Internet-Draft
LDAP Duplication/Replication/Update
Protocols WG
Intended Category: Informational
Expires: March 2004

Richard V. Huber
Gerald F. Maziarski
AT&T Laboratories
Ryan D. Moats
Lemur Networks

September 2003

General Usage Profile for LDAPv3 Replication
draft-ietf-ldup-usage-profile-06.txt

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/lid-abstracts.txt.

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

Abstract

Support for replication in LDAP directory systems is often one of the key factors in the decision to deploy them. But replication brings design constraints along with its benefits.

We discuss some of the factors that should be taken into consideration when designing a replicated directory system. Both programming and architectural/operational concerns are addressed and both single- and multi-master directories are considered.

Table of Contents

1 Introduction....3
2 Meta-data Considerations....3
 2.1 Schema Considerations....3
 2.2 Replication Agreements....5
 2.3 Access Control....5
 2.4 Change Logs....6
3 Naming Considerations....6
4 Conflict Resolution Considerations....7
 4.1 Consistent Access after Changes....7
 4.2 Conflict Resolution in Single-Master Systems....8
 4.3 Problem Cases....8
  4.3.1 Atomicity....8
    4.3.1.1 Locking....8
    4.3.1.2 Partitioning....9
 4.4 General Principles....9
5 Failover Considerations....10
 5.1 Common Issues....10
 5.2 Single Master Issues....11
 5.3 Multi-Master Issues....12
6 Other Issues....12
 6.1 Locking....12
 6.2 Backup and Restore....13
7 Impact of Non-LDAP Changes/Constraints....13
 7.1 Changes Outside of LDAP....13
 7.2 Application Triggers....14
 7.3 Policy Conflicts Across Servers....14
8 Security Considerations....15
9 Acknowledgements....15
10 References....15
Authors' Addresses....16
Full Copyright Statement....16

1 Introduction

As applications come to rely on LDAP directories as part of their mission-critical infrastructure, the need for highly reliable and highly available LDAP systems will increase.

Distributed, replicated directories can increase reliability and reduce capacity problems. Nevertheless, applications which work well with a single, standalone directory may develop problems in a distributed environment unless both the applications and the environment are designed with data distribution as one of the criteria.

While the detailed design criteria will depend partly on whether the distributed directory is a single-master or multi-master system many concerns are common to both. This document flags some issues as being specific to either single-master or multi-master directories; unflagged issues pertain to both.

Any given class of directory applications (e.g. white pages, policy, authentication and authorization) is not inherently single- or multi-master. This choice will depend on the requirements (reliability/availability, update procedures, performance, etc.) of a specific instance of the application. Therefore, this document addresses general issues regarding the deployment of single- and multi-master directory systems. There may be future documents that address specific applications.

2 Meta-data Considerations

Any LDAP directory contains meta-data as well as the user data in the directory. A non-exhaustive list of meta-data includes descriptions of the data in the directory (e.g. schema), policies for use of the data (e.g. access controls), and configuration/status information (e.g. replication agreements).

This meta-data is stored in the directory itself, accessible as regular data or as operational attributes. Issues may arise when meta-data stored in the directory is replicated. However, not replicating meta-data may also be problematic.

This section examines some of the potential problems.

2.1 Schema Considerations

If the schema of one or more of the copies of a replica differs from the schema of the other replicas, then there is a possibility of schema mismatch when data is exchanged between them. The schema extensibility feature of LDAP nearly guarantees that replica groups

comprised of a heterogeneous mix of systems will not contain

homogeneous schema because of directory vendors' built-in extensions. A given directory may not use all of the elements of its schema, so schema differences do not always lead to schema mismatches during replication.

Schema mismatch issues are further complicated by the possibility of replicating the "subschemaSubentry" itself. Some directories use this technique to distribute schema changes. Currently there is no standard for LDAP schema representation within the subschemaSubentry. In the absence of such a standard, full schema interoperability is not possible in the IETF sense. Directory designers should establish common schema on all servers holding a common replica, and should avoid use of vendor-specific attributes.

The following is a partial list of possible schema mismatches:

1. Object class not defined
2. Structure Rule of an object class
3. Structural vs. Auxiliary in an object class
4. Optional vs. Mandatory attribute in an object class
5. Object identifiers differ on an attribute type or on an object class
6. Type and number of attributes defined in a class
7. Attribute type not defined
8. Base syntax of an attribute type
9. Multi-valued vs. single-valued attribute types
10. Matching rule of an attribute type
11. Naming collisions of attribute type names
12. Attribute name aliasing ("street" vs. "streetAddress" vs. "Strasse")
13. ACL format (and consequently, ACL calculus)

Schema mismatches that cause data corruption in one or more of the replicas must result in meta-data (e.g. log entries) in order to comply with Requirement P7 of [RFC3384]. Note that schema differences do not produce corruption in all circumstances. Some schema differences may have little or no impact on the proper storage of replicated data. However, any time data is added to the directory, replication between heterogeneous schemas may result in data corruption due to a schema mismatch.

Options for dealing with such potential mismatches include:

- Use fractional replication to replicate only those attributes that do not have differences
- Remove all schema mismatches
- Use the same schema on all systems

The tool described by requirement AM8 of [RFC3384] would help

designers detect schema conflicts as early as possible.

2.1.1.1 Replication Agreements

Replication Agreements are central to replication, as they allow configuration of most of the aspects of the replication process, including the triggers for replica cycles (from Requirement M1 in [RFC3384]), critical OID information (from Requirement M6 in [RFC3384]), and replication process parameters (Requirement M7 in [RFC3384]). Through the use of a standard replication agreement schema (Requirement SC2 of [RFC3384], [InfoMod]) it is possible to replicate the replication agreement.

If a replication agreement includes replication credentials, the agreement should be read and write protected in the directory, and transport of the replication agreement should be encrypted.

When replication agreements are themselves distributed via replication, they are subject to same "loose consistency" issues (due to replication delay and deferred conflict resolution) as other data. Even a temporary inconsistency among replication agreements may cause unbalanced replication and further inconsistency. As "multi-mastering" complicates "loose consistency" issues, avoidance of these issues by making all replication agreement changes through the same master (see Sections 4 and 5) is strongly advised.

2.2 Access Control

The following considerations complicate replication of Access Control Information:

- Access Control Information (ACI) is treated as though it were stored as attributes in the directory [RFC2820]
- LDAP [RFC2251] declares that changes resulting from a single LDAP MODIFY are atomic (but see caveats for multi-master replication in Sections 3 and 4)
- The ACI affecting a given entry may not be part of that entry (it could be part of a group entry or part of an ancestor of the entry in question)
- The ACI cannot always be changed atomically with associated data changes
- The interaction of replication and partitioning is still unclear (i.e. what happens when access control policy is inherited from an area of replication that is not held locally)

Do not leave windows where data is unprotected.

To reduce risk:

- In all environments, access control changes should be made before adds and after deletes
- In multi-master environments, access control changes and the

associated data changes should be made on same system.

Even when ACI is faithfully replicated (with the same transfer format) among heterogeneous members of a replica group, there is no guarantee that an ACI change is expressed similarly everywhere in the group. This caveat is partly due to the open issues with respect to partitioning mentioned above, and partly due to vendor differences with regard to the expression of security policy.

2.3 Change Logs

Requirement G4 of [RFC3384] states that meta-data must not grow without bound. Since it is unrealistic to assume that meta-data won't be needed during replication, designers must consider how and when meta-data can be purged.

Replicas that use connections with intermittent quality should use explicit replica cycle scheduling. Since the systems know when replication should have occurred, delayed replication can be detected and manual intervention initiated before the meta-data grows without bound. In extreme cases, it may be necessary to remove a replica from the replication group and restore it once better connectivity is available.

In a multi-master system, it is possible for a consumer to receive changes that cannot be applied. For example, a modify request for an entry may arrive before the add request that creates that entry. The replication system will typically queue this change and wait for additional changes (see Section 3.3).

3 Naming Considerations

A number of naming models have been proposed for directories ([RFC1255], [RFC2377]), and many others have been implemented on an ad hoc basis. Each of these models specifies the naming attributes to be used and provides rules for using them which may also include containment rules.

The naming plan applies to the directory as a whole, not the individual servers holding replicas. Therefore, in a heterogeneous replicated environment, all of the replicating servers must be capable of supporting all of the rules for the naming plan in use for that directory.

Some directory implementations have naming constraints (e.g. containment rules, restrictions on attributes that can be used for naming). If such an implementation is part of a replicated directory, those constraints will have to be observed by all participating directories. If the environment contains implementations with incompatible constraints there is a major problem. This should be checked as early in the design phase as possible.

Applications often have their own requirements on naming; if a directory supports multiple applications it may have to support multiple naming schemes. Thus, if two independent applications start sharing previously separate directory information, care should be taken that the naming is consistent across the applications. A difference in name form may be accepted through LDUP without constraint violation, but nevertheless result in unexpected behavior from a cross-application perspective. Consistent naming is not only important to the directory, but to the applications that consume directory information as well.

4 Conflict Resolution Considerations

4.1 Consistent Access after Changes

Many operations on a directory are done as a set of steps. For example, a new object may be created by one operation, and its values may be filled in as part of a separate LDAP operation. An administrator may add a user to a directory, and that user may then try to log in using the new entry.

Replicated LDAP directories provide loose consistency [RFC3384]. A new entry or a change to an existing entry will not reach all replicas immediately; there will be some delay before changes are available on all replicas. Changes made (e.g. adding a new user) on one physical system may appear to be "lost" if checked on another physical system before replication is complete.

In general, LDAP applications should be prepared to operate correctly in the face of replication delays. In some cases, this means designing to allow for delay. In the case of the newly created user, it should be standard practice to ask the user to wait a while before trying to use the entry. In the case where the new object must be filled in, the application should make appropriate use of LDAP sessions to make sure that the same server is reached for both operations.

As a general rule, an LDAP application should bind once and not unbind until a complete set of related operations have been performed. To achieve load balancing of write operations in a multi-master environment, balancing the write-enabled connections is recommended over balancing the individual LDAP write operations.

In the single-master case, all write requests go to one server. If a set of related reads and writes are done, they should all be done on the master if possible.

In some cases, related requests will deal with data in different partitions that are not all available on a single server. In this

case, it is safer to keep sessions open to all servers rather than

closing the session with one server and opening one with another server.

It may not always be obvious to clients that they are using different servers. If a load distribution system is used between the client and the server, the client may find that a change request and a subsequent lookup are directed to different physical servers even though the original requests were sent to the same server name and/or address.

Since LDAP is session oriented, any load distribution system used should take sessions into account. Thus, keeping all related read and write requests within a single bind/unbind session should be the goal in this situation as well.

4.2 Conflict Resolution in Single-Master Systems

It is possible that resolution conflicts could occur in a single master replication system. Because requirement SM2 of [RFC3384] is a "SHOULD" and not a "MUST", it is possible for implementers to reorder changes. If changes are reordered, it is quite possible for a conflict to occur. Consider a case where schema changes are declared critical and must be moved to the front of the replication queue. Then the consumer servers might have to delete an attribute that still has values, and later process requests to delete the values of that now undefined attribute.

However, directory administrators may have scenarios where re-ordering of replication information is desirable. On a case-by-case basis, the directory administrator should make such decisions.

Many vendors may not implement conflict resolution for single-master replication. If such a system receives out-of-order changes from a system that does support them, replication errors will almost certainly occur. Designers should be aware that mismatches in the capabilities of replicating single-master directories could cause problems. Designs should not permit the master to re-order changes unless all slave copies are known to handle the situation correctly.

4.3 Problem Cases

4.3.1 Atomicity

The fact that replication does not guarantee the time order arrival of changes at a consumer allows situations where changes that were applied successfully at the supplier may fail in part when an attempt is made to apply the same change at the consumer. Some examples appear below; additional examples are given in Appendix B.5 of [RFC3384].

4.3.1.1 Locking

There is an entry with distinguished name "DN" that contains attributes X, Y, and Z. The value of X is 1. On replica A, a ModifyRequest is processed which includes modifications to change that value of X from 1 to 0 and to set the value of Y to "USER1". At the same time, replica B processes a ModifyRequest which includes modifications to change the value of X from 1 to 0 and to set the value of Y to "USER2" and the value of Z to 42. The application in this case is using X as a lock and is depending on the atomic nature of ModifyRequests to provide mutual exclusion for lock access.

In the single-server case, the two operations would have occurred sequentially. Since a ModifyRequest is atomic, the entire first operation would succeed. The second ModifyRequest would fail, since the value of X would be 0 when it was attempted, and the modification changing X from 1 to 0 would thus fail. The atomicity rule would cause all other modifications in the ModifyRequest to fail as well.

In the multi-master case, it is inevitable that at least some of the changes will be reversed despite the use of the lock. Assuming the changes from A have priority per the conflict resolution algorithm, the value of X should be 0 and the value of Y should be "USER1" But what is the value of Z at the end of the replication cycle? If it is 42, then the atomicity constraint on the change from B has been violated. But for it to revert to its previous value, grouping information must be retained. Therefore, it is not clear when such information may be safely discarded. Thus, requirement G6 in [RFC3384] may be violated.

The utility of locking mechanisms cannot be guaranteed with multi-master replication, and therefore results are likely to be misleading. As discussed further in section 6.1 below, its use in multi-master environments should be deprecated.

4.3.1.2 Partitioning

Partitioning (design of replica groups) also adds complexity. For example, suppose two servers, A and B, are members of a replica-group for area of replication X while servers B and C are members of replica-group for area Y. It is possible to issue a ModifyRDN operation on server B that moves an entry from area X to area Y. Replication in area X would delete the entry on server A while replication in area Y would add the entry to server C. However, if another change on server C prevented the add operation from working (e.g. an entry with the same RDN but a different GUID exists there already), then the change on server A is inconsistent and will need to be reversed. Other examples of cases of this class include group membership modification and access control scope.

4.4 General Principles

The examples above discuss some of the most difficult problems that can arise in multi-master replication. Dealing with them is difficult and can lead to situations that are quite confusing to the application and to users.

The common characteristics of the examples [RFC3384] are:

1. Several directory users/applications are changing the same data
2. They are changing the data at the same time
3. They are using different directory servers to make these changes
4. They are changing data that are parts of a distinguished name or they are using ModifyRequest to both read and write a given attribute value in a single atomic request.

If any one of these conditions is reversed, the types of problems described above will not occur. There are many useful applications of multi-master directories where at least one of the above conditions does not occur or where careful design can reverse one of the conditions. If, for example, all atomic read/modify requests for a given object can be directed to the same server, condition 3 will not occur. For cases where all four conditions do occur, application designers should be aware of the possible consequences.

5 Failover Considerations

One of the major reasons to use directory replication is to improve reliability of the directory system as a whole. Replication permits hot- and warm-standby configurations to be built easily.

But there are some issues that must be considered during design. In this situation, single-master systems actually raise more concerns than multi-master. Both are addressed below.

5.1 Common Issues

In both the single- and multi-master cases, clients must be able to find an alternate quickly when a server fails. Some possible ways to do this are detailed in [FindingLDAP] and [LDAPinDNS]. If all else fails, a list of possible servers can be built into client applications. Designers should consider how clients are notified that the server is again available.

When the failed server comes back up, it is brought back into synchronization with the other servers and is ready to take requests. It is always possible that the failed server, if it was acting as a supplier, was unable to completely distribute its pending changes before removal from service, leaving its consumers in an inconsistent state. During the period between its removal from service and its eventual return, the inconsistency may have been compounded by further application activity. At the time of publication there is no standard

automatic mechanism to rectify the problem, so the administrator must

use whatever mechanism is available to compare the replicas for consistency as soon after the event as is reasonable.

Note that the process used to bring a failed server back into replication can also be used to add a server to a set of replicating servers. In this case, the new server might be initialized from a backed-up copy of the directory or it may acquire the entire DIB via replication. The former method is usually preferable when the directory is large.

5.2 Single Master Issues

In a single-master system, the master is a single point of failure, as all modification has to originate at the master server. When high availability is a requirement, a quick, automated failover process for converting a slave replica to a new master is desirable, as the failover time becomes a major factor in determining system availability. The considerations in section 5.1 apply here; clients must know how to find the new master or a new slave in case of failure.

To aid in promotion of a slave replica, the master could replicate control information and meta-data (including replication credentials) so that this information is available during failover promotion. This data may either be replicated on a single "failover designate" slave (which would become the master during failover) or it could be replicated to all slaves. The first possibility has the advantage of minimizing the amount of extra replication while the second more robustly handles multiple failovers (i.e. failover of the newly promoted master to another slave before the original master has been restored). If this method is followed, data privacy mechanisms should be used to protect the replication session.

If data privacy mechanisms (e.g. encryption) are used to protect the replication session, the new master must have the necessary key information. Further this key information should be independent of the master that is using it (i.e. not tied to the IP address of the master server). If it is not independent, slave replicas could be pre-configured with the keys for all possible masters to reduce failover time.

Restoration of the failed or broken master can be handled in one of two ways:

- It could join the replica group and function as a slave.
- It could join the replica group and negotiate with the new master to synchronize and then take over as master.

In either case, clients need a way to know that a new server is available. If the broken master is returned to service as a slave,

then the administrator must, external to LDUP, distribute and resolve whatever pending changes remained undistributed and unresolved from

the time immediately before it was removed from service. If the broken master is returned as a new master, then care must be taken with its replacement master to ensure that all of its pending changes are distributed and resolved before it is returned to duty as a slave.

The slave replicas may also use the replication agreement to filter which master is allowed to submit changes. Such a model allows the slave servers to function correctly when the master server is "broken" and sending out incorrect updates. However, then it is necessary to update the replication agreement during the fail over process so that the slaves will accept updates from the new master. This is the case for both the original failure and the restoration of the restored master if that is how the restored master rejoins the replica group.

5.3 Multi-Master Issues

Typically, a multi-master configuration is used when high availability is required for writes as well as reads in the directory. Because there are multiple active servers prepared to take write requests, there is no "switchover" time in this case. But clients still need to be able to find an alternate server, so the considerations of Section 5.1 apply here.

6 Other Issues

6.1 Locking

Section 4.3.1.1 discussed the problems that can arise when the "modify" command in LDAP is used for locking in a multi-master environment. There are more general principles at work there. LDAP is a distributed and replicated directory service that is typically described as "loosely consistent".

In loose consistency, the data should eventually converge among the replicas, but at any given instant, replicas may be in disagreement. This stipulation is the general result of:

1. Delay due to replication intervals
2. Out of natural time order arrival of data at a replica
3. Temporary isolation of distributed systems from one another
4. Failure to accept a change due to conflict resolution failure on a replica

Because of loose consistency, data encountered by an LDAP operation may differ among replicas. Multi-mastering may exacerbate loose consistency, but single-mastering will not totally eliminate it if out-of-order replication is allowed (see Section 4.2). One must carefully assess the effect of loose consistency when considering the use of distributed LDAP servers as a data store. Applications which depend on synchronous consistency may be better suited for

transactional models and/or non-distributed data.

6.2 Backup and Restore

Backup of a directory server should have the following goals:

1. The directory (or the replica) can be unambiguously and faithfully restored from the backup.
2. The backup is an internally consistent snapshot of an entire replica during the time interval it took to make it.
3. Replication can resume on a machine restored from that backup without information loss.

Backup and restore of a single, operating directory server (rather than the entire directory system) presents its own challenges. "Loose consistency" works against the probability of achieving a loss-free copy of all the data in the directory, except under ideal conditions. Backup and restore of distributed directories is a decidedly easier task when the constraint of continuous availability is removed. In most cases, the removal of entire directory systems from write service is impossible, even for small periods of time. It is more practical to remove a single replica from service to achieve a condition of quiescence. Once all write load is removed, including write load due to replication, an internally consistent copy of the data may be obtained.

Replicas that have suffered catastrophic data loss may be restored from backups of working ones temporarily removed from service specifically to make a copy. This scenario illustrates the benefit of having three or more master replicas in the system: no single point of write failure in the event that one of the replicas must be restored from a copy of another.

The M11 requirement from [RFC3384] allows an empty replica to be brought up to date through replication. This feature duplicates, but does not make entirely unnecessary, backup procedures on directory servers. Backups are still needed to recover data that has been lost to all replicas, either through normal LDAP updates or through some catastrophic event.

7 Impact of Non-LDAP Changes/Constraints

7.1 Changes Outside of LDAP

LDAP directories are typically built on top of some database or file system. Thus there are ways to change the data that do not go through the normal LDAP change mechanisms (e.g. ModifyRequest). If the data is modified outside of LDAP, the changes will not be checked for schema conformance nor will access controls be checked as the changes are made. Since both integrity and security checks are omitted,

security can be adversely affected.

Also, many systems use the normal LDAP modification mechanisms to trigger replication. Changes made using non-LDAP mechanisms may not be replicated at all, leading to inconsistencies between replica copies.

7.2 Application Triggers

Directory servers commonly integrate one or more specific applications. To achieve this integration the directory server may intercept updates and run application-specific "trigger" code. Such triggers enforce directory invariants that cannot be expressed by the LDAP schema.

A simple trigger example is password policy enforcement. A directory server might interpret a request to replace the current value of the userPassword attribute with some new value as a request to first check that the new value conforms to the server's password policy (e.g. the value is sufficiently long and complex) before storing the new value. Using this trigger the directory server voids the security risk associated with passwords that are easy to attack.

A more complex trigger example is password hashing. A directory server might interpret a request to replace the current value of the userPassword attribute with some new value as a request to compute one or more secure hashes of the new value and store these hashes in one or more attributes, storing no value in the userPassword attribute. Using this trigger the directory server avoids the security exposure of storing the plaintext password.

Replication between directory servers with different application triggers will compromise directory integrity.

Similarly, one cannot extend the directory with stored procedures that execute on access - such as scripts, programs or controls which change the data - because the expression of such mechanisms may not be guaranteed to be consistent among heterogeneous servers.

7.3 Policy Conflicts Across Servers

In addition to the discussions of ACI in Section 2.3 and triggering in section 7.2, LDUP replication can not (by its definition) handle replication of information that makes use of policy not expressible in the LDAP protocol. A prime example of this is security encoding of attributes (e.g. userPassword). This encoding is typically implementation specific and is not easily expressible via the LDAP protocol. Therefore replication of userPassword attributes between directory servers that use different encoding schemes will impede replication in a way that is not describable as schema or syntax mismatch. This is because of the bind-time policy semantics that are

the true point of conflict.

In general, any attribute with semantics that are outside the scope of what is expressible by the LDAP protocol could result in strange replication errors. Therefore, distributed directory implementers should (in the absence of a way to express such semantics) either strive for a homogeneous set of servers or ensure during acceptance testing that a new server can support the existing semantics of their directory.

8 Security Considerations

This document discusses issues that arise in replication. Some of these issues are security related (e.g. replication of access control information) and the security implications are discussed in the relevant sections.

9 Acknowledgements

This document owes a lot to discussions on the LDUP mailing list. In particular, the authors would like to thank Ed Reed, whose email to the mailing list drove much of section 6.1, and Mark Brown for identifying and generating text on the issues discussed in section 7.

10 References

[FindingLDAP] R. Moats, R. Hedberg, "A Taxonomy of Methods for LDAP Clients Finding Servers", Internet Draft, draft-ietf-ldapext-ldap-taxonomy-05.txt, July 2001.

[InfoMod] R. Moats, R. Huber, J. McMeeking, "LDUP Replication Information Model", Internet Draft, draft-ietf-ldup-infomod-07.txt, June 2003.

[LDAPinDNS] M. Armijo, L. Esibov, P. Leach, R. L. Morgan, "Discovering LDAP Services with DNS", Internet Draft, draft-ietf-ldapext-locate-05.txt, March 2001.

[RFC1255] The North American Directory Forum, "A Naming Scheme for c=US", RFC 1255, September 1991.

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

[RFC2377] A. Grimstad, R. Huber, S. Sataluri, M. Wahl, "Naming Plan for Internet Directory-Enabled Applications", RFC 2377, September 1998.

[RFC2820] E. Stokes, D. Byrne, B. Blakley, P. Behara, ôAccess Control Requirements for LDAPö, RFC2820. May 2000.

[RFC3384] E. Stokes, R. Weiser, R. Moats, R. Huber, "LDAPv3 Replication Requirements", RFC 3384, October 2002.

Authors' Addresses

Richard V. Huber
Room C3-3B30
AT&T Laboratories
200 **Laurel Avenue South**
Middletown, NJ 07748
USA
E-Mail: rvh@att.com
Telephone: +1 732 420 2632
Fax: +1 732 368 1690

Gerald F. Maziarski
Room C3-3Z01
AT&T Laboratories
200 **Laurel Avenue South**
Middletown, NJ 07748
USA
E-Mail: gfm@att.com
Telephone: +1 732 420 2162
Fax: +1 732 368 1690

Ryan D. Moats
Lemur Networks
15621 **Drexel Circle**
Omaha, NE 68135
USA
E-Mail: rmoats@lemurnetworks.net
Telephone: +1 402 894 9456

Full Copyright Statement

Acknowledgement

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