.4.1 Replication Log Implementations ....................... 9

3.4.2 State-Based Implementations ............................ 9

3.5 TERMS AND DEFINITIONS ..................................... 9

3.6 CONSISTENCY MODELS ....................................... 11

3.7 LDAP CONSTRAINTS .......................................... 12

4. DIRECTORY MODEL ............................................. 13

4.1 REPLICA TYPE ............................................. 13

4.1.1 Primary Replica ....................................... 13

4.1.2 Updateable Replica .................................... 13

4.1.3 Read-Only Replica ..................................... 13

4.1.4 Fractional Replicas ................................... 14

4.2 SUB-ENTRIES ................................................ 14

4.3 GLUE ENTRIES ............................................... 14

4.4 UNIQUE IDENTIFIERS ....................................... 14

4.5 CHANGE SEQUENCE NUMBER ................................. 14

4.5.1 CSN Composition ....................................... 15

4.5.2 CSN Representation .................................... 15

4.5.3 CSN Generation ........................................ 16

4.6 STATE CHANGE INFORMATION ............................. 16

4.6.1 Entry Change State Storage and Representation .... 17

4.6.2 Attribute Change State Storage ....................... 18

4.6.3 Attribute Value Change State Storage ................ 18

4.7 LDAP UPDATE OPERATIONS ................................. 18

5. INFORMATION MODEL .......................................... 19

5.1 ENTRIES, SEMANTICS AND RELATIONSHIPS ............. 19

5.2 ROOT DSE ATTRIBUTES ..................................... 20

5.3 REPLICATION CONTEXT AUXILIARY OBJECT CLASS AND ENTRIES ...... 21
5.4 REPLICA OBJECT CLASS AND ENTRIES ..............................21
5.5 LOST AND FOUND ENTRY ...........................................21
5.6 REPLICATION AGREEMENT OBJECT CLASS AND ENTRIES ..............22
  5.6.1 Replication Schedule ........................................23
6.  REPLICATION OF DIRECTORY ADMINISTRATIVE POLICY INFORMATION ....23
  6.1 SCHEMA KNOWLEDGE ............................................24
7.  LDUP UPDATE TRANSFER PROTOCOL FRAMEWORK ..........................25
  7.1 REPLICATION SESSION INITIATION ................................25
    7.1.1 Authentication ............................................25
    7.1.2 Consumer Initiated .......................................26
    7.1.3 Supplier Initiated ........................................26
  7.2 START REPLICATION SESSION ....................................26
    7.2.1 Start Replication Request .................................26
    7.2.2 Start Replication Response ................................26
  7.3 UPDATE TRANSFER ................................................27
  7.4 END REPLICATION SESSION .......................................27
  7.5 INTEGRITY & CONFIDENTIALITY ...................................27
8.  LDUP UPDATE PROTOCOLS .............................................28
  8.1 REPLICATION UPDATES AND UPDATE PRIMITIVES .....................28
  8.2 FRACTIONAL UPDATES ............................................28
9.  LDUP FULL UPDATE TRANSFER PROTOCOL ................................29
  9.1 FULL UPDATE TRANSFER ..........................................29
  9.2 REPLICATION UPDATE GENERATION ................................29
  9.3 REPLICATION UPDATE CONSUMPTION ................................29
  9.4 FULL UPDATE, END REPLICATION SESSION ............................29
  9.5 INTERRUPTED TRANSMISSION ......................................30
10. LDUP INCREMENTAL UPDATE TRANSFER PROTOCOL ..........................30
  10.1 UPDATE VECTOR ..................................................30
  10.2 SUPPLIER INITIATED, INCREMENTAL UPDATE, START REPLICATION
       SESSION ..........................................................32
  10.3 REPLICATION UPDATE GENERATION ................................32
    10.3.1 Replication Log Implementation ..........................32
    10.3.2 State-Based Implementation ..............................32
  10.4 REPLICATION UPDATE CONSUMPTION ................................33
  10.5 UPDATE RESOLUTION PROCEDURES ...................................33
    10.5.1 URP: Distinguished Names .................................34
    10.5.2 URP: Orphaned Entries ....................................34
    10.5.3 URP: Schema - Single Valued Attributes ..................34
3. Introduction

3.1 Scope

This architectural document provides an outline of an LDAP based replication scheme. Further detailed design documents will draw guidance from here.

The design proceeds from prior work in the industry, including concepts from the ITU-T Recommendation X.525 (1993, 1997) Directory Information Shadowing Protocol (DISP) [X525], experience with widely deployed distributed directories in network operating systems, electronic mail address books, and other database technologies. The emphasis of the design is on:

1. Simplicity of operation.
2. Flexibility of configuration.
3. Manageability of replica operations among mixed heterogeneous vendor LDAP servers under common administration.
4. Security of content and configuration information when LDAP servers from more than one administrative authority are interconnected.

A range of deployment scenarios is supported, including multi-master and single-master topologies. Replication networks may include transitive and redundant relationships between LDAP servers.

The controlling framework used to define the relationships, types, and state of replicas of the directory content is defined. In this way the directory content can itself be used to monitor and control the replication network. The directory schema is extended to define object classes, auxiliary classes, and attributes that describe areas of the namespace which are replicated, LDAP servers which hold replicas of various types for the various partitions (_Replication Contexts_) of the namespace, LDAP Access Points (network addresses) where such LDAP servers may be contacted, which namespace replicas are held on given LDAP servers, and the progress of replication operations. Among other things, this knowledge of where directory content is
located could serve as the basis for dynamic generation of LDAP referrals.

An update transfer protocol, which actually brings a replica up to date with respect to changes in directory content at another replica, is defined using LDAPv3 protocol extensions. The representation of directory content and changes will be defined by the LDAP Replication Update Transfer Protocol sub-team. Incremental and full update transfer mechanisms are described. Replication protocols are required to include initial population, change updates, and removal of directory content.

Security information, including access control policy will be treated as directory content by the replication protocols. Confidentiality and integrity of replication information is required to be provided by lower-level transport/session protocols such as IPSEC and/or TLS.

3.2 Document Objectives

The objectives of this document are:

a) To define the architectural foundations for LDAP Replication, so that further detailed design documents may be written. For instance, the Information Model, Update Transfer Protocol, and Update Resolution Procedures documents.

b) To provide an architectural solution for each clause of the requirements document [LDUP Requirements].

c) To preserve the LDAP Data Model and Operational Behavior constraints defined for LDAP in RFC 2251 [See Appendix A].

d) To avoid tying the LDUP working group to the schedule of any other working group.

e) Not to infringe upon known registered intellectual property rights.

3.3 Document Non-Objectives

This document does not address the following issues, as they are considered beyond the scope of the Working Group.
a) How LDAP becomes a distributed directory. There are many issues beyond replication that should be considered. Such as, support for external references, algorithms for computing referrals from the distributed directory knowledge, etc.

b) Specifying management protocols to create Replication Contexts or new Replicas. LDAP may be sufficient for this. The document describes how new Replication Contexts and Replicas are represented, in the directory, as entries, attributes, and attribute values.

c) How transactions will be replicated. However, the architecture should not knowingly prevent or impede them, given the Working Group’s incomplete understanding of the issues at this time.

d) The mapping or merging of disparate Schema definitions.

e) Support of overlapping replicated regions.

f) The case where separate attributes of an entry may be mastered by different LDAP servers. This might be termed a ‘Split Primary’. Replica roles are defined in section 4.1.

3.4 Existing Implementations

In order to define a standard replication scheme that may be readily implemented we must consider the architectures of current LDAP server implementations. Existing systems currently support proprietary replication schemes based on one of two general approaches: log-based or state-based. Some sections of this text may specifically address the concerns of one approach. They will be clearly marked.
3.4.1 Replication Log Implementations

Implementations based on the original University of Michigan LDAP server code record LDAP operations to a operation log. During a replication session operations are replayed from this log to bring the Consumer replica up to date. Example implementations of this type at this time are the Innosoft, Netscape, and Open LDAP Directory Servers.

3.4.2 State-Based Implementations

Directory Server implementations from Novell and Microsoft at this time do not replay LDAP operations from a operation log. When a replication session occurs each entry in the Replicated Area is considered in turn, compared against the update state of the Consumer, and any resultant changes transmitted. These changes are a set of assertions about the presence or absence of entries, attributes, and their values.

3.5 Terms and Definitions

The definitions from the Replication Requirements document have been copied here and extended.

For brevity, an LDAP server implementation is referred to throughout as 'the server'.

The LDAP update operations; Add, Delete, Modify, Modify RDN (LDAPv2) and Modify DN (LDAPv3), are collectively referred to as LDAP Update Operations.

A Naming Context is a subtree of entries in the Directory Information Tree (DIT). There may be multiple Naming Contexts stored on a single server. Naming Contexts are defined in section 17 of [X501].

A _Replication Context_ represents a section of DIT defining a unit of administration for replication. A Replication Context is based at an entry identified as its root and includes all its subordinate entries down the tree to its leaves, or until another Replication Context is encountered. A Naming Context held by a server may be made up of one or more non-overlapping Replication Contexts. Non-replicated portions of a Naming Context may not be explicitly identified as a Replication Context, although implicitly
they are part of a primary Replica with no Replication Agreements.

A Replica is a replicated instance of a _Replication Context_.

A _Replication Context_ is said to be single-mastered if there is only one Replica where it may be updated, and multi-mastered if there is more than one Replica where it may be updated.

A Replication Relationship is established between two or more Replicas that are hosted on servers that cooperate to service a common area (the Replication Context) of the DIT.

A Replication Agreement is defined between two parties of a Replication Relationship. A Replication Agreement is associated with a set of replicas and defines properties such as the Update Transfer Protocol to be used, and the Replication Schedule of a Replication Session.

A Replication Session is an LDAP session between the two servers identified by a replication agreement. Interactions occur between the two servers, resulting in the transfer of updates from the supplier replica to the consumer replica.

The Initiator of a Replication Session is the initiating server.

A Responder server responds to the replication initiation request from the Initiator server.

A Supplier server is the source of the updates to be transferred.

A Consumer server is the recipient of the update sequence.

The Update Transfer Protocol is the means by which the Replication Session proceeds. It defines the protocol for exchanging updates between the Replication Relationship partners.

A Replication Update is an LDAP Extended Operation that contains updates to be applied to the DIT. The Update Transfer Protocol carries a sequence of these messages from the Supplier to the Consumer.
The Update Resolution Procedures repair constraint violations that occur when updates to a multi-mastered Replica collide.

A Fractional Entry Specification is a list of entry attributes to be included, or a list of attributes to be excluded in a replica. An empty specification implies that all entry attributes are included.

A Fractional Entry is an entry that contains only a subset of its original attributes. It results from the replication of changes governed by a Fractional Entry Specification.

A Fractional Replica is a replica that holds Fractional Entries of its Replication Context.

### 3.6 Consistency Models

This replication architecture supports a loose consistency model between replicas of a naming context. It does not attempt to provide the appearance of a single copy of a replica. The contents of each replica may be different, but over time they will be converging towards the same state. This architecture is not intended to support LDAP Clients that require a tight consistency model, where the state of all replicas is always equivalent.

Three levels of consistency are available to LDAP Clients, which are characterized by their deployment topologies. Single-Server, where there is just the Replication Context and no replicas. Single-master, where there are replicas, but only one may be updated. And, multi-master, where there is more than one replica to which LDAP update operations may be directed. The consistency properties of each model are rooted in their serialization of read and write operations.

1) A single-server deployment of a Replication Context provides tight consistency to LDAP applications. LDAP Clients have no choice but to direct all their operations to a single server, serializing both read and write operations.

2) A single-mastered deployment of a Replication Context provides both tight and loose consistency to LDAP applications. LDAP Clients must direct all write operations to the single updateable replica, but may direct their reads to any of the replicas. A client experiences tight consistency by directing all its operations to the single
updateable replica, and loose consistency by directing any read operations to any other replica.

3) A multi-mastered deployment of a Replication Context can provide only loose consistency to LDAP applications. Across the system writes and reads are not serialized. An LDAP Client could direct their read and write operations to a single updateable replica, but they will not receive tight consistency as interleaved writes could be occurring at another replica.

Tight consistency can be achieved in a multi-master deployment for a particular LDAP application if and only if all instances of its client are directed towards the same updateable replica, and the application data is not updated by any other LDAP application. Introducing these constraints to an application ensures that writes are serialized providing tight consistency for the application.

Future work could make use of the architecture proposed in this document as a basis for allowing clients to request session guarantees from a server when establishing a connection.

3.7 LDAP Constraints

The LDAP-v3 Internet RFC [LDAPv3] defines a set of Data Model and Operation Behavior constraints that a compliant LDAP server must enforce. The server must reject an LDAP Update Operation if its application to the target entry would violate any one of these LDAP Constraints. [Appendix A contains the original text clauses from RFC 2251, and also a summary.]

In the case of a single-server or single-mastered Replication Context all LDAP Constraints are immediately enforced at the single updateable replica. An error result code is returned to an LDAP Client that presents an operation that would violate the constraints.

In the case of a multi-mastered Replication Context not all LDAP Constraints can be immediately enforced at the updateable replica to which the LDAP Update Operation is applied. This loosely consistent replication architecture ensures that at each replica all constraints are imposed, but as updates are replicated constraint violations arise that cannot be reported to the appropriate client. Any
constraint violations that occur are repaired by a set of update resolution procedures.

Any LDAP client that has been implemented to expect immediate enforcement of all LDAP Constraints may not behave as expected against a multi-mastered Replication Context.

4. Directory Model

This section describes extensions to the LDAP Directory Model that are required by this replication architecture.

4.1 Replica Type

Each Replica is characterized with a replica type. This may be Primary, Updateable, or Read-Only. A Read-Only Replica may be further defined as being Fractional.

4.1.1 Primary Replica

The Primary Replica is a full copy of the Replica, to which all applications that require tight consistency should direct their LDAP Operations. There can be only one Primary Replica within the set of Replicas of a given Replication Context. It is also permissible for none of the Replicas to be designated the Primary. The Primary Replica MUST NOT be a Fractional Replica.

4.1.2 Updateable Replica

An Updateable Replica is a Replica that accepts all the LDAP Update Operations, but is not the Primary Replica. There could be none, one, or many Updateable Replicas within the set of Replicas of a given Replication Context. An Updateable Replica MUST NOT be a Fractional Replica.

4.1.3 Read-Only Replica

A Read-Only Replica will accept only non-modifying LDAP operations. All modification operations shall be referred to an updateable Replica. The server referred to would usually be a Supplier of this Replica.
4.1.4 Fractional Replicas

Fractional Replicas must always be Read-Only. All LDAP Update Operations must be referred to an Updateable Replica. The server referred to would usually be a Supplier of this Fractional Replica.

4.2 Sub-Entries

Replication management entries are to be stored at the base of the Replication Context. They will be of a ‘ldapSubentry’ objectclass to exclude them from regular searches. Entries with the objectclass ldapSubentry are not returned as the result of a search unless a control is included in the request to make them visible, or the filter component "(objectclass='ldapSubentry')" is included in the search filter.

4.3 Glue Entries

A glue entry is an entry that contains knowledge of its name only. No other information is held with it. Such glue entries will be distinguished through a special object class defined for that purpose. Glue entries may be created during a replication session to repair a constraint violation.

4.4 Unique Identifiers

Distinguished names can change, so are therefore unreliable as identifiers. A Unique Identifier must therefore be assigned to each entry as it is created. This identifier will be stored as an operational attribute of the entry, named 'entryUUID'. The entryUUID attribute is single valued. A consistent algorithm for generating such unique identifiers should be defined for use in the LDUP standards documents that detail the LDUP information model and LDUP protocols.

4.5 Change Sequence Number

Change Sequence Numbers (CSNs) are used to impose a total ordering upon the causal sequence of updates applied to all the replicas of a Replication Context. Every LDAP Update Operation is assigned at least one CSN. A Modify operation MUST be assigned one CSN per modification.
4.5.1 CSN Composition

A CSN is formed of four components. In order of significance they are; the time, a change count, a Replica Identifier, and a modification number. The CSN is composed thus to ensure the uniqueness of every generated CSN. When CSNs are compared to determine their ordering they are compared component by component: first the time, then the change count, then the replica identifier, and finally the modification number.

The time component is a year-2000-safe representation of the real world time, with a granularity of one second.

Because many LDAP Update Operations, at a single replica, may be applied to the same data in a single second, the change count component of the CSN is provided to further order the changes. Each replica maintains a count of LDAP update operations applied against it. It is reset to zero at the start of each second, and is monotonically increasing within that second, incremented for each and every update operation. Should LDAP Update Operations occur at different replicas, to the same data, within the same single second, and happen to be assigned the same change count number, then the Replica Identifier is used to further order the changes.

The Replica Identifier is the value of the RDN attribute on the Replica Subentry that represents the Replica. The Replica Identifier could be assigned programmatically or administratively, in either case short values are advised to minimize resource usage. The IA5CaseIgnoreString syntax is used to compare and order Replica Identifier values.

The fourth and final CSN component, the modification number, is used for ordering the modifications within an LDAP Modify operation.

4.5.2 CSN Representation

The preferred CSN representation is: yyyy mm dd hh:mi:ssz # 0xSSSS # replica id # 0xssss

The ‘z’ in the time stipulates that the time is expressed in GMT without any daylight savings time offsets permitted, and the 0xssss represents the hexadecimal representation of an unsigned integer. Implementations must support 16 bit change counts and should support longer ones (32, 64, or 128 bits).
An example CSN would be "1998081018:44:31z#0x000F#1#0x0000". The update assigned this CSN would have been applied at time 1998081018:44:31z happened to be the 16th operation which was applied in that second, was made against the replica with identifier '1', and was the first modification of the operation that caused the change.

4.5.3 CSN Generation

Because Change Sequence Numbers are primarily based on timestamps, clock differences between servers can cause unexpected change ordering. The synchronization of server clocks is not required, though it is preferable that clocks are accurate. If timestamps are not accurate, and a server consistently produces timestamps that are significantly older than those of other servers, its updates will not have effect and the real world time ordering of updates will not be maintained.

However, an implementation may choose to require clock synchronization. The Network Time Protocol [NTP] [SNTP] offers a protocol means by which heterogeneous server hosts may be time synchronized.

The modifications that made up an LDAP Modify operation are presented in a sequence. This must be preserved when the resultant changes of this operation are replicated.

4.5.3.1 CSN Generation _ Log Based Implementation

The modification number component may not be required, since the ordering of the modifications within an LDAP Modify operation have been preserved in the operation log.

4.5.3.2 CSN Generation _ State Based Implementation

The modification number component may be needed to ensure that the order of the modifications within an LDAP Modify operation is faithfully replicated.

4.6 State Change Information

State changes can be introduced via either LDAP Update Operations or via Replication Updates. A CSN is included
with all changes made to an entry, its attributes, and attribute values. This state information must be recorded for the entry to enable a total ordering of updates. The CSN recorded is the CSN assigned to the state change at the server where the state change was first made. CSNs are only assigned to state changes that originate from LDAP Update Operations.

Each of the LDAP Update Operations changes their target entry in different ways, and record the CSN of the change differently. The state information for the resultant state changes is recorded at three levels. The entry level, attribute level, and attribute value level. The state change may be shown through:

1) The creation of a deletion CSN for the entry, an attribute, or an attribute value.

2) In the addition of a new entry, attribute or attribute value, and its existence CSN.

3) An update to an existing attribute, attribute value, entry distinguished name, or entry superior name, and its update CSN.

4.6.1 Entry Change State Storage and Representation

When an entry is created, with the LDAP Add operation, the CSN of the change is added to the entry as the value of an operational attribute named ‘createdEntryCSN’, of syntax type LDAPChangeSequenceNumber.

\[
\text{createdEntryCSN} ::= \text{csn}
\]

Deleted entries are marked as deleted by the addition of the object class ‘deletedEntry’. The attribute ‘deletedEntryCSN’, of syntax type LDAP Change Sequence Number, is added to record where and when the entry was deleted. Deleted entries are not visible to LDAP clients – they may not be read, they don’t appear in lists or search results, and they may not be changed once deleted. Names of deleted entries are available for reuse by new entries immediately after the deleted entry is so marked. It may be desirable to allow deleted entries to be accessed and manipulated by management and data recovery applications, but that is outside the scope of this document.
deletedEntryCSN ::= csn

A CSN is recorded for both the RDN, and the Superior DN of the entry.

4.6.2 Attribute Change State Storage

When all values of an attribute have been deleted, the attribute is marked as deleted and the CSN of the deletion is recorded. The deleted state and CSN are stored by the server, but have no representation on the entry, and may not be the subject of a search operation. This state information must be stored to enable the Update Resolution Procedures to be performed. It may be desirable to allow the deleted state and CNS information to be accessed and manipulated by management and data recovery applications, but that is outside the scope of this document.

4.6.3 Attribute Value Change State Storage

The Modification CSN for each value is to be set by the server when it accepts a modification request to the value, or when a new value with a later Modification CSN is received via Replication. The modified value and the Modification CSN changes are required to be atomic, so that the value and its Modification CSN cannot be out of synch on a given server. The server stores the state information, but it has no representation on the entry, and may not be the subject of a search operation. It may be desirable to allow the state information to be accessed and manipulated by management and data recovery applications, but that is outside the scope of this document.

When the value of an attribute is deleted the state of its deletion must be recorded, with the CSN of the modifying change. It must be stored to enable the Update Resolution Procedures to be performed.

4.7 LDAP Update Operations

The server must reject LDAP client update operations with a CSN that is older than the state information that would be replaced if the operation were performed. This could occur in a replication topology where the difference between the clocks of updateable replicas was too large.
5. Information Model

This section describes the object classes of the entries that represent the replication topology. The operational information for replication is administered through these entries. The LDUP Working Group will work towards defining an Internet standard to fully detail all these schema elements.

5.1 Entries, Semantics and Relationships

This section defines the organization of operational data for directory replication in terms of the relative placement of the entries that represent Replication Contexts, its Replicas, and their associated Replication agreements. This section also describes the purpose of these objects and abstractly describes their content.
A Replication Context defines an area of DIT with independent replication policies. There are many mechanisms available to identify the set of Replication Contexts in a Directory, including through special auxiliary classes or through operational attributes in root DSE pointing to such entries. The LDUP information model standards will detail an appropriate mechanism.

Entries representing the set of Replicas associated with a Replication Context are created immediately below (children) the Replication Context entries. Replica entries are defined as subentries and are intended to hold attributes that identify the Replica’s LDAP Access Point, its Replica Type, and if it is a Fractional Replica, the attributes it does or does not hold. The attribute value of the entry’s Relative Distinguished Name (RDN) is termed the Replica Identifier and is used as a component of each CSN associated with the replica.

Immediately subordinate to each Replica Subentry are the entries representing the Replication Agreements between this replica and another replica on some other server in the network. A Replication Agreement entry is associated with exactly one remote replica. These entries are defined to hold attributes identifying the remote Replica associated with this agreement, the scheduling policy for replication operations, including times when replication is to be performed, when it is not to be performed, or the policies governing event-driven replication initiation another Replica, the scheduling policy for replication operations, including times when replication is to be performed, when it is not to be performed, or the policies governing event-driven replication initiation.

5.2 Root DSE Attributes

LDUP information model will define Root DSE attributes to identify the set of Replication Contexts and replicas present in an LDAP server.
5.3 Replication Context Auxiliary Object Class and Entries

Each Replication Context contains attributes that hold common configuration and policy information for all replicas of the Replication Context.

A Replication Context Creation attribute records when and where the Replication Context was created.

The Replication Context is based at the entry given the auxiliary class, and continues down the tree until leaf entries or another Replication Context is encountered.

5.4 Replica Object Class and Entries

A replica type characterizes each Replica. This may be Primary, Updateable, or Read-Only. The latter two types may be further defined as being Fractional. The Replica entry will include a Fractional Entry Specification for a Fractional Replica.

There is a need to represent network addresses of servers holding replicas involved in Replication Agreements. For this, the LDUP information model will define an attribute with an appropriate syntax to represent an LDAP server addresses with which to contact replicas.

An Update Vector describes the point to which the Replica has been updated, in respect to all the other Replicas of the Replication Context. The vector is used at the initiation of a replication session to determine the sequence of updates that should be transferred.

Enabling LDAP to be a fully distributed service is not an objective for the design of LDUP information model, though the information stored in replica entries could facilitate certain distributed operations.

5.5 Lost and Found Entry

When replicating operations between servers, conflicts may arise that cause a parent entry to be removed causing its child entries to become orphaned. In this case the Update
Resolution Procedures will make the Lost and Found Entry the child's new superior.

Each Replica Entry names its Lost and Found Entry, which would usually be an entry below the Replica Entry itself. This well-known place allows administrators, and their tools, to find and repair abandoned entries.

5.6 Replication Agreement Object Class and Entries

The Replication Agreement defines:

1. The schedule for Replication Sessions initiation.

2. The server that initiates the Replication Session, either the Consumer or the Supplier.

3. The authentication credentials that will be presented between servers.

4. The network/transport security scheme that will be employed in order to ensure data confidentiality.

5. The replication protocols and relevant protocol parameters to be used for Full and Incremental updates. An OID is used to identify the update transfer protocol, thus allowing for future extensions or bilaterally agreed upon alternatives.

6. If the Replica is Fractional, the Fractional Entry Specification, for the attributes to be included or excluded

Permission to participate in replication sessions will be controlled, at least in part, by the presence and content of replica agreements.

The Supplier must be subject to the access control policy enforced by the Consumer. Since the access control policy information is stored and replicated as directory content, the access control imposed on the Supplier by the Consumer must be stored in the Consumer’s Replication Agreement.
5.6.1 Replication Schedule

There are two broad mechanisms for initiating replication sessions: (1) scheduled event driven and (2) change event driven. The mechanism used to schedule replication operations between two servers is determined by the Schedule information that is part of the Replication Agreement governing the Replicas on those two servers. Because each Replication Agreement describes the policy for one direction of the relationship, it is possible that events propagate via scheduled events in one direction, and by change events in the other.

Change event driven replication sessions are, by their nature, initiated by suppliers of change information. The server that the change is made against schedules a replication session in response to the change itself, so that notification of the change is passed on to other Replicas.

Either consumers or suppliers of change information can initiate scheduled event driven replication sessions. The schedule defines a calendar of time periods during which Replication Sessions should be initiated.

Schedule information may include both scheduled and change event driven mechanisms. For instance, one such policy may be to begin replication within 15 seconds of any change event, or every 30 minutes if no change events are received.

6. Replication of Directory Administrative Policy Information

Administrative policy information governs the behavior of the directory server. Schema, access control, and replication all involve administrative policy information. This policy information (irrespective of how it is represented in the directory as sub-entries, attributes, or attribute values) should be consistently known and enforced by servers managing any replica. Normally, policy information present within a Replication Context is replicated in the same manner as any other directory information. But applicable policy information could reside outside a Replication Context.
Administrative policy information associated with directory replication lies within the replication context to which it applies. Hence, fortunately, any replica will also contain (include) all of its applicable replication policy data. On the other hand, some administrative boundaries (administrative areas) for other services might extend to subordinate Replication Contexts. For instance, some prescriptive access control policy applicable to entries in a Replication Context could be represented by an entry that is an ancestor of the root of the Replication Context. For access control policies to be faithfully enforced by a server hosting a replica of such a Replication Context, all applicable prescriptive policy information must also be available within that server. Following potential models are identified for accomplishing this:

(1) The administration model for specific services (such as access controls) could define mechanisms for representing and transmitting policy information as if it were an element of the root of the Replication Contexts contained within their respective administrative areas.

(2) Directory administrators could establish one or more explicit agreements to cover all applicable entries within superior Replication Contexts that hold administrative policy information.

(3) A replication agreement could implicitly include all administrative entries (and their associated sub-entries) that are ancestors of the Replication Context defined by these agreements. LDUP replication protocol could choose to define this mechanism as the standard, if its implementation is assessed to be relatively less complex.

6.1 Schema Knowledge

Given the strict ordering of replication events, schema modifications will normally be replicated prior to entry operations that use them, and subsequent to data deletions that eliminate references to schema elements to be deleted. In a multi-master environment with multiple suppliers, the order of arrival at a consumer node of such changes cannot be guaranteed. The LDUP standards for reconciliation should define procedures for handling such scenarios.
7. LDUP Update Transfer Protocol Framework

A Replication Session occurs between a Supplier server and Consumer server over an LDAP connection. This section describes the process by which a Replication Session is initiated, started and stopped.

The session initiator, termed the Initiator, could be either the Supplier or Consumer. The Initiator sends an LDAP extended operation to the Responder identifying the replication agreement being acted on. The Supplier then sends a sequence of updates to the Consumer.

All transfers are in one direction only. A two-way exchange requires two replication sessions - one session in each direction.

7.1 Replication Session Initiation

The Initiator starts the Replication Session by opening an LDAP connection to its Responder. The Initiator binds using the authentication credentials provided in the Replication Agreement. The LDUP Update Transfer Protocol will define the LDAP extended operation the Initiator should perform to initialize an LDUP session. For the sake of convenience, this extended LDAP operation for initializing a replication session is referred to as the _Start Replication_ operation. Among other things, this operation will identify the role each server will perform, and what type of replication is to be performed. One server is to be the Consumer, the other the Supplier, and the replication may be either Full or Incremental.

7.1.1 Authentication

The initiation of a Replication Session is to be restricted to privileged clients. The identity and the credentials for the client eligible for initiating a replication session will be specified as attributes within Replication Agreements.
7.1.2 Consumer Initiated

The Consumer binds to the Supplier using the authentication credentials specified in the Replication Agreement. The Consumer sends the Start Replication extended request to begin the Replication Session. The Supplier returns a Start Replication extended response containing a response code. The Consumer then disconnects from the Supplier. If the Supplier has agreed to the replication session initiation, it binds to the Consumer and behaves just as if the Supplier initiated the replication.

7.1.3 Supplier Initiated

The Supplier binds to the Consumer using the authentication credentials provided in the Replication Agreement. The Supplier sends the _Start Replication_ extended request to begin the Replication Session. The Consumer returns a _Start Replication_ extended response containing a response code, and possibly its Update Vector. If the Consumer has agreed to the Replication Session initiation, then the transfer protocol begins.

7.2 Start Replication Session

7.2.1 Start Replication Request

The LDUP Update Transfer Protocol would define an LDAP Extended Request, referred to in this document as _Start Replication Request_, that is sent from the Initiator to Responder. The parameters of the _Start Replication Request_ would identify the Replication Agreement associated with the session, the Update Transfer Protocol associated with the replication session, and other state information necessary to initiate a replication session between the two servers.

7.2.2 Start Replication Response

The LDUP Update Transfer Protocol would define an LDAP Extended Response, _Start Replication Response_, sent in reply to a Start Replication Request, from the Responder to
the Initiator. The parameters of the Start Replication Response include a response code, and an optional Update Vector.

7.3 Update Transfer

Each Update Transfer Protocol is identified by an OID. An LDUP conformant server implementation must support those update protocols that are defined as mandatory in the Update Transfer Protocol standard, and may support many others. A server will advertise its protocols in the Root DSE multi-valued attribute ‘supportedReplicationProtocols’.

The Update Transfer Protocol would define the mechanisms for a Consumer to receive a complete (full) update or incremental update based on the current state of replication represented in the Update Vector. A full update is necessary for initializing a consumer replica upon establishment of replication agreements.

7.4 End Replication Session

The _End Replication Request_ initiated by the supplier terminates a Replication Session. The purpose of this request and response is to secure the state of the Update Vector associated with the two replicas that participated in replication. This is necessary for proper resumption of replication during subsequent LDUP sessions.

7.5 Integrity & Confidentiality

Data integrity (i.e., protection from unintended changes) and confidentiality (i.e., protection from unintended disclosure to eavesdroppers) SHOULD be provided by appropriate selection of underlying transports, for instance TLS, or IPSEC. Replication MUST be supported across TLS LDAP connections. Servers MAY be configured to refuse replication connections over unprotected TCP connections.
8. LDUP Update Protocols

This Internet-Draft defines two transfer protocols for the supplier to push changes to the consumer. Other protocols could be defined to transfer changes, including those that pull changes from the supplier to the consumer, but those are left for future work.

8.1 Replication Updates and Update Primitives

Both LDUP Update Protocols define how Replication Updates are transferred from the Supplier to the Consumer. Each Replication Update consists of a set of Update Primitives that describe the state changes that have been made to a single entry. Each Replication Update is associated with a single entry identified by its UUID.

The Update Transfer Protocol would define a set of Update Primitives each of which codifies an assertion about the state change of an entry that resulted from a directory update operation. The primitives will include sufficient data to allow recreation of corresponding state changes on the consumer’s replica. An assertion-based approach has been chosen in such a way that the Primitives are idempotent, meaning that re-application of a Primitive to an Entry will cause no change to the entry. This is desirable as it provides some resilience against some kinds of system failures.

Each Update Primitive contains a CSN that represents an ordering among all such primitives generated anywhere in the network. The consumer uses this ordering information to reconcile among those primitives that lead to consistency violation.

8.2 Fractional Updates

When fully populating or incrementally bringing up to date a Fractional Replica each of the Replication Updates must only contain updates to the attributes in the Fractional Entry Specification.
9. LDUP Full Update Transfer Protocol

9.1 Full Update Transfer

This Full Update Protocol provides a bulk transfer of the replica contents for the initial population of new replicas, and the refreshing of existing replicas. The LDUP Update Transfer protocol standard will define the ways for this transfer is initiated. The Consumer must replace its entire replica contents with that sent from the Supplier.

The Consumer MUST NOT service any requests for this Naming Context whilst the full update is in progress. The Consumer could instead return a referral to another replica, possibly the supplier. [REF]

9.2 Replication Update Generation

The entire state of a Replicated Area can be mapped onto a sequence of Replication Updates, each of which contains a sequence of Update Primitives that describe the entire state of a single entry.

The sequence of Replication Updates must be ordered such that no entry is created before its parent.

9.3 Replication Update Consumption

A Consumer will receive the Replication Updates, extract the sequence of Update Primitives, and must apply them to the DIB in the order provided.

9.4 Full Update, End Replication Session

A Full Update should also result in the replication of all appropriate LDUP meta-data (which are part of the Replication Context), such as the sub-entry representing the Replica being updated and the Update Vector associated with it. The Supplier could be accepting updates whilst the update is in progress. Once the Full Update has completed, an Incremental Update should be performed to transfer these changes.
9.5 Interrupted Transmission

If the Replication Session terminates before the End Replication Request is sent, then the Replica could be in an inconsistent state. Until the replica is restored to a consistent state, the consumer MUST NOT permit LDAP Clients to access the incomplete replica. The Consumer could refer the Client to the Supplier Replica, or return an error result code.

10. LDUP Incremental Update Transfer Protocol

For efficiency, the Incremental Update Protocol transmits only those changes that have been made to the Supplier replica that the Consumer has not already received. In a replication topology with transitive redundant replication agreements, changes may propagate through the replica network via different routes.

The Consumer must not support multiple concurrent replication sessions with more than one Supplier for the same Replication Context. A Supplier that attempts to initiate a Replication Session with a Consumer already participating as a Consumer in another Replication Session will receive the appropriate error..

10.1 Update Vector

The Supplier uses the Consumer’s Update Vector to determine the sequence of updates that should be sent to the Consumer.

Each Replica entry includes an Update Vector to record the point to which the replica has been updated. The vector is a set of CSN values, one value for each known updateable Replica. Each CSN value in the vector corresponds to the most recent change known locally that occurred in the updateable replica that this Update Vector value represents.

For example, consider two updateable replicas of a Replication Context, one is assigned replica identifier ‘1’, the other replica identifier ‘2’. Each is responsible for maintaining its own update vector, which will contain
two CSNs, one for each replica. So, if both replicas are identical they will have equivalent update vectors.

Both Update Vectors =

(1998081018:44:31z#0x000F#1#0x0000,
1998081018:51:20z#0x0001#2#0x0000)

Subsequently, at 7pm, an update is applied to replica ‘2’, so its update vector is updated.

Replica ‘1’ Update Vector =

(1998081018:44:31z#0x000F#1#0x0000,
1998081018:51:20z#0x0001#2#0x0000)

Replica ‘2’ Update Vector =

(1998081018:44:31z#0x000F#1#0x0000,
1998081019:00:00z#0x0000#2#0x0000)

Since the Update Vector records the state to which the replica has been updated, a supplier server, during Replication Session initiation, can determine the sequence of updates that should be sent to the consumer. From the example above no updates need to be sent from replica ‘1’ to replica ‘2’, but there is at least one update pending from replica ‘2’ to replica ‘1’.

Because the Update Vector embodies knowledge of updates made at all known replicas it supports replication topologies that include transitive and redundant connections between replicas. It ensures that changes are not transferred to a consumer multiple times even though redundant replication agreements may exist. It also ensures that updates are passed across the replication network between replicas that are not directly linked to each other.

It may be the case that a CSN for a given replica is absent, for one of two reasons.

1. CSNs for Read-Only replicas might be absent because no changes will have ever been applied to that Replica, so there are no changes to replicate.

2. CSNs for newly created replicas may be absent because no changes to that replica have yet been propagated.
An Update Vector might also contain a CSN for a replica that no longer exists. The replica may have been temporarily taken out of service, or may have been removed from the replication topology permanently. An implementation may choose to retire a CSN after some configurable time period.

10.2 Supplier Initiated, Incremental Update, Start Replication Session

The Consumer Responder must return its Update Vector to the Supplier Initiator. The Supplier uses this to determine the sequence of Replication Updates that need to be sent to the Consumer.

10.3 Replication Update Generation

The Supplier generates a sequence of Replication Updates to be sent to the consumer. To enforce LDAP Constraint 20.1.6, that the LDAP Modify must be applied atomically, each Replication Update must contain the entire sequence of Update Primitives for all the LDAP Operations for which the Replication Update contains Update Primitives. Stated less formally, for each primitive the update contains, it must also contain all the other primitives that came from the same operation.

10.3.1 Replication Log Implementation

A log-based implementation might take the approach of mapping LDAP Operations onto an equivalent sequence of Update Primitives. A systematic procedure for achieving this will be fully described in the standard document defining Update Reconciliation Procedures.

The Consumer Update Vector is used to determine the sequence of LDAP Operations in the operation log that the Consumer has not yet seen.

10.3.2 State-Based Implementation

A state-based implementation might consider each entry of the replica in turn using the Update Vector of the consumer to find all the state changes that need to be transferred. Each state change (entry, attribute, or value - creation,
deletion, or update) is mapped onto the equivalent Update Primitive. All the Update Primitives for a single entry might be collected into a single Replication Update. Consequently, it could contain the resultant primitives of many LDAP operations.

10.4 Replication Update Consumption

A Consumer will receive Replication Updates, extract the sequence of Update Primitives, and must apply them to the DIB in the order provided. LDAP Constraint 20.1.6 states that the modifications within an LDAP Modify operation must be applied in the sequence provided.

Those Update Primitives must be reconciled with the current replica contents and any previously received updates. In short, updates are compared to the state information associated with the item being operated on. If the change has a more recent CSN, then it is applied to the directory contents. If the change has an older CSN it is no longer relevant and its change must not be effected.

If the consumer acts as a supplier to other replicas then the updates are retained for forwarding.

10.5 Update Resolution Procedures

The LDAP Update Operations must abide by the constraints imposed by the LDAP Data Model and LDAP Operational Behavior, Appendix A. An operation that would violate at least one of these constraints is rejected with an error result code.

The loose consistency model of this replication architecture and its support for multiple updateable replicas of a Replication Context means that LDAP Update Operations could be valid at one replica, but not in another. At the time of acceptance, the accepting replica may not have received other updates that would cause a constraint to be violated, and the operation to be rejected.

Replication Updates must never be rejected because of a violation of an LDAP Constraint. If the result of applying the Replication Update causes a constraint violation to occur, then some remedial action must be taken to satisfy the constraint. These Update Resolution Procedures are
introduced here, and fully described in These Update Resolution Procedures are introduced here will be fully defined within LDUP Update Resolution Procedures.

10.5.1 URP: Distinguished Names

LDAP Constraints 20.1.1 and 20.1.10 ensure that each entry in the replicated area has a unique DN. A Replication Update could violate this constraint producing two entries, with different unique identifiers, but with the same DN. The resolution procedure is to rename the both entries so that its RDN includes its own unique identifier. This ensures that the DN of both the entries shall be unique.

10.5.2 URP: Orphaned Entries

LDAP Constraints 20.1.11 ensures that every entry must have a parent entry. A Replication Update could violate this constraint producing an entry with (as yet) no parent entry. The resolution procedure is to create a Glue Entry to take the place of the absent parent. The Glue Entry’s superior will be the Lost and Found Entry. This well-known place allows administrators and their tools (including subsequent Replication Sessions) to find and repair orphaned entries.

10.5.3 URP: Schema - Single Valued Attributes

LDAP Constraint 20.1.7 enforces the single-valued attribute schema restriction. A Replication Update could violate this constraint creating a multi-value single-valued attribute. The resolution procedure is to replace the earlier value of a single-valued attribute with the newer value. In this way the most recently added value will be retained, and the older one discarded.

10.5.4 URP: Schema - Required Attributes

LDAP Constraint 20.1.7 enforces the schema objectclass definitions on an entry. A Replication Update could violate this constraint creating an entry that does not have attribute values for required attributes. The resolution procedure is to ignore the schema violation and mark the entry as a glue entry for administrative repair or correction in a subsequent replication session.
10.5.5 URP: Schema - Extra Attributes

LDAP Constraint 20.1.3 and 20.1.7 enforces the schema objectclass definitions on an entry. A Replication Update could violate this constraint creating an entry that has attribute values not allowed by the objectclass values of the entry. The resolution procedure is to ignore the schema violation and mark the entry as a glue entry for administrative repair or correction in a subsequent replication session.

10.5.6 URP: Duplicate Attribute Values

LDAP Constraint 20.1.5 ensures that the values of an attribute constitute a set of unique values. A Replication Update could violate this constraint. The resolution procedure is to enforce this constraint, recording the most recently assigned CSN with the value.

10.5.7 URP: Ancestry Graph Cycle

LDAP Constraint 20.4.2.1 prevents against a cycle in the DIT. A Replication Update could violate this constraint causing an entry to become it’s own parent, or for it to appear even higher in it’s ancestry graph. The resolution procedure is to break the cycle by changing the parent of the entry closest to be the lost and found entry.

10.6 Incremental Update, End Replication Session

If the Supplier sent none of its own updates to the Consumer, then the Supplier’s CSN within the Supplier’s update vector should be updated with the earliest possible CSN that it could generate, to record the time of the last successful replication session. The Consumer will have received the Supplier’s Update Vector in the replica sub-entry it holds for the Supplier replica.

The Consumer’s resultant Update Vector CSN values will be at least as great as the Supplier’s Update Vector.

The Supplier may request that the Consumer return its resultant Update Vector so that the Supplier can update its replica sub-entry for the Consumer Replica. The Supplier requests this by setting a flag in the End Replication
Request. The default flag value is TRUE meaning the Consumer Update Vector must be returned.

10.7 Interrupted Transmission

If the Replication Session terminates before the End Replication Request is sent then the Consumer’s Update Vector may or may not be updated to reflect the updates received. The Start Replication request includes a Replication Update Ordering flag that states whether the updates were sent in CSN order per replica.

If updates are sent in CSN order per replica then it is possible to update the Consumer Update Vector to reflect that some portion of the updates to have been sent have been received and successfully applied. The next Incremental Replication Session will pick up where the failed session left off.

If updates are not sent in CSN order per replica then the Consumer Update cannot be updated. The next Incremental Replication Session will begin where the failed session began. Some updates will be replayed, but because the application of Replication Updates is idempotent they will not cause any state changes.

11. Purging State Information

The state information stored with each entry need not be stored indefinitely. A server implementation may choose to periodically, or continuously, remove state information that is no longer required. The mechanism is implementation-dependent, but to ensure interoperability between implementations, the state information must not be purged until all known replicas have received and acknowledged the change associated with a CSN. This is determined from the Purge Vector [11.1].

All the CSNs stored that are lower than the Purge Vector may be purged, because no changes with older CSNs can be replicated to this replica.
11.1 Purge Vector

The Purge Vector is an Update Vector constructed from the Update Vectors of all known replicas. Below the root of a Replication Context is one sub-entry for each known replica of that Replication Context. Each of those entries contains the last known update vector for that replica. The lowest CSN for each replica are taken from these update vectors to form the Purge Vector. The Purge Vector is used to determine when state information and updates need no longer be stored.

11.2 Purging Deleted Entries, Attributes, and Attribute Values

The following conditions must hold before an item can be deleted from the Directory Information Base.

1) The LDAP delete operation has been propagated to all replication agreement partners.

2) All the CSNs in other replica Update Vectors representing changes to be sent to the server holding the deleted entry have advanced beyond the CSN on the deletion (similarly for deleted attributes and attribute values).

3) The CSN generator of the other Replicas must have advanced beyond the deletion CSN of the deleted entry. Otherwise, it is possible for one of those Replicas to generate operations with CSNs earlier than the deleted entry.

12. Replication Configuration and Management

Replication management entries, such as replica or replication agreement entries can be altered on any updateable replica. These entries are implicitly included in the directory entries governed by any agreement associated with this Replication Context. As a result, all servers with a replica of a Replication Context will have access to information about all other replicas and associated agreements.

The deployment and maintenance of a replicated directory network involves the creation and management of all the replicas of a Replication Context and replication agreements.
among these replicas. This section outlines, through an example, the administrative actions necessary to create a new replica and establish replication agreements. Typically, administrative tools will guide the administrator and facilitate these actions. The objective of this example is to illustrate the architectural relationship among various replication related operational information.

A copy of an agreement should exist on both the supplier and consumer side for the replication update transfer protocol to be able to start. For this purpose, the root of the Replication Context, replica objects and the replication agreement objects are created first on one of the servers. A copy of these objects is then manually created on the second server associated with the agreement.

The scenario below starts with a server (named DSA1) that holds an updateable replica of a Replication Context, RC1. Procedures to establish an updateable replica of the Replication Context on a second server (DSA2) are outlined.

Note that when entries are created on two or more separate servers in the operations described below, they need to be created with the same entry UUIDs so that they don’t collide with one another when replication of their information actually occurs. This may be done through some administrative control that allows the entry UUID to be set by the create entry operation.

1. On DSA1: Add RC1’s context prefix to the value of Root DSE attribute ‘replicaRoot’.

2. On DSA1: Alter the ‘ObjectClass’ attribute of the root entry of RC1 to include the "replicationContext" auxiliary class.

3. On DSA1: Create a replica object, RC1-R1, (as a child of the root of RC1) to represent the replica on DSA1. The attributes include replica type (updateable, read-only etc.) and DSA1 access point information.

4. On DSA2: Add RC1’s context prefix to the value of Root DSE attribute ‘replicaRoot’.

5. On DSA2: Create a copy of the root entry of RC1 as a copy of the one in DSA1 (including the replicationContext auxiliary class)
6. On DSA2: Create a copy of the replica object RC1-R1

7. On DSA2: Create a second replica object, RC1-R2 (as a sibling of RC1-R1) to represent the replica on DSA2.

8. On DSA1: Create a copy of the replica object RC1-R2

9. On DSA1: Create a replication agreement object, RC1-R1-R2 to represent update transfer from RC1-R1 to RC1-R2. This object is a child of RC1-R1.

10. On DSA2: Create a copy of the replication agreement, RC1-R1-R2.

11. On DSA2: Create a replication agreement, RC1-R2-R1, to represent update transfer from RC1-R2 to RC1-R1. This object is a child of RC1-R2.

12. On DSA1: Create a copy of the replication agreement, RC1-R2-R1.

After these actions update transfer to satisfy either of the two agreements can commence.

If data already existed in one of the replicas, the update transfer protocol should perform a complete update of the data associated with the agreement before normal replication begins.

13. Time

The server assigns a CSN for every LDAP update operation it receives. Since the CSN is principally based on time, the CSN is susceptible to the Replica clocks drifting in relation to each other (either forwards or backwards).

The server must never assign a CSN older than or equal to the last CSN it assigned.

The server must reject update operations, from any source, which would result in setting a CSN on an entry or a value that is earlier than possible. The error code
serverClocksOutOfSync (72) should be returned if it is clear that the update is not simply an old one that should be silently ignored. In particular, additions or modifications with CSNs prior to those on the servers Purge Vector should be rejected.

14. Security Considerations

The preceding architecture discussion covers the server authentication, session confidentiality, and session integrity in sections 7.1.1 and 7.5.

The IETF draft "Authentication Methods" for LDAP, provides a detailed LDAP security discussion. Its introductory passage is paraphrased below. [AUTH]

A Replication Session can be protected with the following security mechanisms.

1) Authentication by means of the SASL mechanism set, possibly backed by the TLS credentials exchange mechanism,

2) Authorization by means of access control based on the Initiators authenticated identity,

3) Data integrity protection by means of the TLS protocol or data-integrity SASL mechanisms,

4) Protection against snooping by means of the TLS protocol or data-encrypting SASL mechanisms,

The configuration entries that represent Replication Agreements may contain authentication information. This information must never be replicated between replicas.

Updates to a multi-mastered entry may collide causing the Update Resolution Procedures [10.5] to reject or reverse one of the changes to the entry. The URP algorithms resolve conflicts by using the total ordering of updates imposed by the assignment of CSNs for every operation. As a consequence updates originating from system administrators have no priority over updates originating from regular system users.
15. Acknowledgements

This document is a product of the LDUP Working Group of the IETF. The contribution of its members is greatly appreciated.

16. References


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


17. Intellectual Property Notice

The IETF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on the IETF’s procedures with respect to rights in standards-track and standards-related documentation can be found in BCP-11. [BCP-11] Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF Secretariat.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to practice this standard. Please address the information to the IETF Executive Director.

18. Copyright Notice


This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be
19. Authors’ Address

John Merrells  
Netscape Communications, Inc.  
501 East Middlefield Road  
Mountain View  
CA 94043  

E-mail: merrells@netscape.com  
Phone: +1 650-937-5739  

Edwards E. Reed  
Reed-Matthews, Inc.  
1064 East 140 North  
Lindon  
UT 84042  

E-mail: eer@oncalldba.com  
Phone: +1 801-796-7065
20. Appendix A _ LDAP Constraints

20.1 LDAP Constraints Clauses

This is an enumeration of the Data Model and Operation Behaviour constraint clauses defined in RFC 2251. [LDAPv3]

1) Data Model - Entries have names: one or more attribute values from the entry form its relative distinguished name (RDN), which MUST be unique among all its siblings. (p5)

2) Data Model - Attributes of Entries - Each entry MUST have an objectClass attribute. (p6)

3) Data Model - Attributes of Entries - Servers MUST NOT permit clients to add attributes to an entry unless those attributes are permitted by the object class definitions. (p6)

4) Relationship to X.500 - This document defines LDAP in terms of X.500 as an X.500 access mechanism. An LDAP server MUST act in accordance with the X.500 (1993) series of ITU recommendations when providing the service. However, it is not required that an LDAP server make use of any X.500 protocols in providing this service, e.g. LDAP can be mapped onto any other directory system so long as the X.500 data and service model as used in LDAP is not violated in the LDAP interface. (p8)
5) Elements of Protocol - Common Elements - Attribute - Each attribute value is distinct in the set (no duplicates). (p14)

6) Elements of Protocol - Modify Operation - The entire list of entry modifications MUST be performed in the order they are listed, as a single atomic operation. (p33)

7) Elements of Protocol - Modify Operation - While individual modifications may violate the directory schema, the resulting entry after the entire list of modifications is performed MUST conform to the requirements of the directory schema. (p33)

8) Elements of Protocol - Modify Operation - The Modify Operation cannot be used to remove from an entry any of its distinguished values, those values which form the entry’s relative distinguished name. (p34)

9) Elements of Protocol - Add Operation - Clients MUST include distinguished values (those forming the entry’s own RDN) in this list, the objectClass attribute, and values of any mandatory attributes of the listed object classes. (p35)

10) Elements of Protocol - Add Operation - The entry named in the entry field of the AddRequest MUST NOT exist for the AddRequest to succeed. (p35)

11) Elements of Protocol - Add Operation - The parent of the entry to be added MUST exist. (p35)

12) Elements of Protocol - Delete Operation - ... only leaf entries (those with no subordinate entries) can be deleted with this operation. (p35)

13) Elements of Protocol - Modify DN Operation - If there was already an entry with that name [the new DN], the operation would fail. (p36)

14) Elements of Protocol - Modify DN Operation - The server may not perform the operation and return an error code if the setting of the deleteoldrdn parameter would cause a schema inconsistency in the entry. (p36)
20.2 LDAP Data Model Constraints

The LDAP Data Model Constraint clauses as written in RFC 2251 [LDAPv3] may be summarised as follows.

a) The parent of an entry must exist. (LDAP Constraint 11 & 12.)

b) The RDN of an entry is unique among all its siblings. (LDAP Constraint 1.)

c) The components of the RDN must appear as attribute values of the entry. (LDAP Constraint 8 & 9.)

d) An entry must have an objectclass attribute. (LDAP Constraint 2 & 9.)

e) An entry must conform to the schema constraints. (LDAP Constraint 3 & 7.)

f) Duplicate attribute values are not permitted. (LDAP Constraint 5.)

20.3 LDAP Operation Behaviour Constraints

The LDAP Operation Behaviour Constraint clauses as written in RFC 2251 [LDAPv3] may be summarised as follows.

A) The Add Operation will fail if an entry with the target DN already exists. (LDAP Constraint 10.)

B) The Add Operation will fail if the entry violates data constraints:

   a - The parent of the entry does not exist. (LDAP Constraint 11.)

   b - The entry already exists. (LDAP Constraint 10.)

   c - The entry RDN components appear as attribute values on the entry. (LDAP Constraint 9.)

   d - The entry has an objectclass attribute. (LDAP Constraint 9.)

   e - The entry conforms to the schema constraints. (LDAP Constraint 9.)
f - The entry has no duplicated attribute values. (LDAP Constraint 5.)

C) The modifications of a Modify Operation are applied in the order presented. (LDAP Constraint 6.)

D) The modifications of a Modify Operation are applied atomically. (LDAP Constraint 6.)

E) A Modify Operation will fail if it results in an entry that violates data constraints:
   
   c - If it attempts to remove distinguished attribute values. (LDAP Constraint 8.)
   
   d - If it removes the objectclass attribute. (LDAP Constraint 2.)
   
   e - If it violates the schema constraints. (LDAP Constraint 7.)
   
   f - If it creates duplicate attribute values. (LDAP Constraint 5.)

F) The Delete Operation will fail if it would result in a DIT that violates data constraints:
   
   a - The deleted entry must not have any children. (LDAP Constraint 12.)

G) The ModDN Operation will fail if it would result in a DIT or entry that violates data constraints:
   
   b - The new Superior entry must exist. (Derived LDAP Data Model Constraint A)
   
   c - An entry with the new DN must not already exist. (LDAP Constraint 13.)
   
   c - The new RDN components do not appear as attribute values on the entry. (LDAP Constraint 1.)
   
   d - If it removes the objectclass attribute. (LDAP Constraint 2.)
   
   e - It is permitted for the operation to result in an entry that violates the schema constraints. (LDAP Constraint 14.)
20.4 New LDAP Constraints

The introduction of support for multi-mastered entries, by the replication scheme presented in this document, necessitates the imposition of new constraints upon the Data Model and LDAP Operation Behaviour.

20.4.1 New LDAP Data Model Constraints

1) Each entry shall have a unique identifier generated by the UUID algorithm available through the ‘entryUUID’ operational attribute. The entryUUID attribute is single valued.

20.4.2 New LDAP Operation Behaviour Constraints

1) The LDAP Data Model Constraints do not prevent cycles in the ancestry graph. Existing constraints Data Model Constraint _20.4.1_ (a) and Operation Constraint _20.4.2_ (B) would prevent this in the single master case, but not in the presence of multiple masters.

2) The LDAP Data Model Constraints state that only the LDAP Modify Operation is atomic. All other LDAP Update Operations are also considered to be atomically applied to the DIB.