Internet-Draft Ryan Moats
LDAP Duplication/Replication/Update Lemur Networks, Inc.
Protocols WG Rick Huber
Expires September 2003 AT&T Laboratories
John McMeeking
IBM
March 2003

Mandatory LDAP Replica Management
Filename: draft-ietf-ldup-mrm-02.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

The goal of standards for LDAP replication is to allow interoperable replication among products from many different vendors. Defining the mechanism to move data among replicas is a necessary part of this work, but management of the replicated environment must also be standardized for replication to be truly interoperable.

This document presents the replication management functions that must be performed. Whenever possible, these functions are defined in terms of existing LDAP functionality using existing LDAP operations and existing data definitions. In some cases, changes or additions to the existing model are required, and specifications for these changes are

included in this document.

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 [RFC2119].

Table of Contents

1 Introduction

In the LDAP replication architecture [Arch], the LDAP servers and replication agreements between them are represented by entries in the directory tree, as part of the replicated naming context. The LDAP replication information model [InfoMod] describes the contents of these entries.

Replication management entries, such as replicaSubentries or replication agreements, can be altered on any updateable replica using standard LDAP operations [RFC2251]. These entries are explicitly

included in the directory entries governed by any agreement associated with this area of replication. As a result, all servers with a replica

of an area of replication will have access to information about all other replicas of that area of replication and associated agreements.

The deployment and maintenance of a replicated directory network involves the creation and management of the replicas themselves and associated replication control information (e.g. replicationSubentries and replication agreements). This document outlines the administrative actions necessary to create and maintain replication agreements. Typically, administrative tools will guide the administrator and facilitate these actions.

2 Administrative Precursors

In this document the term "administrative user" refers to an identity that will be performing replication configuration by binding to and invoking operations on directory servers. Most LDAP server implementations have the concept of a superuser or power user, however this need not be the same as the administrative user, so long as the administrative user has been granted appropriate privileges. The administrative user MAY be running as an autonomous process, and MUST be capable of securely maintaining its own credentials.

Deployments SHOULD create an administrative user identity that is granted access to all servers holding a copy of a replicated area to perform the procedures described below, in particular to read the root DSE, the replicationContext prefix entry and all subordinate subentries. The administrative user who will be viewing or modifying the replication status MUST have already been provided with and established in the directory server or servers appropriate authentication credentials and authorization rights to retrieve attributes and invoke DIT modification operations that are beyond the ability of the 'average' directory user.

3 Operational "Atoms"

The following operational atoms are used to build up more complex tasks in section 4.

Most of these operational "atoms" make the following assumptions:

Through prior LDAP or out-of-band means the administrative user MUST have been granted the following access control permissions to the directory in order to establish replication:

- Create the naming context prefix entry
- Create subentries immediately below the naming context prefix entry

In several sections below, we refer to "source" and "target" servers. The "source" server is a server that already holds a copy of the area of replication. It may already be replicating that area with other servers. The "target" server does not currently hold a copy of the area of replication. The "target" is being added to the replica-group.

The LDUP Architecture [Arch] allows for read-only replicas. Setting up and maintaining a read-only replica will sometimes require that the administrator create or modify data on the read-only replica. In this case, the Write-Unwriteable control (Section 3.16) should be used.

In the remainder of the document, any numbered list of steps is intended to be performed in the order given. Un-numbered (dash) lists will be used where order is unimportant.

3.1 Add area of replication to a server

The client SHOULD invoke a ModifyRequest in which the object field is the context prefix of the replication context, and the modification list consists of a single item to add the value replicationContext to the attribute objectClass. If the server responds with the resultCode attributeOrValueExists, then the value is already there. If the server responds with a resultCode other than attributeOrValueExists or success, then this is an error. Should an error occur at this point, the server is in an inconsistent state and needs to be fixed.

After this step is completed, the server will begin storing change information for this area of replication.

3.2 Delete area of replication from a server

On the target server, the client SHOULD invoke a SearchRequest to find all replicaSubentry objects which refer to the area of replication:

1. Retrieve all the values of the replicaContextRoots attribute in the root DSE of the server. Each value is the distinguished name of the base entry of a Replication Context held on the server.
2. Perform a one-level LDAP search in each replicaContextRoot for objects of class replicaSubentry whose replicaURI attribute points to the given server (e.g. use a filter of "(&(objectclass=replicaSubentry-2)(replicaURI=ldap://thisserver))") where "thisserver" is the fully-qualified DNS name of the given server with the associated port if it is not port 389.

The client then examines the subtreeSpecifications to determine which one it wants and then deletes all replicaSubentries with that subtreeSpecification.

If any of the DelRequests fail, the area of replication has not been completely removed.

After this step is completed, the target server will no longer

replicate this area of replication nor will it record changes for later replication. However, the other servers in the replica group for this area will still have a replicaSubentry for the given server, and this should be cleaned up on all servers that hold the replica.

WG Issue: This is a reason for the subtreeSpecification issue. If it were a DN, this would be much easier. Otherwise, we're going to need a matching rule for subtreeSpecification.

3.3 Copy base of area of replication between servers

In this section, the "target server" is the server on which the client has just modified the root DSE.

The client MUST separately contact another server, one that already holds a copy of this replication context, and issue a SearchRequest on that server in which the baseObject is the DN of the base of the area of replication, the scope baseObject, the filter "(objectClass=*)" and the attributes list {"*", entryUuid}. If the client cannot obtain the single entry at this point, the procedure will fail.

Now that it has the entry, the client SHOULD invoke an AddWithMetaData (see 3.15.1) on the target server with entry set to the DN of the base of the area of replication and attributes the same list as obtained in the previous search.

If the server returns a resultCode other than success, it is an error, and the server will be in an inconsistent state.

3.4 Create server entry for an area of replication

Each server needs to have in its copy of the area of replication a replicaSubentry for each of the servers involved in replicating that area before replication can be started. These entries MUST have the following attributes:

- objectclass, with values top, Subentry and replicaSubentry
- cn
- replicaURI
- replicaType

and MAY contain other attributes, as described in the Information Model [InfoMod].

WG Issue: This means that InfoMod needs to explicitly define replicaOnline as DSA specific with a default value of FALSE.

3.5 Delete server entry from an area of replication

On the target server, the client SHOULD invoke a SearchRequest to find all replicaSubentry objects which refer to the area of replication:

1. Retrieve all the values of the replicaContextRoots attribute in the root DSE of the server. Each value is the distinguished name of the base entry of the base entry of a Replication Context held on the server.

2. Perform a one-level LDAP search in each replicaContextRoot for objects of class replicaSubentry whose replicaURI attribute

points to the given server (e.g. use a filter of "(&(objectclass=replicaSubentry-2) (replicaURI=ldap://thisserver))") where "thisserver" is the fully-qualified DNS name of the given server. The subentry control MUST be included on this operation.

The client then examines the subtreeSpecifications and the replicaURI to determine which one(s) it wants and then deletes all replicaSubentries with that proper subtreeSpecification and replicaURI.

If any of the DelRequests fail, the area of replication has not been completely removed.

WG Issue: This is another reason for the subtreeSpecification issue. If it were a DN, this would be much easier. Otherwise, we're going to need a matching rule for subtreeSpecification.

3.6 Modify replica

3.6.1 Change Replica Type

Note: This section covers only the simple protocol operation to change the replica type. Section 4.16 covers the full set of operations for converting from a ReadOnly to an Updateable replica.

The client SHOULD invoke a ModifyRequest with the Write-Unwriteable control (Section 3.16) in which the object field is the replicationSubentry, and the modification list consists of a single item to change the value of the attribute replicaType. If the server responds with the resultCode attributeOrValueExists, then the value is already there. If the server responds with a resultCode other than attributeOrValueExists or success, then this is an error.

3.6.2 Change Between Full/Partial Replica

Note: This section currently discusses how to change between fractional and full replication. Once the information model supports sparse replication, it will need to be added here.

This section only discusses the simple LDAP protocol operations to change between full and partial replicas. Additional actions are discussed in Section 4.15.

To switch from fractional to full replication, a client SHOULD invoke a ModifyRequest with the Write-Unwriteable control (Section 3.16) in which the object field is the replicaSubentry, and the modification list consists of two items: one to remove the attribute attributeExclusionFilter and the other to remove the attribute attributeInclusionFilter. If the server responds with a resultCode other than noSuchAttribute or success then an error has occurred.

To switch from full to fractional replication, a client SHOULD invoke a ModifyRequest with the Write-Unwriteable control in which the object field is the replicaSubentry, and the modification list consists of one or two items: one to add the attribute attributeExclusionFilter and/or the other to add the attribute attributeInclusionFilter. If the server

responds with a resultCode other than attributeOrValueExists or success, then an error has occurred.

3.6.3 Change Replica URI for one server for one area of replication

Note: This section covers only the simple protocol operation to change the replica URI. Replication will carry this change to other servers.

The client SHOULD invoke a ModifyRequest in which the object field is the replicationSubentry, and the modification list consists of a single item to delete the old value of the attribute replicaURI and add the new value. If the server responds with the resultCode attributeOrValueExists, then the value is already there. If the server responds with a resultCode other than attributeOrValueExists or success, then this is an error.

3.7 Add Replication Agreement

Each server that acts as a supplier in replication sessions needs to have in its copy of the area of replication a replicaAgreementSubentry for each server that it replicates to. ReplicaAgreementSubentry objects are created as subordinates of the replicaSubentry representing the supplier server. These entries MUST have the following attributes:

- objectclass, with values top, Subentry and replicaAgreementSubentry-2, and
- cn

and MAY contain other attributes, as described in the Information Model [InfoMod]. The cn attribute value has no significance to replication. The cn attribute SHOULD have a value that is meaningful to the replication administrator.

These objects MUST have the following attributes (defined as optional in the Information Model) for replication to occur:

- replicaDN

If replicaDN is absent, no replication occurs under this agreement.

A client AddRequest creates the replicaAgreementSubentry. The entry field of the AddRequest is the DN of the agreement (formed by using the cn attribute as the RDN naming attribute in combination with the DN of the parent replicaSubentry). The attributes field of the AddRequest contains the required attributes for the agreement and any optional attributes that are to be defined at this time.

Note: while replicationAgreementSubentries could be replicated, since the replicationCredentialsDN are contained in the replicationAgreementSubentry, there is a bootstrapping issue with replicating via anonymous replication. To avoid this, clients MAY create the replicationAgreementSubentries on all servers. If they do this, they should use the AddWithMetaData operation (Section 3.15.1) to

ensure that all replicationAgreementSubentries have the same entryUUID.

3.8 Delete Replication Agreement

To delete a replication agreement a LDAP client SHOULD invoke a DeleteRequest naming the replicaAgreementSubentry to be deleted.

The termination of replication agreements should be done with caution as it can easily result in a partition of the directory servers if performed incorrectly.

Once all replication agreements have been terminated between a server and others for a naming context, then that copy of the context on the server will be divergent, and any updates made there will not be propagated to any other server.

3.9 Modify Replication Agreement

To modify a replication agreement, the client SHOULD invoke a ModifyRequest naming the replicaAgreementSubentry. The modification list MAY include any of the following attributes:

- attributeExclusionFilter
- description

- replicaDN
- replicationMechanismOID
- replicationCredentialsDN
- replicationScheduleDN

No further administrative action is required for these changes to take affect.

Changing the replicaDN attribute has the effect of ending the existing agreement with the replica named by the old replicaDN value (if a value was present). Similary, changing the replicationMechanismOID attribute to specify ietf-ldup-full-update has the effect of ending incremental replication relationship. In this respect, these changes are equivalent to deleting a replication agreement and have the same considerations (see section 3.8)

The replicationStatus attribute cannot be modified.

3.10 Suspend Replication

The client SHOULD invoke a ModifyRequest in which the object field is the DN of the target replicationSubentry, and the modification list consists of a single item to change the value of replicaOnline attribute to false. If the server responds with the resultCode attributeOrValueExists, then the value is already there. If the server responds with a resultCode other than attributeOrValueExists or success, then this is an error.

If replicaOnline is FALSE for a replicaSubentry that represents the server containing the instance of the replicaSubentry, the server MUST NOT initiate or accept any new incremental replication sessions. If replicaOnline is FALSE for a replicaSubentry that represents a replica

other than the server containing the instance of the replicaSubentry, the server MUST NOT initiate or accept any new incremental replication sessions with that replica.

3.11 Resume Replication

The client SHOULD invoke a ModifyRequest in which the object field is the DN of the target replicationSubentry, and the modification list consists of a single item to change the value of replicaOnline attribute to true. If the server responds with the resultCode attributeOrValueExists, then the value is already there. If the server responds with a resultCode other than attributeOrValueExists or success, then this is an error.

If replicaOnline is TRUE for a replicaSubentry that represents the server containing the instance of the replicaSubentry, the server MAY initiate or accept new incremental replication sessions. If replicaOnline is TRUE for a replicaSubentry that represents a replica other than the server containing the instance of the replicaSubentry, the server MAY initiate or accept new incremental replication sessions with that replica.

3.12 Trigger an Immediate Replica Cycle

An immediate replication cycle can be triggered using an LDAP extended operation - "Trigger Replication". This operation takes 4 or 5 arguments:

. The distinguished name of the replicaSubentry for the target replica. If there is no instance of this entry on the server where the extended operation is executed, the operation is not performed and an error is returned.
. The LDAP URL of the target server. (Note: Per [Req] M8, a single Replication Agreement may accommodate more than one pair of servers. Thus this argument is necessary.)
. Type of replication session to be performed (full update or incremental update).
. Flags indicating whether the replication session is online or offline, and if offline whether the server should create the offline file or read the offline file. See Section 4.1.1 for more details of offline replication.
. If the replication session is offline, the name of the file to be used.

All other required information (connection information, authentication information, etc.) is obtained from the Replication Agreement.

The server on which the operation is executed immediately initiates a replica cycle with the target server. Updates are transferred in both

directions if allowed by the replication agreement.

If either the target server or the server executing the extended operation has a currently-active replica cycle for the replication

context specified by the extended operation, the extended operation will fail ([Arch] Section 10).

Since replication is normally an asynchronous operation, the Trigger Replication extended operation completes and returns status when replication is started. It does not wait for completion of the replica cycle and the returned status indicates whether the cycle was started successfully, not whether it completed successfully. Further progress of the replica cycle can be checked using other mechanisms (e.g. Section 4.9).

The detailed syntax of the operation and associated response are presented in Section 5.3.

3.13 Immediately Terminate a Replica Cycle

Note: this section discusses just the operation for terminating a replica cycle. See section 4.18 for a discussion of suspending replication or changing the replication schedule.

An in-progress replication cycle can be terminated immediately using an LDAP extended operation - "Terminate Replication". This operation takes 2 arguments:

1. The distinguished name of the replicaSubentry for the target replica. If there is no instance of this entry on the server where the extended operation is executed, the operation is not performed and an error is returned.
2. The LDAP URL of the target server. (Note: Per [Req] M8, a single

   Replication Agreement may accommodate more than one pair of servers. Thus this argument is necessary.)

All other required information is obtained from the Replication Agreement.

The server on which the operation is executed immediately terminates its current replica cycle with the target server. Termination will not interrupt transmission of a Replication Update [Arch], it will occur only between Replication Updates.

The detailed syntax of the operation and associated response are presented in Section 5.3.

Note: If data is queued awaiting replication between a pair of servers and if replication is set to trigger on updates, the replication system may automatically start a new replica cycle shortly after a cycle is terminated. To avoid this, the replication schedule should be adjusted to suspend replication before the Terminate Replication extended operation is issued.

3.14 Search with Meta-Data

Many pieces of replication meta-data are used by LDUP. In some cases (entryUUID, createdEntryCSN) they are stored as operational attributes

and can be read using the LDAP search operation. Other meta-data (deleted entries, CSNs related to changes of attribute values) are not visible via normal LDAP operations but must be obtainable by administrative users attempting to deal with replication problems (e.g. dealing with Lost+Found entries).

"Search with Meta-Data" is an extended operation that is modeled on the LDAP search operation [RFC2251]. There is an additional parameter that indicates what additional data should be returned be the search. The following values are permitted:

- deletedEntries - Any deleted entries in the scope of the search

  are returned. Deleted entries can be distinguished from non-deleted entries because the deleted entries will have a value in their deletedEntryCSN attribute. If the deletedEntries controlValue is given, the deletedEntryCSN attribute is automatically added to the AttributeDescriptionList of the search.
- attributeChangeState - Each attribute value returned as part of the search will be returned as a triple consisting of the

  attribute value, the deleted flag associated with the value, and the modificationCSN associated with the value.

The formal specification of the Search with Meta-Data extended operation is given in Section 5.3.

Note: Search with Meta-Data is designed as an extended operation rather than a control because the format of the returned data (sets of triples) is different from a normal LDAP search operation. Other than this, Search with Meta-Data operates the same as search.

3.15 Changing Replication Meta-Data

When fixing a replication problem, administrators need a way to modify meta-data values that are normally treated as operational attributes (createdEntryCSN, entryUUID) or as completely hidden data (deletedEntryCSN, deleted flag, modificationCSN).

This set of extended operations mirrors the normal LDAP operations that allow modification, but they allow modification of the associated meta-data as well.

3.15.1 Add with Meta-Data

The Add with Meta-Data extended operation is modeled on the LDAP Add operation [RFC2251].

The differences from the standard LDAP Add operation are:

- Each value is specified as a triple consisting of the value, the deleted flag to be associated with that value, and the modificationCSN to be associated with that value.
- A value may be supplied for the createdEntryCSN attribute.
- A value may be supplied for the entryUUID attribute.

- If the Add with Meta-Data extended operation is executed on a readonly replica it will be executed locally rather than returning a referral to an updateable replica.

The formal specification of Add with Meta-Data is in Section 5.3.

3.15.2 Delete with Meta-Data

The Delete with Meta-Data extended operation is modeled on the LDAP Delete operation [RFC2251]. The differences from the standard LDAP Delete operation are:

- A deletedEntryCSN may be supplied with the Delete with Meta-Data extended operation. In this case the delete operation is performed as in [RFC2251], except that the value of the deletedEntryCSN is taken from the controlValue.
- It may be flagged as a noRecordDelete. In this case the targeted entry is deleted with no meta-data left behind (no deletedEntry or "tombstone").
- If the Delete with Meta-Data extended operation is executed on a readonly replica it will be executed locally rather than returning a referral to an updateable replica.

The formal specification of the Delete with Meta-Data extended operation is given in Section 5.3.

3.15.3 Modify with Meta-Data

The Modify with Meta-Data extended operation is modeled on the LDAP

Modify operation [RFC2251].

The differences from the standard LDAP Modify operation are:

- Each value is specified as a triple consisting of the value, the deleted flag to be associated with that value, and the modificationCSN to be associated with that value.
- The createdEntryCSN attribute may be modified.
- The entryUUID attribute may be modified.
- If the Modify with Meta-Data extended operation is executed on a readonly replica it will be executed locally rather than returning a referral to an updateable replica.

The formal specification of Modify with Meta-Data is in Section 5.3.

3.16 Write-Unwriteable Control

There are a number of attributes defined in [InfoMod] which are marked "NO-USER-MODIFICATION". When fixing replication problems, there may be times when these values need to be changed. In addition, when administering readonly replicas it may be necessary for administrators to modify values on readonly replicas. The Write-Unwriteable control handles these cases.

The control can be attached to an LDAP modify operation. If the control is present, it allows the modify operation to change values that are marked "NO-USER-MODIFICATION" (the only exception is

attributes which are locally calculated by the implementation). Also, if the control is present, modify operations can be executed on readonly replicas. In this case the readonly replica will not return a referral to an updateable replica and the operation will be performed on the readonly replica.

The formal specification of the control is given in Section 5.5.

The current list of NO-USER-MODIFICATION attributes in [InfoMod] is:

attributeExclusionFilter (Section 8.2.4)
attributeInclusionFilter (Section 8.2.5)
replicationStatus (Section 8.2.7)
replicaType (Section 8.2.8)
updateVector (Section 8.2.9)
secondsToWaitDefault (Section 8.2.18)
secondsToWait1 (Section 8.2.19)
secondsToWait2 (Section 8.2.21)

4 Common Tasks

There are many tasks that administrators need to perform in a replicated environment. This section describes many typical tasks and describes how they are performed in terms of the atoms defined above.

4.1 Add a new replica to an existing replica group

Assume there is an existing replica group for a given area of replication. To add one new server (which does not already hold a copy of the area) to the replica group, perform the following steps:

1. Copy the base entry of the area of replication from one of the current servers to the new server (Section3.3).
2. On one of the existing servers in the replica group, create the replicaSubentry for the new server with the replicaOnline attribute set to "false" (Section 3.4). Replication will copy this to all the existing replicas (trigger immediate replication per Section 3.12 if necessary). Since replicaOnline does not replicate, it needs to be set "false" on all existing servers.
3. On the new server, create a replicaSubentry for one of the other servers in the replica group with the replicaOnline attribute set to "false". The UUID of this replicaSubentry MUST be identical to the UUID it has on the existing replicas, so Add with Meta-Data (Section 3.15.1) should be used to create the entry.
4. If the authentication mechanism used for replication requires

credentials, make appropriate entries on all existing servers for the new server and make appropriate entries for all the existing servers on the new server. If any of the credential data is in the current area of replication, use Add with Meta-Data on the

new server and make the UUIDs the same as those on the existing servers.

5. Create the replication agreement(s) on one of the existing servers and let it replicate to the other existing servers (Section 3.7).
6. Set the replicaOnline attribute to true for the new server on both the source and target servers (Section 3.11).
7. On the source server, trigger an immediate "full update" replica cycle (Section 3.12). The data replicated will include the replicaSubentries and replication agreements for all other servers in the replica group since they are in the area of replication.
8. On the new server, set the replicaOnline attributes in the replicaSubentries for all the other servers to "true".
9. On the servers in the replica group other than the source server, set the replicaOnline attribute in the replicaSubentry for the target server to "true".

4.1.1 Large area of replication support

In some cases, an area of replication is so large or available bandwidth so small that out-of-band mechanisms (e.g. mailing a tape) need to be used to transport the initial copy from the source to the target. The target then needs to be updated with changes made to the source since the copy was made.

While LDIF is typically used to transport bulk LDAP data, it is not suitable here because it does not transport replication meta-data (CSNs, GUIDs, etc.). To ensure that all needed data is available; bulk

replication data is stored as a stream of protocol elements from [Proto] in network byte order. There is an LDAP extended operation that will either cause a server to generate or read a stream of protocol elements (see Section 3.12). Use of the extended operation to generate or a read a file is OPTIONAL, but if the file is generated it MUST be a stream of protocol elements.

The following restrictions are imposed on the data stream:

- The BIND operation is unauthenticated. The extended operation that causes the destination server to read protocol elements from a local file should be restricted to privileged users only (See Section 6); this provides authentication for the operation.
- The "supplier initiated" protocol flow is used.
- A "full update" replication session is assumed.
- Since input is coming from a file, there are no responses. The stream of protocol elements should assume that all response codes are "REPL_SUCCESS".
- Since this is a "full update" session, [Proto] specifies that "the consumer's update vector should be assumed to be set to times

'earlier than' the oldest times known to the supplier server."
Thus the protocol stream can be generated without receiving a
createGroupingResponse extended operation from the consumer.

- The value of the groupCookie in the endGroupingRequest extended operation is ignored.

Once the file has been loaded on the destination server, the destination server should initiate a replication session with the source server (Section 3.12). This will pick up changes that have occurred since the original file was created.

In summary, to add a new server to an existing replica group for a large replica, follow these steps (steps 1-7 are the same as steps 1-7 of Section 4.1):

1. Copy the base entry of the area of replication from one of the current servers to the new server (Section3.3).
2. On one of the existing servers in the replica group, create the replicaSubentry for the new server with the replicaOnline attribute set to "false" (Section 3.4). Replication will copy this to all the existing replicas. Since replicaOnline does not replicate, it needs to be set "false" on all existing servers. (Trigger immediate replication per Section 3.12 if necessary.)
3. On the new server, create a replicaSubentry for one of the other servers in the replica group with the replicaOnline attribute set to "false". The UUID of this replicaSubentry MUST be identical to the UUID it has on the existing replicas, so Add with Meta-Data (Section 3.15.1) should be used to create the entry.
4. If the authentication mechanism used for replication requires credentials, make appropriate entries on all existing servers for the new server and make appropriate entries for all the existing servers on the new server. If any of the credential data is in the current area of replication, use Add with Meta-Data on the new server and make the UUIDs the same as those on the existing servers.
5. Create the replication agreement(s) on one of the existing servers and let it replicate to the other existing servers (Section 3.7).
6. Set the replicaOnline attribute to "true" for the new server on both the source and target servers (Section 3.11).
7. Generate an update file on the source server (Section 3.12)
8. Transport the file to the destination site by any appropriate means (FTP, express parcel, postal mail, etc.)
9. Load the file onto the destination server (create a disk file from tape if necessary)
10. Import the file into LDAP (Section 3.12)
11. Set the replicaOnline attribute to true for the new server
12. Trigger a replica session between the source and destination machine to pick up changes since the file was generated (Section 3.12)

4.2 Verify replication information is present between two servers

This section describes steps that verify the proper replication information is present for replication to occur between two replicas. There are two parts to this process:

1. Verify that the necessary objects have been created on both replicas. These objects include replicaSubentry objects representing the replicas, replication agreements describing consumer-supplier relationships between the replicas, and the credential and schedule objects named in the agreements.
2. Verify that it is valid to replicate between the replicas. For example, a fractional replica cannot act as a supplier to a full replica.

4.2.1 Verify that replication objects exist

This section describes how to verify that the necessary replication objects exist. Within this section the term replica refers to one of the two replicas for which information is being verified. It is assumed that an administrator has identified the replicas (both the IP address/host name and the replicaId), the area of replication, and which of the replicas are expected to act as suppliers. One or both of the replicas must be a supplier to the other replica.

1. The client SHOULD invoke an object scope search specifying the DN of the root entry of the area of replication. This entry MUST exist on both servers, and the objectclass attribute values MUST include the value replicationContext.
2. On the replicas which the administrator has indicated is a supplier, the client SHOULD invoke a baseObject scope search of the root DSE requesting the replicationSubentries attribute. The value for this attribute MUST name exactly one replicaSubentry object that is a child of the root entry of the area of replication. The distinguished name of the entry MUST be of the form:

   cn=<replicaId>,<DN of root entry of area of replication>
3. On each supplier replica, the client SHOULD invoke a baseObject scope search specifying the DN of replicaSubentry that was identified in the replicaSubentries attribute of the root DSE in the previous step. This object MUST exist for replication to occur, and MUST include the value replicaSubentry in the list of objectclass values. The client SHOULD invoke an object scope search for the other server. The base DN for this search is of the form:

   cn=<replicaId>,<DN of root entry of area of replication>

   where <replicaId> is the replicaId of the other replica. This object MUST exist, and MUST include the value replicaSubentry in the list objectclass values.
4. On each supplier replica, the client SHOULD invoke a oneLevel search, specifying the DN of replicaSubentry for that replica as the baseObject, and a search filter like:

   (&(objectclass=replicationAgreementSubentry-2)
   (replicaDN=<consumer-replica-subentry-DN>))

   where <consumer-replica-subentry-DN is the DN of the

replicaSubentry representing the other server, which is the consumer in this agreement. There MUST be at least one entry in the search results. For incremental update replication to occur at

least one of the entries in the search results MUST have a replicationMechanismOID attribute which is either absent or has the value ietf-ldup-incremental-update [Proto].

5. On each supplier replica, the client SHOULD invoke a baseObject for each object named in the replicationCredentialsDN and replicationScheduleDN attribute values of the replicationAgreement entries discovered in the previous step. These attributes are optional, but if the values are present, the objects named by the values of these attributes MUST exist.

If all the above steps are successful, the objects necessary for replication to occur have been created.

4.2.2 Verify that it is valid for replication to occur between these servers

For a replica to act as a supplier to another replica, the set of entries and attributes specified in the consumer replica's fractional entry specification MUST also be present in the supplier's fractional entry specification. The fractional entry specification is defined by the attributeExclusionFilter and attributeInclusionFilter attributes of the replicaSubentry object for the replica. If these attributes are not present the replica is a full replica.

The client SHOULD perform a baseObject scope search on the supplier replica specifying the replicaSubentry objects for both replicas to obtain the fractional entry specification for the replicas. The following conditions MUST be satisfied:

1. The attributeExclusionFilter of the supplier MUST NOT contain attributes that are not present in the attributeExclusionFilter of the consumer. This condition is also satisfied if the attributeExclsionFilter attribute is absent from either the supplier replicaSubentry or both the supplier and consumer replicaSubentry objects.
2. The attributeExclusionFilter of the supplier MUST NOT contain attributes that are present in the attributeInclusionFilter of the consumer. This condition is also satisfied if the attributeExclusion filter attribute is absent from the supplier replicaSubentry, or if the attributeInclusionFliter attribute is absent from the consumer replicaSubentry.
3. The attributeInclusionFilter of the consumer MUST NOT contain attributes that are not present in the attributeInclusionFilter of the supplier. This condition is also satisfied if the attributeInclusionFilter attribute either is absent from the consumer replicaSubentry or is absent from both the supplier and consumer replicaSubentry objects.

4.3 Start replication between two servers

For this operation, the client SHOULD follow the steps in Section 4.1 followed by starting replication as specified in Section 3.11.

4.4 Suspend Replication activity on one area of a server

For this operation, the client SHOULD issue a search request to find the replicationSubentry of the target area on the target server. Then the client SHOULD suspend replication for this area as specified in Section 3.10.

Note: As replicaOnline is DSA-specific, changing the value to false on one server will not change the value on other servers. Those servers will keep trying to initiate replication and failing. To avoid this, replicaOnline must be set to false on all servers.

4.5 Suspend Replication activity on all areas of a server

The client SHOULD retrieve all the values of the replicaSubentries attribute in the root DSE of the server. The client SHOULD then suspend replication on the replicationSubentry for the target server by following Section 3.10.

Note: As replicaOnline is DSA-specific, changing the value to false on one server will not change the value on other servers. Those servers will keep trying to initiate replication and failing. To avoid this, replicaOnline must be set to false on all servers.

4.6 Restart Replication activity on one area of a server

For this operation, the client SHOULD issue a search request to find the replicationSubentry of the target area on the target server. Then the client SHOULD resume replication for this area as specified in Section 3.11.

Note: if the client turned off replicaOnline on all servers in an area of replication (as discussed in sections 4.4 and 4.5 above), the client will need to turn on replicaOnline on all servers to resume replication.

4.7 Restart Replication activity on all areas of a server

The client SHOULD retrieve all the values of the replicaSubentries attribute in the root DSE of the server. Then the client SHOULD take each area in turn and contact each server in that area's replication group. The client SHOULD then resume replication for each replicationSubentry for the target server by following Section 3.11.

Note: if the client turned off replicaOnline on all servers in an area of replication (as discussed in sections 4.4 and 4.5 above), the client will need to turn on replicaOnline on all servers to resume replication.

4.8 Halt replication

To halt replication (i.e. terminate a current replication session and then prevent further replication occurring), the client follows the appropriate steps from sections 4.4 and 4.5 above, inserting the step

of issuing a termination operation (as specified in Section 3.13) after replicaOnline is set to FALSE. The reason for terminating after setting replicaOnline to FALSE is to avoid a new replication session starting immediately after the terminating operation is complete.

Note: the difference between Halt and Suspend replication is that suspend allows a currently ongoing replication session to finish, while halt specifically invokes the operation of 3.13 to immediately terminate an ongoing replication session.

4.9 List status of a particular area of replication on a given server

There are many pieces of information that could be considered "status". A number of them are listed below, and a description of how to read them is provided.

An area of replication is defined by its replicaSubentry (the replicaSubentry that refers to the local server in its replicaURI attribute). The DN of this subentry will be called SDN in the rest of this section.

4.9.1 Local Replica On-Line

To determine whether the given area of replication on the local server is on-line, check the replicaOnline attribute of SDN.

4.9.2 Remote Replicas On-Line

To determine which other servers have the same replica on-line, do a one-level LDAP search in the parent container of SDN. The filter should be set to find entries of objectclass replicaSubentry-2 which have subtreeSpecification identical to the subtreesSpecification of SDN The replicaURI and replicaOnline attributes of the objects that match the filter will show what other servers hold this replica and whether they are on-line or off-line. Since replicaOnline is DSA-specific, this search MUST be performed on each server that holds the replica.

4.9.3 Check Status of Outward Replication

If the area of replication in question uses replication agreements, there is an optional replicationStatus attribute in the replication agreement. If it exists, it holds a human-readable text message containing the status of the most recent attempt at replication for the replication agreement which contains it. To get the status for all outgoing replication from the local server, do a one-level LDAP search with base SDN, filter of objectclass=replicaAgreementSubentry-2, and retrieve the replicationStatus attribute.

4.9.4 Check Status of Incoming Replication

To get the status for incoming replication to the local server, locate the replicaSubentries for other replicas of the area of replication as described in Section 4.9.2. Then for each replicaSubentry, do a one-level LDAP search with base being that replicaSubentry using a filter

of (objectclass=replicaAgreementSubentry-2), and retrieve the replicationStatus attribute.

To be safe, replicationStatus SHOULD always be checked on the supplier server. The copy on the consumer server may not be correct if there was a replication problem. In the multi-master case, any server may act as a supplier and all servers that hold the replica SHOULD be checked.

4.10 List all areas of replication defined on a given server and their status

To find all the areas of replication on a given server, do the following on that server:

1. Retrieve all the values of the replicaContextRoots attribute in the root DSE of the server. Each value is the distinguished name of the base entry of a Replication Context held on the server.
2. Perform a one-level LDAP search in each replicaContextRoot for objects of class replicaSubentry whose replicaURI attribute points to the given server (e.g. use a filter of "(&(objectclass=replicaSubentry-2)(replicaURI=ldap://thisserver))") where "thisserver" is the fully-qualified DNS name of the given server.

This will provide a list of all replicas held on the given server. Section 4.9 describes how to determine the status of each area.

4.11 Restore a server and replication agreements after a server crash

To restore a server and replication agreements after a server crash, the following steps SHOULD be performed.

4.11.1 Recent Backup is Available

If a sufficiently recent backup is available, each area of replication MAY be recovered by doing the following:

Note that the backup must contain UUIDs and CSNs.

1. Suspend replication to the server from all supplier servers (see Section 3.10).
2. Restore the directory data and replication meta-data from the backup.
3. If replication agreements to the server have been deleted, recreate the desired replication agreements.
4. Compare the update vector for this replica, to the update vectors

for other replicas (on the servers that hold the replicas). If the update vector for this server is greater than the minimum of the CSNs present in the other update vectors, resume replication to this server. No further action is necessary, as the other servers contain sufficient replication updates to recover all

subsequent updates via incremental replication. Otherwise, containue to the next step.

5. Suspend replication to a replica that acts as a supplier to this server. This is done so that no changes occur on the supplier while recovery is in progress.
6. Compare the DITs for each area of replication on this server with the DIT on one of the servers acting as a supplier to this server (section 4.20). For each difference found, repair the entry (section 4.21).
7. Set the recovered server's update vector on the recovered server to the value of update vector for the area of replication on the supplier server.
8. Resume replication to the supplier replica and to the recovered server.

4.11.2 Backup is Not Available

If a sufficiently recent backup is not available, the following steps SHOULD be performed:

1. **Suspend replication to the server from all supplier servers (see Section 3.10).**
2. **Clear the contents of the server.** The mechanism for doing this is implementation specific.
3. **Initialize the areas of replication on the server as if adding a new server to a replica group (section 4.1).**
4. **Start replication.**

4.12 Split an Area of Replication

To split an area of replication, the atoms are:

1. **Add the replicationContext objectclass to the root entry of the new area of replication (Section 3.1)**
2. **Create replicaSubentry objects under the new area of replication for each replica of the parent area of replication (Section 3.4)**
3. **Create any replication credentials objects and replicaEventSchedule objects that will be referenced by the** replicaAgreement objects to be created in step 4.

   JAM - I put the above there in case of any referential integrity issues.

4. **Create replicaAgreement objects under the new replicaSubentries, where agreements are created that correspond to each agreement** defined for the parent (Section 3.7).

   WG Issue: Schedules are referenced by DN. RVH - Can the schedules of a different area of replication be referenced? Is there

something that says that the schedules must be IN the area of replication they control? JAM - I think that schedules and credentials need only be accessible to the replicas that use them. That said, they are defined as subentries, and ought at least be

under a replicaSubentry. RDM - If this means that there are no

reusable schedules and credentials, I think that is a Bad Thing. TJH- Thinks nothing in infomod says that schedules are not reusable. RVH- But infomod requires schedule to be under the replicaSubentry, so they can't be reused because they can only be under one replicaSubentry. Maybe we will settle this when we talk to the infomod authors. JAM - infomod says it is thought these objects will be placed below replicaAgreements but this is not required. Maybe we can drop this issue? Or maybe we want infomod to explicitly say that schedules can be placed elsewhere (so they can be shared/reused?). Also, I think we decided that schedules did not need to be subentries, or at least that subtreespecification had no meaning for schedules.

5. **Modify the subtreespecification of the superior replica's replicaSubentry to exclude the new subordinate area of** replication.

JAM - Doesn't the scope of a subentry exclude subordinate subentries governing the same kind of administrative area? So we could remove this step?

These operations must be performed on each server containing a replica of the parent area of replication.

WG Issue: RVH - Why? - Aren't all of the items changed WITHIN the original area of replication? So won't they replicate as long as step **4 is done last? JAM - Once the new area of replication is defined on** one server, it quite likely defines the bounds of the superior. That means that any further activity in the new area is replicated according to the subentries and agreements defined for it - initially none. It would be cleaner, in some respects, though, it would be nice if defining the new area of replication cloned the subentries, agreements, etc. from the superior and was replicated. RDM - If that isn't covered in info-mod, I think it probably should be.

4.13 Move an existing area of replication to a new server

First add the area of replication to the new server as described in Section 4.1. Then delete the area of replication from the old server as described in Section 3.2.

Note that the "atom" in Section 3.2 will remove the old server from the replica group but it will not delete the entries in the area of replication from the old server. If the entries are to be deleted, this can be done using standard LDAP operations after the old server is removed from the replica group.

4.14 Join two Areas of Replication

This section describes how to join two areas of replication.

4.14.1 Preconditions

Before joining two areas of replication, certain preconditions need to be satisfied:

1. **Any server that contains a replica of one area of replication must also contain a replica of the other area of replication. This may** require copying either area of replication to additional servers, or deleting either area of replication from servers.
2. **The replicas on any given server MUST be of the same type.** Both replicas must be updateable, both-readonly, or both primary. Furthermore, if the replicas are readonly, they must both be full replicas, or must both be fractional replicas with identical fractional entry specifications.
3. **One area of replication must be directly subordinate to the other.**

4.14.2 Procedure

1. In each of the superior area's replicaSubentries, change the subtreespecification attribute to include the subordinate area.
2. Remove the replicationContext object class from the root of the subordinate area (this will replicate, so it only needs to be done on one server).
3. To clean up, remove the replicaSubentry entries (and any subordinate replication agreements) from the subordinate area of

   replication.

4.14.3 Server requirements

When the replicationContext objectclass is removed from the root of an area of replication, the server MUST immediately treat entries within the area of replication as belonging to the parent area of replication (if there is any). This includes replicating any pending replication updates (those not yet replicated to other replicas) as if they occurred under the parent area of replication, as well as preserving any Lost and Found entries.

The replicaSubentries have a subtreespecification attribute which defines the "bottom" of the area of replication. At some point this has to be changed. If the subtreespecification is changed BEFORE the subordinate replicationContext is removed, we should be OK. Depending on the implementation, some changes may be sent twice (once in each of the overlapping areas of replication), but that shouldn't matter - conflict resolution can sort things out.

If a server receives a request to delete the replicationContext from an area of replication, and there is a parent area of replication, the Server MUST verify that these replicas are of the same type, and if fractional, that the fractional entry specifications are identical. If the replicas are not of the same type, the request MUST fail with resultCode unwillingToPerform.

4.15 Stop Replicating an Area of Replication.

This section describes how to stop replicating an area of replication. At the end of the procedure, the subtree represented by the area of replication will exist on one server, all replication agreements will have been deleted, and the root of the area of replication will no

longer be an area of replication. The server on which the subtree will remain is referred to as the surviving replica.
To stop replicating an area of replication, a client with administrative authority should perform the following operations:

1. **Change the replica type of the non-surviving replicas to readonly** (see section 3.6.1)
2. **Halt replication by changing the replicaOnline attribute of all replicaSubentries on all servers to "false".**

After halting, the client MAY optionally delete information by:

3. **Delete all replication agreements (Section 3.8).**
4. **Delete all replicaSubentries under the area of replication** (section 3.5)
5. **Issue a modifyRequest to the surviving server where the object field is the DN of the area of replication, and the modifications** list consists of a single item, delete the attribute objectclass with value replicationContext (Section 3.2).

4.16 Convert a read-only replica to an updateable replica

To convert a read-only replica to an updateable replica, the client SHOULD send a single request that follows section 4.8.1 to change the replica type in the replicaSubentry on the target server to '2' (Updatable) using the write unwritable control (3.16).

If the read-only replica is not a full replica, this request SHOULD also follow section 3.6.2 to make it a full replica before making it updateable.

As noted in [InfoMod], "The consequences of having incomplete updateable replicas are not fully understood. LDAP DSAs MAY require updateable replicas to be complete replicas." If the DSA requires the updatebale replica to be complete and the client sends a single request that follows section 3.6.1 *only* and the readOnly replica is not currently complete, the DSA may respond with an unwillingToPerform error.

The DSA upon receiving and processing the request should trigger an immediate replica cycle to receive necessary data to make the target replica complete.

Note: single-master implementations may not support the concept of two updateable masters being simultaneously active. In this case, the client must convert the original master to read only via the following section before converting the new master to updateable via these steps.

4.17 Convert an updateable replica to a read-only replica

To convert an updateable replica to a read-only replica, the client SHOULD send a single request that follows section 3.6.1 to change the replica type in the replicaSubentry to '3' (Read-Only).

The server, upon receiving and processing such a request SHOULD:

1. Accept only LDAP search requests for that replica.
2. Finish replicating changes that had been accumulated.

4.18 Postpone a Replica Cycle to a Later Time

To temporarily halt replication to a particular server see Section 4.4.

To temporarily halt replication on all servers for a particular area of replication see Section 4.5. To resume replication after these temporary halts, see Section 4.6 and 4.7 respectively.

To change the schedule for scheduled replication, find the replicaAgreementSubentry for the given replication (see Section 4.9 and 4.10). The replicationScheduleDN attribute of the replicaAgreementSubentry contains the DN of the scheduling information.

Information on how to set the scheduling information can be found in [Infomod], [RFC3060], and [Policy].

If a replica cycle is already in progress, it can be terminated as described in Section 3.13.

Note that there are several events that may trigger a replica cycle, and schedule is only one of them. If, for example, a given replication agreement triggers replication whenever a change is made in the area of replication, a new cycle may be triggered as soon as the current cycle is terminated.

4.19 Examine Replication Audit History on a Server

While whether DSAs store replication audit history in the directory is outside the scope of this document, DSAs MUST store audit history in a file available to users of the underlying operating system (OS). A person wanting to examine the replication audit history should make use of the underlying OS. Whether they are required to have special permissions is outside the scope of this document.

The reason for storing this information outside the directory is so that the administrator may still have access to it in cases of directory failure or inaccessibility.

4.20 Compare Two Replicas on Two Servers for Differences

It may be desirable to compare replicas of an area of replication on two servers for differences. The process for doing this is to do a recursive singleLevel search starting at the root entry of the area of replication. The search SHOULD specify an attribute list that includes the value ?*? (all non-operational attributes), as well as the following operational attributes: entryUuid. Each search is performed

on both servers, and the results compared as follows:

1. **If the DN of an entry matches the DN of a subordinate area of replication identified in the replicationContexts root DSE** attribute for that server, exclude that entry from further processing.

2. **Compare the set of RDNs from the search on each server to determine if there are entries present on one server that are not** present on the other server.
3. **For each entry that exists in both servers, compare the set of attribute values returned from each server to determine if there** attribute values present in the entry on one server that are not present in the entry on the other server.
4. **For each entry that is not the root of a subordinate area of replication, form the search and comparisons described above.**

4.21 Fix an Entry Without Triggering Replication

When conflicts cause entries to be put in the Lost+Found area, the administrator needs a mechanism to make appropriate changes. These changes may include fixes to replication meta-data (UUIDs, CSNs, etc.) that cannot be changed using normal LDAP operations. Once the revised entries have been stored, any future replication operations will be based on the modified meta-data.

The "atoms" of Section 3.14 can be used to read meta-data that is not readable through normal LDAP operations. entryUUID and createdEntryCSN are available as operational attributes so they can be read with a normal LDAP "search". The atoms of Section 3.15 can be used to write meta-data.

Typically, an administrator would resolve a replication conflict using the following steps:

1. Read (using Section 3.14) and temporarily store all the meta-data

   associated with a conflicting set of entries (where some of the entries may be in Lost+Found)
2. Decide what the final result should be (including all associated meta-data)
3. Depending on the complexity of the change and on whether the entry to be changed has children:
   a. Delete (using Delete with Meta-Data defined in Section 3.15.2)

      the conflicting entry or entries, and
   b. Build the "correct" new entry or entries (with all appropriate meta-data) on all affected nodes using Add with Meta-Data from Section 3.15.1; or
   c. Use Modify with Meta-Data (Section 3.15.3) to make the change.

Changes may need to be made on several replicas. In all cases, care should be taken to keep the UUIDs of the entries consistent across replicas.

4.22 Check Reported Schema Mismatches Discovered During Replication

DSAs MAY store the result of reported schema mismatches in the directory. They SHOULD store the schema mismatch and any resulting action in the Audit History. The record SHOULD include the type of mismatch (some examples may be found in [USAGE]) as well as the resulting action: items moved to lost+found, items not added, etc.

4.23 Adding a new directory server to a replica group and initializing the contents

In this case, the client:

1. **Copies the base entry for the area of replication from a source to the target (section 3.3)**
2. **Create the entries for the new server on all servers in the replica group (section 3.4)**
3. **Create the entries for the existing replica group servers on the new** (section 3.4)
4. **Create the replication agreement on the new server and one other server (section 3.7)**
5. **Client issues a "Initiate Full Update" request to a full replica for the new replica -- or new replica requests consumer initiated full** update (section 3.12).

4.24 Restore from the master failure in a single-master system

To provide a fast fail-over mechanism for the failure of the master server in a single-master system, the following steps SHOULD be performed:

1. When the system is initially set up, at least one server SHOULD be designated as the fail-over server. This does not reflect a special replica type for the server, rather it reflects an administrative decision. This server is referred to as the fail-over replica in the following steps.
2. For each area of replication, replication agreements SHOULD be created between the fail-over server and each of the replicas to which the master server acts as a supplier. The fail-over server SHOULD be defined as the supplier in these agreements. These agreements serve two purposes: The fail-over server will not purge replication updates until all replicas have received the change. And the server is pre-configured to take over as master with minimal action.
3. An agreement MAY also be created to act as a supplier to the master. This allows the master server to be updated via replication if the failure does not involve a loss of data on the master server.
4. In the event of a failure of the master server, change the replicaType of the fail-over server to updateable or primary (see section 5.17). The fail-over server is now ready to act as a master server.

WG Issue: The above steps imply the need to ensure that the master replicates changes to the fail-over server before replicating a change to other servers. Otherwise, a replica may have changes that have not been replicated to the fail-over server. This can be accomplished by

requiring that if writable replica notes that there is only one other replica with replication agreements defined, the supplier should replicate to that server first. TJH- More generally, replicate first to servers that are suppliers.

WG Issue: Step 2 implies that a replica that acts as a supplier to no other server need only keep sufficient state information to satisfy idempotency and conflict resolution.

If the above steps are not taken, a full read-only replica can be promoted to a master by following these steps:

1. Designate one replica as the new master server. This master SHOULD be a replica that has an update vector at least as recent as any other replica. Do not change the replicaType at this time.
2. Perform pair-wise DIT comparisons between the new master and each other replica. Record the discrepancies and ensure that the affected entries are removed or fixed on all servers (see section 4.21 - fix entries)
3. On the new master, change the replicaType to updateable or primary and mark the replicaSubentry for the new master as offline. This ensures that the server will hold client updates for future replication but not replicate them at this time.
4. Add replication agreements, replication credentials entries, and replication schedule entries as needed between the new master and all other replicas.
5. On the new master, mark the replicaSubentry online. Replication of the replication agreements and other client updates will start.

5 Formal Specifications

The Replica Management features depend heavily on defined LDAP and LDUP structure, operations, and data formats. But some changes will be needed to accommodate Replica Management. All these changes are pulled together in this section for easy reference.

5.1 New/Modified Object Classes

TBD

5.2 New/Modified Attributes

TBD

5.3 New/Modified Extended Operations

Trigger Replica Operation

TBD

5.4 New/Modified Replication Primitives

TBD

5.5 New/Modified Controls

TBD

6 Security Considerations

For security purposes it is important to be able to limit the number of individuals with administrative access and to track the actions performed by each administrator. Thus, servers SHOULD allow multiple administrative users, and they SHOULD allow each administrative user to have distinct rights. It SHOULD be possible to log all of the administrative actions discussed in this document and the log entry SHOULD include the identity of the administrator performing the action.

In all cases, it is assumed that the client establishes a connection to the LDAP server and SHOULD authenticate using a recommended authentication method [RFC2829] that establishes the identity of the client user and SHOULD provide for connection integrity. In deployments where the underlying network service is vulnerable to eavesdropping and clients are intending to retrieve sensitive server credentials, the chosen method SHOULD also provide for encryption of data in transit.

In general, where the client is unaware of any network level protection services, it is RECOMMENDED that the client immediately after connection establishment invoke Start TLS to establish connection integrity and confidentiality, and follow this by authentication by one of:

- the "DIGEST-MD5" SASL mechanism,
- the "simple" authentication choice, or
- the "EXTERNAL" SASL mechanism if the client provided its certificate during TLS establishment.

The client MAY determine the supported authentication mechanisms of the server from the supportedSASLMechanisms attribute of the root DSE after Start TLS has been invoked, and use this to decide whether to use DIGEST-MD5 or EXTERNAL. See [RFC2830] for more information on TLS.

Some of the controls/extended operations defined in this document allow modification of the data that controls replication document (Modify with Meta-Data) or modification of data in the DIT (Trigger Immediate Replica Cycle from a file). Unauthorized use of these features can destroy a directory. Directories which support these features MUST also provide a mechanism to restrict their use to authorized users.

7 Acknowledgements

Thanks to Mark Wahl and Ed Reed for providing a lot of the initial text.

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

8 References

[Arch] J. Merrells, E. Reed, U. Srinvasan, "LDAP Replication Architecture", draft-ietf-ldup-model-01.txt.

[InfoMod] E. Reed, "LDAP Replication Information Model", draft-ietf-ldup-infomod-00.txt.

[Policy] J. Strassner, A. Westerinen, E. Ellesson, B. Moore, R. Moats, "Policy Core LDAP Schema", Internet draft, draft-ietf-policy-core-schema-13.txt, November 2001.

[Proto] E. Stokes, G. Good, R. Harrison, T. Hahn, "The LDUP Replication

Update Protocol", Internet Draft, draft-ietf-ldup-protcol-03.txt, November 2001.

[Req] E. Stokes, R. Weiser, R. Moats, R. Huber, "LDAPv3 Replication Requirements", Internet Draft, draft-ietf-ldup-replica-req-10.txt, July 2001.

[RFC2119] S. Bradner, "Key Words for Use in RFCs to Indicate Requirement Levels", RFC 2119, March 1997.

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

[RFC2829] M. Wahl, H. Alvestrand, J. Hodges, RL Morgan, "Authentication Methods for LDAP", RFC 2829, May 2000.

[RFC2830] J. Hodges, R. Morgan, M. Wahl, "Lightweight Directory Access Protocol (v3): Extension for Transport Layer Security", RFC 2830, May 2000.

[RFC3060] B. Moore, E. Ellesson, J. Strassner, A. Westerinen, " Policy Core Information Model -- Version 1 Specification", RFC 3060, February 2001.

[Usage] R. Huber, et al. "General Usage Profile for LDAPv3 Replication," draft-ietf-ldup-usage-profile-02, November 2001.

Authors' Addresses:

Ryan Moats
Lemur Networks, Inc.
Email: rmoats@lemurnetworks.net

Rick Huber
AT&T Laboratories
Email: rvh@att.com

John McMeeking
IBM
Email: jmcmeek@us.ibm.com

Full Copyright Statement

Acknowledgement

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