LDAP Replication Architecture

Copyright (C) The Internet Society (1998). All Rights Reserved.

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 obsoleted 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-00.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 22 September 1999.
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 5
3.1 Scope 5
3.2 Document Objectives 6
3.3 Document Non-Objectives 6
3.4 Terms and Definitions 7
4. Overview 8
4.1 Directory Model 8
4.2 Information Model 9
4.3 Policy Information 9
4.4 Update Transfer Protocol 9
4.5 Conflict Detection and Resolution 10
4.6 Replication Configuration and Management 10
4.7 Update Vector 10
4.8 Time 10
5. Directory Model 11
5.1 Replica Type 11
5.1.1 Primary Replica 11
5.1.2 Updatable Replica 11
5.1.3 Read-Only Replica 11
5.1.4 Sparse Replica 11
5.1.5 Fractional Replicas 12
5.1.6 Partial Replica 12
5.2 Sub-Entries 12
5.3 Unique Identifiers 12
5.4 LDAP Change Sequence Numbers 12
5.5 State Storage and Representation 13
5.5.1 Entry Change State Storage and Representation 14
5.5.2 Attribute Change State Storage and Representation 14
5.5.3 Attribute Value Change State Storage and Representation 14
5.6 LDAP Update Operations 15
5.7 Purging State Information 15
5.7.1 Purging Deleted Entries, Attributes, and Attribute Values 15
6. Information Model 16
6.1 Entries, Semantics & Relationships 16
6.1.1 Root DSE Attributes 16
6.1.2 Naming Context Auxiliary Object Class and Entries 16
6.1.3 Replica Object Class and Entries 17
6.1.4 Lost and Found Entry 17
6.1.5 Replication Agreement Object Class and Entries 17
6.1.5.1 Replication Schedule 18
6.1.6 Update Vector 19
7. Policy Information 20
INTERNET-DRAFT      LDAP Replication Architecture     April 22, 1998

7.1 Access Control 20
7.2 Schema Knowledge 21
8. Update Transfer Protocol 21
8.1 Replication Session Initiation 22
8.1.1 Authentication 22
8.1.2 Consumer Initiated 22
8.1.3 Supplier Initiated 22
8.2 Start Replication Session 23
8.2.1 Consumer Initiated, Start Replication Session 23
8.2.2 Supplier Initiated, Full Update, Start Replication Session 23
8.2.3 Supplier Initiated, Incremental Update, Start Replication Session 23
8.2.4 Start Replication Request 23
8.2.5 Start Replication Response 24
8.3 Update Transfer 25
8.3.1 Full Update Transfer 25
8.3.2 Incremental Update Transfer 26
8.3.2.1 Conflict Detection and Resolution 26
8.3.2.2 Update Conflict Detection and Resolution 27
8.3.2.3 Naming Conflict Detection and Resolution 27
8.3.2.4 Orphaned Entry Detection and Resolution 27
8.4 End Replication Session 28
8.4.1 Full Update, End Replication Session 28
8.4.2 Incremental Update, End Replication Session 28
8.4.3 End Replication Request 28
8.4.4 End Replication Response 29
8.5 Integrity & Confidentiality 30
9. Replication Configuration and Management 30
10. Time 32
11. Security Considerations 32
12. Acknowledgements 33
13. References 33
15. Copyright Notice 35
16. Authors’ Address 35
17. Appendix A - Open Issues 36
17.1 Configuration Entries 36
17.2 Management 36
17.3 Purging 36
17.4 Update Transfer: Errors, Recovery, Diagnostics and Repair 36
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 are 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 of the namespace, LDAP Access Points (network addresses) where such LDAP servers may be contacted, which namespaces are held on given LDAP servers, and the progress of replication operations. Among other things, this knowledge of where directory content is located will provide the basis for dynamic generation of LDAP referrals. [REF]

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 following list enumerates the objectives of this document.

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 Conflict Detection and Resolution documents.

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

c) To preserve the atomicity of LDAP operations. Updates to an entry, from multiple sources, will be combined such that the resultant entry is equivalent to a serial execution of the operations.

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

e) Not to infringe known registered intellectual property.

### 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 naming contexts or new replicas. LDAP may be sufficient for this. The document describes how new replicas and naming contexts 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 5.1.

3.4 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 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 Replica is an instance of a replicated Naming Context.

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

A Replication Agreement is defined between two parties of a Replication Relationship. The properties of the agreement codify the Unit of Replication, 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 order of events, and update exchange mechanism between the Replication Relationship partners.

An Entry Filter is an LDAP filter expression that describes the entries to be replicated.

A Sparse Replica contains a subset of the naming context entries, being modified by the Entry Filter criteria associated with the replica.

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 has been modified by a Fractional Entry Specification.

A Fractional Replica is a replica that holds Fractional Entries of its naming context. Note that a Fractional Replica can also be a sparse replica.

A Partial Replica is a Sparse and/or Fractional Replica.

4. Overview

This section provides an overview of the LDAP Replication architecture. It is broken down into Directory Model, Information Model, Policy Information, Update Transfer Protocol, Replication Configuration and Management, Update Vector, and Time. The remainder of the document discusses each in greater detail.

4.1 Directory Model

The basic directory model must be extended in a number of ways.

a) The Replication Management entries require a sub-entry object class to hide them from end-user clients.

b) A form of timestamp, a Change Sequence Number (CSN), must be recorded with every change to every entry. The change may be the
creation of a new entry, the modification of an existing entry, or
the deletion of an existing entry.

c) Server implementations may need to include a CSN purging feature to
control Directory Information Base (DIB) storage space.

4.2 Information Model

A hierarchy of entries is defined that describe a Naming Context, its
Replicas, and its Replication Agreements.

The Naming Context Auxiliary Class is added to container entries that
may have separately defined replication policy. [LDUP Info]

Immediately subordinate to a Naming Context entry are the Replica
Subentry container entries that identify its LDAP Access Point, its
Replica Type, if it is a Sparse Replica, the LDAP search filter
defining which entries it holds, 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 ID and
is used as a component of each CSN.

Immediately subordinate in the namespace to a Replica Subentry are
Replication Agreement leaf entries which each identify 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.

4.3 Policy Information

Administrative policy information needs to be consistently known and
applied by all replicas of a Naming Context. As such, the Naming
Context Auxiliary Class provides a convenient way to define attributes
which can communicate those policies among all replicas and users of
the directory.

4.4 Update Transfer Protocol

A Replication Session occurs between a Supplier server and Consumer
server over an LDAP connection. 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.

4.5 Conflict Detection and Resolution

As the Consumer receives updates from the Supplier it must check for conflicts and correct them. There are three problem cases that must be detected and resolved. Update conflicts, naming conflicts, and orphaned entries. Change Sequence Numbers are used to totally order the sequence of updates, thereby resolving any conflicting updates.

4.6 Replication Configuration and Management

The management entries described in the Information Model may be created, modified, and deleted by administrative clients to configure and manage the replication network. The administrative operations performed over LDAP are discussed further in section 9.

4.7 Update Vector

Each Replica holds an Update Vector that records the most recent change it has received for each of the other Replicas of this Naming Context. The vector is used at the initiation of a replication session to determine the sequence of updates that should be transferred.

4.8 Time

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 which 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 synchronisation. The Network Time Protocol [NTP] [SNTP] offers a protocol means by which heterogeneous server hosts may be time synchronised.
5. Directory Model

The following sections describe changes to the basic directory model that are required by the replication architecture.

5.1 Replica Type

Each Replica is characterized with a replica type. This may be Primary, Updatable, or Read-Only. The latter two types may be further defined as being Sparse, Fractional, or Partial.

5.1.1 Primary Replica

The Primary Replica is a full copy of the Replica, to which all applications that require strong consistency should direct their LDAP operations. There can be only one Primary Replica within the set of Replicas of a given Naming Context. It is also permissible for none of the Replicas to be designated the Primary. The Primary Replica must not be a Partial Replica.

5.1.2 Updatable Replica

An Updatable Replica is a Replica that accepts all LDAP operations, but is not the Primary Replica. There could be none, one, or many Updatable Replicas within the set of Replicas of a given Naming Context. An Updatable Replica must not be a Fractional Replica.

5.1.3 Read-Only Replica

A Read-Only Replica will accept only non-modifying LDAP operations. All modification operations shall be referred to an updateable Replica, usually to its supplier. A Read-Only Replica may be a Partial Replica.

5.1.4 Sparse Replica

Sparse Replicas can be Read-Only or Updatable. Update operations targeted at entries outside of the Entry Filter scope must be referred to another Updatable Replica. The server referred to would usually be a Supplier of this Sparse Replica.
5.1.5 Fractional Replicas

Fractional Replicas must always be Read-Only. All updates operations must be referred to an Updatable Replica. The server referred to would usually be a Supplier of this Fractional Replica.

5.1.6 Partial Replica

A Partial Replica is a Sparse and/or Fractional Replica, so the constraints that apply to Sparse and Fractional Replicas apply. Partial Replicas that are Fractional must always be Read-Only.

5.2 Sub-Entries

Replication management entries are to be stored at the base of the replicated naming context. They will be of a subentry objectclass to exclude them from regular searches. Entries with the objectclass subentry are not returned as the result of a search unless the filter component "(objectclass=subentry)" is included.

5.3 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. The unique identifier could be generated by a number of algorithms, so we propose that the first octet be a prefix to identify its type. The prefix zero is reserved to signify the UUID (Universally Unique IDentifier) format, also known as GUID (Globally Unique IDentifier) [UUID]. An example UUID would be, 58DA8D8F-9D6A-101B-AFC0-4210102A8DA7.

Support for alternative algorithms is provided so that future better unique identifier generation algorithms may be easily adopted. Implementations may also wish to impose some structure to their unique identifiers to ease implementation of Conflict Detection & Resolution.

5.4 LDAP Change Sequence Numbers

Every change, caused by an LDAP update operation, is assigned a sequence number. The Change Sequence Number (CSN) is formed of four components. In order of significance they are; the time, a change count, a Replica ID, and a modification number.
The time component is a year-2000-safe representation of the real world time, with a granularity of one second. Should LDAP update operations occur at different replicas, to the same data, within the same single second, then the change count is used to further order the changes.

Because LDAP update operations at a single replica may also occur 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 changes made against it, which is reset to zero at the start of each second, and is monotonically increasing within the second, incremented for each LDAP update operation applied to the replica. 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, then the Replica ID is used to further order the changes.

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

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

The preferred time syntax 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, 128).

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 update which occurred in that second, and was made against the replica with id ‘1’.

5.5 State Storage and Representation

All changes made to an entry, and its attributes, include the CSN assigned at the server where the change was first made. Each of the LDAP update operations Add, Modify, Modify RDN (LDAP v2), Modify DN (LDAP v3), and Delete change their target entry in different ways, and
record the CSN of the change differently. This state information must be stored for each entry to enable Conflict Detection and Resolution.

State information is recorded at three levels within each entry. At the entry level, attribute level, and attribute value level. Each is briefly described below.

5.5.1 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.

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

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.

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

5.5.2 Attribute Change State Storage and Representation

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 Conflict Detection and Resolution to be performed.

5.5.3 Attribute Value Change State Storage and Representation

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 sync on a given server. The state information is stored by the server, but it
has no representation on the entry, and may not be the subject of a search operation.

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

5.6 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. Result code 72, serverClocksOutOfSync, is returned to the client.

5.7 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 can be determined by constructing a Purge Vector.

The Purge Vector is an Update Vector constructed from the Update Vectors of all known replicas. Each replica has a sub-entry for each known replica stored below its naming 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.

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.

5.7.1 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 updates from all the other replicas with timestamps less than the timestamp on the deletion have been propagated to the server holding the deleted object (similarly for deleted attributes and attribute values).

3) The clocks 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 object.

6. Information Model

This section describes the object classes of the entries that represent the replication topology. The where, when and how of Naming Context replication is administered through these entries. The LDUP Working Group will publish an Internet Draft to fully detail all these schema elements. [LDUP Info]

6.1 Entries, Semantics & Relationships

6.1.1 Root DSE Attributes

The Root DSE attribute ‘replicaRoot’, publishes the names of the Replicas that are held on that server. Each value of the attribute is the Distinguished Name of the root entry of the Replicated Area.

6.1.2 Naming Context Auxiliary Object Class and Entries

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

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

The Access Control Policy OID attribute defines the syntax and semantics of Access Control Information for entries within the Naming Context.

The Naming Context is based at the entry given the auxiliary class, and continues down the tree until another Naming Context is encountered.
6.1.3 Replica Object Class and Entries

Each Replica is characterized by a replica type. This may be Primary, Updatable, or Read-Only. The latter two types may be further defined as being Sparse, Fractional, or Partial. The Replica entry will include an Entry Filter for a Sparse Replica, a Fractional Entry Specification for a Fractional Replica, and both an Entry Filter and Fractional Entry Specification for a Partial Replica.

There is a need to represent network addresses of servers holding replicas and participating in Replication Agreements. The X.501 Access Point syntax is not sufficient, in that it is tied specifically to OSI transports. Therefore, a new syntax will be defined for LDAP which serves the same purpose, but uses IETF-style address information. [LDUP Info]

An Update Vector (detailed below) describes the point to which the Replica has been updated, in respect to all the other Replicas of the Naming Context.

The intent is to enable distributed operations in LDAP with the replica information stored there, but not to complete the process of turning LDAP into a fully distributed service.

6.1.4 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 conflict resolution algorithm will make the Lost and Found Entry the child's new superior.

Each Replica Entry names it’s 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.

6.1.5 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 Sparse, the Entry Filter.

7. If the Replica is Fractional, the Fractional Entry Specification.

8. If the Replica is Partial, the Entry Filter and Fractional Entry Specification.

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.

6.1.5.1 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, which 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.

Scheduled event driven replication sessions can be initiated by either consumers or suppliers of change information. 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.

6.1.6 Update Vector

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 is the most recent change, made at that Replica, that has been replicated to this Replica.

For example, consider two updatable replicas of a naming context, one is assigned replica id ‘1’, the other replica id ‘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, 1998081018:51:20z#0x0001#2 \}

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, 1998081018:51:20z#0x0001#2 \}

Replica ‘2’ Update Vector =

\{ 1998081018:44:31z#0x000F#1, 1998081019:00:00z#0x0000#2 \}

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 an 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.

7. Policy Information

Policy information governs the behavior of the server. It may be represented in the DIT as sub-entries, attributes, and attribute values.

When replicating a naming context that is itself a subtree of another naming context, there may be policy information stored in its antecedent entries. The most common examples are prescriptive access control information and inherited schema definition. Implementations may also define other policy attributes, or sub-entries, that apply to a whole subtree. For a naming context to be faithfully reproduced, this generational information must also be replicated. In all cases the policy information is transmitted as if it were an element of the Replica root entry.

Policy information is always replicated in the same manner as any other entries, attributes, and attribute values.

7.1 Access Control

The Access Control Models supported by a server are identified by the ‘accessControlScheme’ multi-valued attribute of the Root DSE entry. Each model is assigned an OID so that Consumers and Suppliers can determine if their access control policy will be faithfully imposed when replicated.

An access control policy must be consistently applied by all servers holding replicas of the same Naming Context. Therefore, the Access Control Model OIDs are defined in the LDAP data dictionary and are used to ensure the policy is faithfully imposed.
Control Policy attribute is to be an operational attribute of the Naming Context Auxiliary Class. Thus, any consumer of the directory, and any server which would replicate a Naming Context, will know that an Access Control Policy is defined for the Naming Context, and by reference to the OID value of this attribute, know what policy mechanism to invoke to enforce that policy. Administrators are strongly cautioned against placing replicas of naming contexts on servers that cannot enforce the policy required by the Access Control Policy OID. Servers should refuse to accept replicas with policies they are unable to properly interpret.

7.2 Schema Knowledge

Schema subentries should be subordinate to the naming contexts to which they apply. Given our model, a single server may hold replicas of several naming contexts. It is therefore essential that schema should not be considered to be a server-wide policy, but rather to be scoped by the namespace to which it applies.

Schema modifications replicate in the same manner as other directory data. Given the strict ordering of replication events, schema modifications will naturally be replicated prior to entry creations which use them, and subsequent to data deletions which eliminate references to schema elements to be deleted. Servers should not replicate information about entries which are not defined in the schema. Servers should not replicate modifications to existing schema definitions for which there are existing entries and/or attributes which rely on the schema element.

Should a schema change cause an entry to be in violation of the new schema, it is recommended that the server preserve the entry for administrative repair. The server could add a known object class to make the entry valid and to mark the entry for maintenance.

8. Update Transfer Protocol

This section describes the process by which a Replication Session is established, how updates are transferred, and how a session is terminated.

Subject to Replication Agreements, either the supplier or the consumer server can initiate the replication session. This document only defines a transfer protocol for the supplier to push changes to the consumer. Other protocols could be defined to transfer changes,
including those which pull changes from the supplier to the consumer, but those are left for future work.

8.1 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 extended LDAP operation Start Replication is then sent by the Initiator to the Responder. This operation identifies which 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. If the Responder does not support the requested type of replication then an error is returned.

8.1.1 Authentication

The initiation of a Replication Session is to be restricted to only permitted clients. The identity and credentials of a connected server are determined via the bind operation. Access control on the Replication Agreement determines if the Replication Session may proceed. Otherwise, the insufficientAccessRights error is returned.

8.1.2 Consumer Initiated

The Consumer binds to the Supplier using the authentication credentials provided 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.

8.1.3 Supplier Initiated

The Supplier binds to the Consumer using the authentication credentials provided in the Replication Agreement. The Supplier sends the Start Replication Request 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.
8.2 Start Replication Session

8.2.1 Consumer Initiated, Start Replication Session

The Supplier should not return its Update Vector when the Consumer initiates a replication session.

8.2.2 Supplier Initiated, Full Update, Start Replication Session

The Consumer should not return its Update Vector when it is about to be initialised or reinitialised.

8.2.3 Supplier Initiated, Incremental Update, Start Replication Session

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

8.2.4 Start Replication Request

A client may request this operation by transmitting an LDAP PDU containing an LDAP ExtendedRequest, defined as follows. [LDAPv3]

\[
\text{ExtendedRequest ::= [APPLICATION 23] SEQUENCE {}
\]
\[
\text{requestName [0] LDAPOID,}
\]
\[
\text{requestValue [1] OCTET STRING }
\]
\[
}
\]

The requestName field must be set to the string "2.16.840.1.113730.3.5.3".

The requestValue field will contain as a value the DER-encoding of the following ASN.1 data type:

\[
\text{SEQUENCE {}
\]
\[
\text{namingContextDN LDAPDN,}
\]
\[
\text{replicaID OCTET STRING,}
\]
\[
\text{protocolOID LDAPOID}
\]
\[
}
\]

The parameters of the Start Replication Request are:
- The Distinguished Name of the entry at the root of the Naming Context.
- The Replica Identifier of the Initiator.
- The OID identifying the Update Transfer Protocol to be used.

From the DN and Replica ID the Responder can determine which Replication Agreement is being acted on. The Protocol OID identifies the Update Transfer Protocol that the Initiator wishes to use, this indicates whether the it is to be a Full or Incremental update.

### 8.2.5 Start Replication Response

If a server implements this extension, then when the request is made it will return an LDAP PDU containing an ExtendedResponse, defined as follows. [LDAPv3]

```
ExtendedResponse ::= [APPLICATION 24] SEQUENCE {
    COMPONENTS OF LDAPResult,
    responseName [10] LDAPOID OPTIONAL,
```

The responseName field must be set to the string "2.16.840.1.113730.3.5.4". The responseValue field, when present, will contain as a value the DER-encoding of the following ASN.1 data type:

```
SEQUENCE {
    startReplicationResult [0] startReplicationStatus,
    updateVector [1] SET OF OCTET STRING OPTIONAL,
}
```

```
startReplicationStatus ::= ENumerated {
    success (0), -- operation succeeded
    operationsError (1), -- server internal failure
    protocolError (2), -- protocol error
    insufficientAccessRights (50), -- refused operation
```
busy            (51), -- already being updated.
other           (80)
}

The startReplicationResult field indicates any error condition encountered in the processing of the operation.

In Supplier Initiated Replication the Consumer Responder responds with its Update Vector.

8.3 Update Transfer

Each Update Transfer Protocol is identified by an OID. A conformant server implementation must support the two update protocols defined here, and may support many others. A server will advertise its protocols in the Root DSE multi-valued attribute 'supportedReplicationProtocols'.

The two mandatory to implement protocols will be defined by the LDUP Working Group in another Internet Draft. One is to provide a Full Update for initialisation and re-initialisation of a replica, the other is to maintain that replica via an Incremental Update.

Each entry to be transferred is passed through the Entry Filter and Fractional Entry Specification.

Necessary extended operations will be defined to support efficient transfer of change information from supplier to consumer servers.

8.3.1 Full Update Transfer

This Full Update Protocol will provide a bulk transfer of the replica contents for the initial population of new replicas, and the refreshing of existing replicas.

Upon receiving a full update request the Consumer must replace any existing information in its replica with that sent from the Supplier.

The Consumer need not service any requests for this Naming Context whilst the full update is in progress. The Consumer could return a referral to another replica, possibly the supplier. [REF]
8.3.2 Incremental Update Transfer

For efficiency, the Incremental Update Protocol transmits only those changes that have been made to the Naming Context since updates were last transmitted, 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 Supplier uses the Consumer’s Update Vector to determine the sequence of updates that should be sent to the Consumer.

As the transmission of updates proceeds the Consumer may return the last CSN received as a form of committed acknowledgement. This provides an implicit restart value for the Supplier, should the connection be interrupted.

Each individual change may contain a sequence of entry modifications that must be preserved when transferred. The atomicity of the LDAP operation must be preserved when it is applied to the Consumer Replica.

Changes need not be transmitted in ascending change sequence number.

Each change transmitted includes the Unique Identifier of the subject entry and the CSN of the originating operation. This allows the Consumer to keep track of which changes it has received.

The Consumer must not support multiple concurrent replication sessions with more than one Supplier for the same Naming 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 busy error code.

8.3.2.1 Conflict Detection and Resolution

A consequence of permitting multiple updateable replicas within a replication topology is that conflicting update changes may occur. Updates are conflicting if they are concurrently accepted by different replicas, meaning that, at the time of acceptance, the accepting replica had not yet seen the other update.

This section briefly describes the types of change collisions that can occur. The LDUP Working Group will publish an Internet Draft fully defining the Conflict Detection and Resolution mechanism.
8.3.2.2 Update Conflict Detection and Resolution

A Consumer will receive replication changes from its various agreement partners. Those changes must be reconciled with the current replica contents and any previously received changes. In broad outline, received replication changes 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 replica has replication agreements where it acts as a supplier then the change is retained for forwarding at the appropriate time. If the change has an older CSN it is no longer relevant and is simply discarded.

8.3.2.3 Naming Conflict Detection and Resolution

Naming collisions may occur when updates are received by a server for a given named entry whose Unique Identifier is different from that of an already existing entry with the same distinguished name. This may occur because events from various replicas cannot be guaranteed to arrive in sequence, or because of conflicting data changes being entered at two or more different replicas.

Consider the following scenario:

The entry named "A" has a Unique Identifier of "3", and it exists on each of three replicas, X, Y, and Z. At replica X, entry "A" is deleted, and then another entry "A" is created with the same name, but with the Unique Identifier of "4". If at replica Y a modification to entry "A" is entered before the deletion and creation events from X are received, Y will attempt to replicate the modification to "A" to X. When received, X will note that the Unique Identifier of the entry "A" which is being modified is "3". Because the entry "A" with Unique Identifier "3" is marked as "deleted" on X, server X will simply ignore the modification, since it applies to a deleted object, and not to the entry currently defined with name "A", whose Unique Identifier is "4". Thus, the confusion over which entry was modified is resolved.

8.3.2.4 Orphaned Entry Detection and Resolution

An entry may also become an orphan, if its parent entry is removed. In this case the conflict may be resolved by making the orphan a child of the Lost and Found Entry. This well known place allows administrators and their tools to find and repair abandoned entries.
8.4 End Replication Session

A Replication Session is terminated by the Supplier by sending an End Replication LDAP extended request, see section 8.4.3. The purpose of the request and response operations is to carry the Update Vector from the Supplier to the Consumer in the Full Update case, and to convey the Update Vector from the Consumer to the Supplier in the Incremental Update case.

8.4.1 Full Update, End Replication Session

Since the Full Update also replicates the sub-entry that represents the Supplier Replica the Consumer will have received the Update Vector that represents the update state of the Consumer.

After a Full Update transfer the Supplier sends the Update Vector that reflects the update state of the full replica information sent. The Consumer records this as its Update Vector.

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.

8.4.2 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.

8.4.3 End Replication Request

A client may request this operation by transmitting an LDAP PDU containing an LDAP ExtendedRequest, defined as follows. [LDAPv3]
ExtendedRequest ::= \[APPLICATION 23\] SEQUENCE {
  
  requestName [0] LDAP OID,
  
  requestValue [1] OCTET STRING }

The requestName field must be set to the string "2.16.840.1.113730.3.5.5".

The requestValue field will contain as a value the DER-encoding of the following ASN.1 data type:

\[
\text{SEQUENCE} \{
  \text{returnUpdateVector BOOLEAN}
\}
\]

When the update has completed the Supplier sends this extended request to inform the Consumer that all updates have been sent, and to advise the Consumer if its Update Vector should be returned.

8.4.4 End Replication Response

If a server implements this extension, then when the request is made it will return an LDAP PDU containing an ExtendedResponse, defined as follows. [LDAPv3]

ExtendedResponse ::= \[APPLICATION 24\] SEQUENCE {
  
  COMPONENTS OF LDAPResult,
  
  responseName [10] LDAP OID OPTIONAL,
  

The responseName field must be set to the string "2.16.840.1.113730.3.5.6". The responseValue field, when present, will contain as a value the DER-encoding of the following ASN.1 data type:

\[
\text{SEQUENCE} \{
  \text{endReplicationResult [0] endReplicationStatus,}
  
  \text{updateVector [1] SET OF OCTET STRING OPTIONAL}
\}
\]
endReplicationStatus ::= ENUMERATED {
    success          (0), -- operation succeeded
    operationsError  (1), -- server internal failure
    protocolError    (2), -- protocol error
    other            (80)
}

The endReplicationResult field indicates any error condition encountered in the processing of the operation. If the returnUpdateVector flag in the request was set then the Consumer returns its Update Vector to the Supplier.

8.5 Integrity & Confidentiality

Data integrity (ie, protection from unintended changes) and confidentiality (ie, 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.

9. 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 naming context. As a result, all servers with a replica of a naming 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 naming 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 naming context, replica objects and the replication agreement objects are created first on one of the servers. A copy of these objects are 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 naming context NC1. Procedures to establish an updateable replica of the naming context on a second server (DSA2) are outlined.

On DSA1:

1) Add the context prefix for NC1 to the Root DSE attribute 'replicaRoot' if it does not already exist.

2) Alter the 'ObjectClass' attribute of the root entry of NC1 to include the "namingContext" auxiliary class.

3) Create a replica object, NC1R1, (as a child of the root of NC1) to represent the replica on DSA1. The attributes include replica type (updateable, read-only etc.) and DSA1 access point information.

4) Create a copy of the replica object NC1R2 (after it is created on DSA2)

5) Create a replication agreement, NC1R1-R2 to represent update transfer from NC1R1 to NC1R2. This object is a child of NC1R1.

On DSA2:

1) Add NC1’s context prefix to the Root DSE attribute ‘replicaRoot’.

2) Create a copy of the root entry of NC1 as a copy of the one in DSA1 (including the namingContext auxiliary class)

3) Create a copy of the replica object NC1R1

4) Create a second replica object, NC1R2 (as a sibling of NC1R1) to represent the replica on DSA2.

5) Create a copy of the replication agreement, NC1R1-R2

6) Create a replication agreement, NC1R2-R1, to represent update transfer from NC1R2 to NC1R1. This object is a sibling of NC1R1-R2.
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.

10. 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 which is earlier than the one that is there. The error code serverClocksOutOfSync (72) should be returned.

11. Security Considerations

The preceding architecture discussion covers the server authentication, session confidentiality, and session integrity in sections 8.1.1 and 8.5.

The internet 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.

12. Acknowledgements

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

13. References


[LDUP Info.] - E. Reed, "LDUP Replication Information Model", Internet Draft, draft-reed-ldup-infomod-00-1.txt, August 1998.


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


Merrells, Reed, Srinivasan  [Page 33]
 Expires 22 September 1999

INTERNET-DRAFT      LDAP Replication Architecture     April 22, 1998


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 implementors 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 which may cover technology that may be required to practice this standard. Please address the information to the IETF Executive Director.

Merrells, Reed, Srinivasan
Copyright Notice

Copyright (C) The Internet Society (1998). All Rights Reserved.

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 prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

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
Novell, Inc.
122 E 1700 S
Provo, UT 84606
E-mail: ed_reed@novell.com
Phone: +1 801-861-3320
Fax: +1 801-861-2220
Appendix A - Open Issues

17.1 Configuration Entries

There have been a number of proposals to keep the replication configuration entries out of the replicated area. The Replica and Replication Agreement entries would appear in some configuration area of the DIT that would not be replicated.

17.2 Management

Topics that need coverage:
1. Creation, destruction, and modification of Naming Contexts, Replicas, and Replication Agreements.
2. Promotion and demotion of Replicas from Read-Only to Updateable to Primary.
4. SONM redundant replication agreements.
5. Triggering a Full Update for a Replication Agreement.

17.3 Purging

The constraints that must hold to prevent premature purging of change and state information have been under discussion.

17.4 Update Transfer: Errors, Recovery, Diagnosis, and Repair

1. How should the protocol react to connection loss?
2. Should the Replication Agreement protocol parameters define the acknowledgement frequency?
3. Should keep-alive and progress feedback be provided for long operations. For instance replica preparation during Full Update for reinitialisation?