NFSv4 Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 12, 2011

J. Lentini
C. Everhart
NetApp
D. Ellard
Raytheon BBN Technologies
R. Tewari
M. Naik
IBM Almaden

March 11, 2011

NSDB Protocol for Federated Filesystems
draft-ietf-nfsv4-federated-fs-protocol-11

Abstract

This document describes a filesystem federation protocol that enables file access and namespace traversal across collections of independently administered fileservers. The protocol specifies a set of interfaces by which fileservers with different administrators can form a fileserver federation that provides a namespace composed of the filesystems physically hosted on and exported by the constituent fileservers.

Requirements Language

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].

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on September 12, 2011.

Copyright Notice

Table of Contents

1. Introduction 5
2. Overview of Features and Concepts 6
2.1. File-access Protocol 6
2.2. File-access Client 6
2.3. Fileserver 6
2.4. Referral 6
2.5. Namespace 6
2.6. Fileset 7
2.7. Fileset Name (FSN) 7
2.8. Fileset Location (FSL) 7
2.8.1. Mutual Consistency across Fileset Locations 8
2.8.2. Caching of Fileset Locations 9
2.8.3. Generating A Referral from Fileset Locations 9
2.9. Namespace Database (NSDB) 10
2.10. Mount Points, Junctions and Referrals 11
2.11. Unified Namespace and the Root Fileset 12
3. Examples 12
3.1. Creating a Fileset and its FSL(s) 12
3.1.1. Creating a Fileset and an FSN 13
3.1.2. Adding a Replica of a Fileset 13
3.2. Junction Resolution 13
3.3. Example Use Cases for Fileset Annotations 14
4. NSDB Configuration and Schema 15
4.1. LDAP Configuration 15
4.2. LDAP Schema 16
4.2.1. LDAP Attributes 19
4.2.2. LDAP Objects 37
5. NSDB Operations 40
5.1. NSDB Operations for Administrators 41
5.1.1. Create an FSN 42
5.1.2. Delete an FSN 43
5.1.3. Create an FSL 43
5.1.4. Delete an FSL 47
5.1.5. Update an FSL 47
5.2. NSDB Operations for Fileservers 48
5.2.1. NSDB Container Entry (NCE) Enumeration 48
5.2.2. Lookup FSLs for an FSN 48
5.3. NSDB Operations and LDAP Referrals 50
6. Security Considerations 50
7. IANA Considerations 51
7.1. Registry for the fedfsAnnotation Key Namespace 51
7.2. Registry for FedFS Object Identifiers 51
7.3. LDAP Descriptor Registration 54
8. Glossary 57
9. References 59
9.1. Normative References 59

9.2. Informative References 61
Appendix A. Acknowledgments 62
Authors' Addresses 62

1. Introduction

A federated filesystem enables file access and namespace traversal in a uniform, secure and consistent manner across multiple independent fileservers within an enterprise or across multiple enterprises.

This document specifies a set of protocols that allow fileservers, possibly from different vendors and with different administrators, to cooperatively form a federation containing one or more federated filesystems. Each federated filesystem's namespace is composed of the filesystems physically hosted on and exported by the federation's fileservers. A federation MAY contain a common namespace across all its fileservers. A federation MAY project multiple namespaces and enable clients to traverse each one. A federation MAY contain an arbitrary number of namespace repositories, each belonging to a different administrative entity, and each rendering a part of the namespace. A federation MAY also have an arbitrary number of administrative entities responsible for administering disjoint subsets of the fileservers.

Traditionally, building a namespace that spans multiple fileservers has been difficult for two reasons. First, the fileservers that export pieces of the namespace are often not in the same administrative domain. Second, there is no standard mechanism for the fileservers to cooperatively present the namespace. Fileservers may provide proprietary management tools and in some cases an administrator may be able to use the proprietary tools to build a shared namespace out of the exported filesystems. However, relying on vendor-specific proprietary tools does not work in larger enterprises or when collaborating across enterprises because the fileservers are likely to be from multiple vendors or use different software versions, each with their own namespace protocols, with no common mechanism to manage the namespace or exchange namespace information.

The federated filesystem protocols in this document define how to construct a namespace accessible by an NFSv4 [3530bis] or NFSv4.1 [RFC5661] client and have been designed to accommodate other file access protocols in the future.

The requirements for federated filesystems are described in [RFC5716]. A protocol for administering a fileserver's namespace is described in [FEDFS-ADMIN]. The mechanism for discovering the root of an NFSv4 namespace is described in [FEDFS-DNS-SRV].

In the rest of the document, the term fileserver denotes a fileserver that is part of a federation.

2. Overview of Features and Concepts

2.1. File-access Protocol

A file-access protocol is a network protocol for accessing data. The NFSv4 protocol and the NFSv4.1 protocol are both examples of a file-access protocol.

2.2. File-access Client

File-access clients are standard off-the-shelf network attached storage (NAS) clients that communicate with fileservers using the NFSv4 protocol, the NFSv4.1 protocol, or some other file-access protocol.

2.3. Fileserver

Fileservers are servers that store the physical fileset data or refer the client to other fileservers. A fileserver can be implemented in a number of different ways, including a single system, a cluster of systems, or some other configuration. A fileserver provides access to a federated filesystem via NFSv4, NFSv4.1, or some other file-access protocol.

2.4. Referral

A referral is a mechanism by which a fileserver redirects a file-access protocol client to a different fileserver. The exact information contained in a referral varies from one file-access protocol to another. The NFSv4 protocol defines the fs_locations attribute for referral information. The NFSv4.1 protocol defines the fs_locations_info attribute for referral information.

2.5. Namespace

The goal of a unified namespace is to make all managed data available to all clients via the same path in a common filesystem-like namespace. This should be achieved with minimal or zero client configuration. In particular, updates to the common namespace should not require configuration changes at the client. Filesets, which are the unit of data management, are a set of files and directories. From the perspective of the clients, the common namespace is constructed by mounting filesets that are physically located on different fileservers. The namespace, which is defined in terms of fileset definitions, fileset identifiers, the location of each fileset in the namespace, and the physical location of the implementation(s) of each fileset, is stored in a set of namespace repositories, each managed by an administrative entity. The

namespace schema defines the model used for populating, modifying, and querying the namespace repositories. It is not required by the federation that the namespace be common across all fileservers. It should be possible to have several independently rooted namespaces.

2.6. Fileset

A fileset is defined to be a container of data and is the basic unit of data management. Depending on the configuration, they may be anything between an individual directory of an exported filesystem to an entire exported filesystem at a fileserver.

2.7. Fileset Name (FSN)

A fileset is uniquely represented by its fileset name (FSN). An FSN is considered unique across the federation. After an FSN is created, it is associated with one or more fileset locations (FSLs) on a fileserver.

The attributes of an FSN are:

NsdbName: the network location of the NSDB node that contains authoritative information for this FSN.

FsnUuid: a 128-bit UUID (universally unique identifier), conforming to [RFC4122], that is used to uniquely identify an FSN.

2.8. Fileset Location (FSL)

An FSL describes the location where the fileset data resides. An FSL contains generic and type specific information which together describe how to access the fileset. An FSL's type indicates which protocol(s) may be used to access its data. An FSL's attributes can be used by a fileserver to decide which locations it will return to a client.

All FSLs have the following attributes:

FslUuid: a 128-bit UUID, conforming to [RFC4122], that is used to uniquely identify an FSL.

FsnUuid: the 128-bit UUID of the FSL's FSN.

NsdbName: the network location of the NSDB node that contains authoritative information for this FSL.

FslHost: the network location of the host fileserver storing the physical data

FslTTL: the time in seconds during which the FSL may be cached

Annotations: optional name/value pairs that can be interpreted by a fileserver. The semantics of this field are not defined by this document. These tuples are intended to be used by higher-level protocols.

Descriptions: optional text descriptions. The semantics of this field are not defined by this document.

This document defines an FSL subtype for NFS. An NFS FSL contains information suitable for use in an NFSv4 fs_locations [3530bis] or NFSv4.1 fs_locations_info attribute [RFC5661].

A fileset MAY be accessible by protocols other than NFS. For each such protocol, a corresponding FSL subtype SHOULD be defined. The contents and format of such FSL subtypes are not defined in this document.

2.8.1. Mutual Consistency across Fileset Locations

All of the FSLs that have the same FSN (and thereby reference the same fileset) are equivalent from the point of view of client access; the different locations of a fileset represent the same data, though potentially at different points in time. Fileset locations are equivalent but not identical. Locations may either be read-only or read-write. Typically, multiple read-write locations are backed by a clustered filesystem while read-only locations are replicas created by a federation-initiated or external replication operation. Read-only locations may represent consistent point-in-time copies of a read-write location. The federation protocols, however, cannot prevent subsequent changes to a read-only location nor guarantee point-in-time consistency of a read-only location if the read-write location is changing.

Regardless of the type, all locations exist at the same mount point in the namespace and, thus, one client may be referred to one location while another is directed to a different location. Since updates to each fileset location are not controlled by the federation protocol, it is the responsibility of administrators to guarantee the functional equivalence of the data.

The federation protocol does not guarantee that the different locations are mutually consistent in terms of the currency of the data. It relies on the client file-access protocol (e.g., NFSv4) to

contain sufficient information to help the clients determine the currency of the data at each location in order to ensure that the clients do not revert back in time when switching locations.

2.8.2. Caching of Fileset Locations

To resolve an FSN to a set of FSL records, the fileserver queries the appropriate NSDB for the FSL records. A fileserver MAY cache these FSL records for a limited period of time. The period of time, if any, during which FSL records MAY be cached is indicated by the FSL's TTL field.

The combination of FSL caching and FSL migration presents a challenge. For example, suppose there are three fileservers named A, B, and C and fileserver A contains a junction to fileset X stored on fileserver B. Now suppose that fileset X is migrated from fileserver B to fileserver C and the corresponding FSL information for fileset X in the appropriate NSDB is updated. If fileserver A has a cached FSL for fileset X, a user traversing the junction on fileserver A will be referred to fileserver B even though fileset X has migrated to fileserver C. If fileserver A had not cached the FSL record, it would have queried the NSDB and obtained the correct location of fileset X.

Administrators are advised to be aware of FSL caching when performing a migration. When migrating a fileset, administrators SHOULD create a junction at the fileset's old location referring back to the NSDB entry for the fileset. This junction will redirect any users who follow stale FSL information to the correct location. Thus, in the above example, fileserver A would direct clients to fileserver B, but fileserver B would in turn direct clients to fileserver C.

Such supplemental junctions (on fileserver B in the example) would not be required to be in place forever. They need to stay in place only until cached FSL entries for the target fileset are invalidated. Each FSL contains a TTL field, a count in seconds of the time interval the FSL MAY be cached. This is an upper bound for the lifetime of the cached information and a lower bound for the lifetime of the supplemental junctions. For example, suppose this field contains the value 3600 seconds (one hour). In such a case, administrators MUST keep the supplemental junctions in place for at least one hour after the fileset move has taken place, and FSL data MUST NOT be cached by a referring fileserver for more than one hour without a refresh.

2.8.3. Generating A Referral from Fileset Locations

After resolving an FSN to a set of FSL records, the fileserver can generate a referral to redirect the client to one or more of the

FSLs. The fileserver will convert the FSL records to a referral format understood by the client, such as an NFSv4 fs_locations attribute or NFSv4.1 fs_locations_info attribute.

In order to give the client as many options as possible, the fileserver SHOULD include the maximum possible number of FSL records in a referral. However, the fileserver MAY omit some of the FSL records from the referral. For example, the fileserver might omit an FSL record with a different file access protocol from the one in use between the fileserver and client, or the fileserver might omit an FSL record because of limitations in the file access protocol's referral format, or the fileserver might omit an FSL record based on some other criteria.

For a given FSL record, the fileserver MAY convert or reduce the FSL record's contents in a manner appropriate to the referral format. For example, an NFS FSL record contains all the data necessary to construct an NFSv4.1 fs_locations_info attribute, but an NFSv4.1 fs_locations_info attribute contains several pieces of information that are not found in an NFSv4 fs_locations attribute. A fileserver will construct entries in an NFSv4 fs_locations attribute using the relevant contents of an NFS FSL record. Whenever the fileserver converts or reduces FSL data, the fileserver SHOULD attempt to maintain the original meaning where possible. For example, an NFS FSL record contains the rank and order information that is included in an NFSv4.1 fs_locations_info attribute (see NFSv4.1's FSLI4BX_READRANK, FSLI4BX_READORDER, FSLI4BX_WRITERANK, and FSLI4BX_WRITEORDER). While this rank and order information is not explicitly expressible in an NFSv4 fs_locations attribute, the fileserver can arrange the NFSv4 fs_locations attribute's locations list base on the rank and order values.

2.9. Namespace Database (NSDB)

The NSDB service is a federation-wide service that provides interfaces to define, update, and query FSN information, FSL information, and FSN to FSL mapping information. An individual repository of namespace information is called an NSDB node. Each NSDB node is managed by a single administrative entity. A single admin entity can manage multiple NSDB nodes.

The difference between the NSDB service and an NSDB node is analogous to that between the DNS service and a particular DNS server.

Each NSDB node stores the definition of the FSNs for which it is authoritative. It also stores the definitions of the FSLs associated with those FSNs. An NSDB node is authoritative for the filesets that it defines. An NSDB node can cache information from a peer NSDB

node. The fileserver can always contact a local NSDB node (if it has been defined) or directly contact any NSDB node to resolve a junction. Each NSDB node supports an LDAP [RFC4510] interface and can be accessed by an LDAP client.

An NSDB MAY be replicated throughout the federation. If an NSDB is replicated, the NSDB MUST exhibit loose, converging consistency as defined in [RFC3254]. The mechanism by which this is achieved is outside the scope of this document. Many LDAP implementations support replication. These features MAY be used to replicate the NSDB.

2.10. Mount Points, Junctions and Referrals

A mount point is a directory in a parent fileset where a target fileset may be attached. If a client traverses the path leading from the root of the namespace to the mount point of a target fileset it should be able to access the data in that target fileset (assuming appropriate permissions).

The directory where a fileset is mounted is represented by a junction in the underlying filesystem. In other words, a junction can be viewed as a reference from a directory in one fileset to the root of the target fileset. A junction can be implemented as a special marker on a directory that is interpreted by the fileserver as a mount point, or by some other mechanism in the underlying filesystem.

What data is used by the underlying filesystem to represent the junction is not defined by this protocol. The essential property is that the server must be able to find, given the junction, the FSN for the target fileset. The mechanism by which the server maps a junction to an FSN is outside the scope of this document. The FSN (as described earlier) contains the authoritative NSDB node, the optional NSDB search base if one is defined, and the FsnUuid (a UUID for the fileset).

When a client traversal reaches a junction, the client is referred to a list of FSLs associated with the FSN targeted by the junction. The client can then redirect its connection to one of the FSLs. This act is called a referral. For NFSv4 and NFSv4.1 clients, the FSL information is returned in the fs_locations and fs_locations_info attributes respectively.

The federation protocols do not limit where and how many times a fileset is mounted in the namespace. Filesets can be nested; a fileset can be mounted under another fileset.

2.11. Unified Namespace and the Root Fileset

The root fileset, when defined, is the top-level fileset of the federation-wide namespace. The root of the unified namespace is the top level directory of this fileset. A set of designated fileservers in the federation can export the root fileset to render the federation-wide unified namespace. When a client mounts the root fileset from any of these designated fileservers it can view a common federation-wide namespace. The root fileset could be implemented either as an exported NFS file system or as data in the NSDB itself. The properties and schema definition of an NSDB-based root fileset and the protocol details that describe how to configure and replicate the root fileset are not defined in this document.

3. Examples

In this section we provide examples and discussion of the basic operations facilitated by the federated filesystem protocol: creating a fileset, adding a replica of a fileset, resolving a junction, and creating a junction.

3.1. Creating a Fileset and its FSL(s)

A fileset is the abstraction of a set of files and the directory tree that contains them. The fileset abstraction is the fundamental unit of data management in the federation. This abstraction is implemented by an actual directory tree whose root location is specified by a fileset location (FSL).

In this section, we describe the basic requirements for starting with a directory tree and creating a fileset that can be used in the federation protocols. Note that we do not assume that the process of creating a fileset requires any transformation of the files or the directory hierarchy. The only thing that is required by this process is assigning the fileset a fileset name (FSN) and expressing the location of the implementation of the fileset as an FSL.

There are many possible variations to this procedure, depending on how the FSN that binds the FSL is created, and whether other replicas of the fileset exist, are known to the federation, and need to be bound to the same FSN.

It is easiest to describe this in terms of how to create the initial implementation of the fileset, and then describe how to add replicas.

3.1.1. Creating a Fileset and an FSN

1. Choose the NSDB node that will keep track of the FSL(s) and related information for the fileset.

2. Create an FSN in the NSDB node.

   The FSN UUID is chosen by the administrator or generated automatically by administration software. The former case is used if the fileset is being restored, perhaps as part of disaster recovery, and the administrator wishes to specify the FSN UUID in order to permit existing junctions that reference that FSN to work again.

   At this point, the FSN exists, but its fileset locations are unspecified.

3. For the FSN created above, create an FSL with the appropriate information in the NSDB node.

3.1.2. Adding a Replica of a Fileset

Adding a replica is straightforward: the NSDB node and the FSN are already known. The only remaining step is to add another FSL.

Note that the federation protocols only provide the mechanisms to register and unregister replicas of a fileset. Fileserver-to-fileserver replication protocols are not defined.

3.2. Junction Resolution

A fileset may contain references to other filesets. These references are represented by junctions. If a client requests access to a fileset object that is a junction, the fileserver resolves the junction to discover one or more FSLs that implement the referenced fileset.

There are many possible variations to this procedure, depending on how the junctions are represented by the fileserver and how the fileserver performs junction resolution.

Step 4 is the only step that interacts directly with the federation protocols. The rest of the steps may use platform-specific interfaces.

1. The fileserver determines that the object being accessed is a junction.

2. The fileserver does a local lookup to find the FSN of the target fileset.

3. Using the FSN, the fileserver finds the NSDB node responsible for the target FSN.

4. The fileserver contacts that NSDB node and asks for the set of FSLs that implement the target FSN. The NSDB node responds with a (possibly empty) set of FSLs.

5. The fileserver converts one or more of the FSLs to the location type used by the client (e.g., a Network File System (NFSv4) fs_location, as described in [3530bis]).

6. The fileserver redirects (in whatever manner is appropriate for the client) the client to the location(s).

3.3. Example Use Cases for Fileset Annotations

Fileset annotations MAY be used to convey additional attributes of a fileset

For example, fileset annotations can be used to define relationships between filesets that can be used by an auxiliary replication protocol. Consider the scenario where a fileset is created and mounted at some point in the namespace. A snapshot of the read-write FSL of that fileset is taken periodically at different frequencies say a daily snapshot or a weekly snapshot. The different snapshots are mounted at different locations in the namespace. The daily snapshots are considered as a different fileset from the weekly ones but both are related to the source fileset. For this we can define an annotation labeling the filesets as source and replica. The replication protocol can use this information to copy data from one or more FSLs of the source fileset to all the FSLs of the replica fileset. The replica filesets are read-only while the source fileset is read-write.

This follows the traditional Andrew File System (AFS) model of mounting the read-only volume at a path in the namespace different from that of the read-write volume [AFS].

The federation protocol does not control or manage the relationship among filesets. It merely enables annotating the filesets with user-defined relationships.

Another potential use for annotations is recording references to an FSN. A single annotation containing the number of references could be defined or multiple annotations, one per reference, could be used

to store detailed information on the location of each reference. As with the replication annotation described above, the maintenance of reference information would not be controlled by the federation protocol. The information would mostly likely be non-authoritative because the the ability to create a junction does not require the authority to update the FSN record. In any event, such annotations could be useful to administrators for determining if an FSN is referenced by a junction.

4. NSDB Configuration and Schema

This section describes how an NSDB is constructed using an LDAP Version 3 [RFC4510] Directory. Section 4.1 describes the basic properties of the LDAP configuration that MUST be used in order to ensure compatibility between different implementations. Section 4.2 defines the new LDAP attribute types, the new object types, and specifies how the distinguished name (DN) of each object instance MUST be constructed.

4.1. LDAP Configuration

An NSDB is constructed using an LDAP Directory. This LDAP Directory MAY have multiple naming contexts. For each naming context, the LDAP Directory's root DSE will have a namingContext attribute. Each namingContext attribute contains the DN of the naming context's root entry. For each naming context that contains federation entries (e.g. FSNs and FSLs):

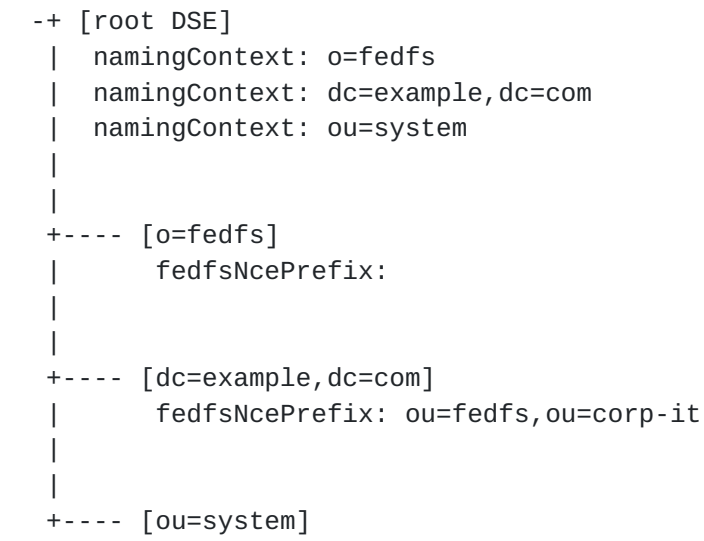
1. There MUST be an LDAP entry that is superior to all of the naming context's federation entries in the Directory Information Tree (DIT) This entry is termed the NSDB Container Entry (NCE). The NCE's children are FSNs. An FSNs children are FSLs.

2. The naming context's root entry MUST include the fedfsNsdbContainerInfo (defined below) as one of its object classes. The fedfsNsdbContainerInfo's fedfsNcePrefix attribute is used to locate the naming context's NCE.

If a naming context does not contain federation entries, it will not contain an NCE and its root entry will not include a fedfsNsdbContainerInfo as one of its object classes.

A fedfsNsdbContainerInfo's fedfsNcePrefix attribute contains a string. Prepending this string to the namingContext value produces the Distinguished Name (DN) of the NSDB Container Entry. An empty fedfsNcePrefix string value indicates that the NSDB Container Entry is the namingContext's root entry.

For example, an LDAP directory might have the following entries:



In this case, the o=fedfs namingContext has an NSBD Container Entry at o=fedfs, the dc=example,dc=com namingContext has an NSDB Container Entry at ou=fedfs,ou=corp-it,dc=example,dc=com, and the ou=system namingContext has no NSDB Container Entry.

The NSDB SHOULD be configured with one or more privileged LDAP users. These users are able to modify the contents of the LDAP database. An administrator that performs the operations described in Section 5.1 SHOULD authenticate using the DN of a privileged LDAP user.

It MUST be possible for an unprivileged (unauthenticated) user to perform LDAP queries that access the NSDB data. A fileserver performs the operations described in Section 5.2 as an unprivileged user.

All implementations SHOULD use the same schema, or, at minimum, a schema that includes all of the objects, with each of the attributes, named in the following sections.

Given the above configuration guidelines, an NSDB SHOULD be constructed using a dedicated LDAP directory. Separate LDAP directories are RECOMMENDED for other purposes, such as storing user account information. By using an LDAP directory dedicated to storing NSDB records, there is no need to disturb the configuration of any other LDAP directories that store information unrelated to an NSDB.

4.2. LDAP Schema

The schema definitions provided in this document use the LDAP schema syntax defined in [RFC4512]. The definitions are formatted to allow

the reader to easily extract them from the document. The reader can use the following shell script to extract the definitions:

```
<CODE BEGINS>

#!/bin/sh
grep '^ *///' | sed 's?^ */// ??' | sed 's?^ *///$??'

<CODE ENDS>
```

If the above script is stored in a file called "extract.sh", and this document is in a file called "spec.txt", then the reader can do:

```
<CODE BEGINS>

sh extract.sh < spec.txt > fedfs.schema

<CODE ENDS>
```

The effect of the script is to remove leading white space from each line, plus a sentinel sequence of "///".

As stated above, code components extracted from this document must include the following license:

<CODE BEGINS>

```
/// #
/// # Copyright (c) 2010 IETF Trust and the persons identified
/// # as authors of the code.  All rights reserved.
/// #
/// # The authors of the code are the authors of
/// # [draft-ietf-nfsv4-federated-fs-protocol-xx.txt]: J. Lentini,
/// # C. Everhart, D. Ellard, R. Tewari, and M. Naik.
/// #
/// # Redistribution and use in source and binary forms, with
/// # or without modification, are permitted provided that the
/// # following conditions are met:
/// #
/// # - Redistributions of source code must retain the above
/// #   copyright notice, this list of conditions and the
/// #   following disclaimer.
/// #
/// # - Redistributions in binary form must reproduce the above
/// #   copyright notice, this list of conditions and the
/// #   following disclaimer in the documentation and/or other
/// #   materials provided with the distribution.
/// #
/// # - Neither the name of Internet Society, IETF or IETF
/// #   Trust, nor the names of specific contributors, may be
/// #   used to endorse or promote products derived from this
/// #   software without specific prior written permission.
/// #
/// #   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS
/// #   AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED
/// #   WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
/// #   IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
/// #   FOR A PARTICULAR PURPOSE ARE DISCLAIMED.  IN NO
/// #   EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
/// #   LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
/// #   EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
/// #   NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
/// #   SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
/// #   INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
/// #   LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
/// #   OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
/// #   IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
/// #   ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
/// #
```

<CODE ENDS>

4.2.1. LDAP Attributes

This section describes the required attributes of the NSDB LDAP schema. The following definitions are used below:

- The "name" attribute described in [RFC4519].
- The Integer syntax (1.3.6.1.4.1.1466.115.121.1.27) described in [RFC4517].
- The "integerMatch" rule described in [RFC4517].
- The Octet String syntax (1.3.6.1.4.1.1466.115.121.1.40) described in [RFC4517].
- The "octetStringMatch" rule described in [RFC4517].
- The Boolean syntax (1.3.6.1.4.1.1466.115.121.1.7) described in [RFC4517].
- The "booleanMatch" rule described in [RFC4517].
- The "distinguishedNameMatch" rule described in [RFC4517].
- The DN syntax (1.3.6.1.4.1.1466.115.121.1.12) described in [RFC4517].

4.2.1.1. fedfsUuid

A fedfsUuid is the base type for all of the universally unique identifiers (UUIDs) used by the federated filesystem protocols.

To minimize the probability of two UUIDs colliding, a consistent procedure for generating UUIDs SHOULD be used throughout a federation. Within a federation, UUIDs SHOULD be generated using the procedure described for version 1 of the UUID variant specified in [RFC4122]. This is the time-based UUID variant provided by many UUID programming libraries (e.g., the OSF DCE uuid_generate_time(1) API).

The UUID's text representation (as defined in [RFC4122]) SHOULD be encoded as a UTF-8 string.

A fedfsUuid is a single-valued LDAP attribute.

<CODE BEGINS>

```
        ///
        /// attributetype (
        ///     1.3.6.1.4.1.31103.1.1 NAME 'fedfsUuid'
        ///     DESC 'A UUID used by NSDB'
        ///     SUP name
        ///     SINGLE-VALUE
        ///     )
        ///
```

<CODE ENDS>

4.2.1.2. fedfsNetAddr

A fedfsNetAddr is the locative name of a network service in either IPv4, IPv6, or DNS name notation. It MUST be a UTF-8 string and SHOULD be prepared using the server4 rules defined in Chapter 12 "Internationalization" of [3530bis].

An IPv4 address MUST be represented using the standard dotted decimal format defined by the IPv4address rule in Section 3.2.2 of RFC 3986 [RFC3986]. An IPv6 address MUST be represented using the format defined in Section 2.2 of RFC 4291 [RFC4291].

A DNS name MUST be represented using a fully qualified domain name. A system (i.e. fileserver or administrative host) SHOULD resolve the fully qualified domain name to a network address using the system's standard resolution mechanisms.

This attribute is single-valued.

<CODE BEGINS>

```
        ///
        /// attributetype (
        ///     1.3.6.1.4.1.31103.1.2 NAME 'fedfsNetAddr'
        ///     DESC 'The network name of a host or service'
        ///     SUP name
        ///     SINGLE-VALUE
        ///     )
        ///
```

<CODE ENDS>

4.2.1.3. fedfsNetPort

A fedfsNetPort is the decimal representation of a transport service's port number. A fedfsNetPort MUST be encoded as an Integer syntax value [RFC4517].

This attribute is single-valued.

<CODE BEGINS>

```
        ///
        /// attributetype (
        ///     1.3.6.1.4.1.31103.1.3 NAME 'fedfsNetPort'
        ///     DESC 'A transport port number of a service'
        ///     EQUALITY integerMatch
        ///     SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
        ///     SINGLE-VALUE
        ///     )
        ///
```

<CODE ENDS>

4.2.1.4. fedfsFsnUuid

A fedfsFsnUuid represents the UUID component of an FSN. An NSDB SHOULD ensure that no two FSNs it stores have the same fedfsFsnUuid.

The fedfsFsnUuid is a subclass of fedfsUuid, with the same encoding rules.

This attribute is single-valued.

<CODE BEGINS>

```
        ///
        /// attributetype (
        ///     1.3.6.1.4.1.31103.1.4 NAME 'fedfsFsnUuid'
        ///     DESC 'The FSN UUID component of an FSN'
        ///     SUP fedfsUuid
        ///     SINGLE-VALUE
        ///     )
        ///
```

<CODE ENDS>

4.2.1.5. fedfsNsdbName

A fedfsNsdbName is the NSDB component of an FSN.

It MUST be a UTF-8 string containing a DNS name. The DNS name MUST be represented using a fully qualified domain name. A system (i.e. fileserver or administrative host) SHOULD resolve the fully qualified domain name to a network address using the system's standard resolution mechanisms.

FSNs are immutable and invariant. The attributes of an FSN, including the fedfsNsdbName, are expected to remain constant. Therefore, a fedfsNsdbName SHOULD NOT contain a network address, such as an IPv4 or IPv6 address, as this would indefinitely assign the network address.

This attribute is single-valued.

<CODE BEGINS>

```
        ///
        /// attributetype (
        ///     1.3.6.1.4.1.31103.1.5 NAME 'fedfsNsdbName'
        ///     DESC 'The NSDB node component of an FSN'
        ///     SUP name
        ///     SINGLE-VALUE
        ///     )
        ///
```

<CODE ENDS>

4.2.1.6. fedfsNsdbPort

A fedfsNsdbPort is the decimal representation of an NSDB's port number. The fedfsNsdbPort attribute is a subclass of fedfsNetPort, with the same encoding rules.

This attribute is single-valued.

<CODE BEGINS>

```
        ///
        /// attributetype (
        ///     1.3.6.1.4.1.31103.1.6 NAME 'fedfsNsdbPort'
        ///     DESC 'The transport port number of an NSDB'
        ///     SUP fedfsNetPort
        ///     SINGLE-VALUE
        ///     )
        ///
```

<CODE ENDS>

4.2.1.7. fedfsNcePrefix

A fedfsNcePrefix stores a distinguished name (DN) prefix.

This attribute is single-valued.

<CODE BEGINS>

```
        ///
        /// attributetype (
        ///     1.3.6.1.4.1.31103.1.7 NAME 'fedfsNcePrefix'
        ///     DESC 'NCE prefix'
        ///     EQUALITY distinguishedNameMatch
        ///     SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
        ///     SINGLE-VALUE
        ///     )
        ///
```

<CODE ENDS>

OID 1.3.6.1.4.1.1466.115.121.1.12 is the DN syntax [RFC4517].

4.2.1.8. fedfsFslUuid

A fedfsFslUuid represents the UUID of an FSL. An NSDB SHOULD ensure that no two FSLs it stores have the same fedfsFslUuid.

The fedfsFslUuid attribute is a subclass of fedfsUuid, with the same encoding rules.

This attribute is single-valued.

<CODE BEGINS>

```
        ///
        /// attributetype (
        ///     1.3.6.1.4.1.31103.1.8 NAME 'fedfsFslUuid'
        ///     DESC 'UUID of an FSL'
        ///     SUP fedfsUuid
        ///     SINGLE-VALUE
        ///     )
        ///
```

<CODE ENDS>

4.2.1.9. fedfsFslHost

A fedfsFslHost is the host component of an FSL. The fedfsFslHost attribute is a subclass of fedfsNetAddr, with the same encoding rules.

This attribute is single-valued.

<CODE BEGINS>

```
        ///
        /// attributetype (
        ///     1.3.6.1.4.1.31103.1.9 NAME 'fedfsFslHost'
        ///     DESC 'Service location for a fileserver'
        ///     SUP fedfsNetAddr
        ///     SINGLE-VALUE
        ///     )
        ///
```

<CODE ENDS>

4.2.1.10. fedfsFslPort

A fedfsFslPort is the decimal representation of a file service's port number. The fedfsFslPort attribute is a subclass of fedfsNetPort, with the same encoding rules.

This attribute is single-valued.

<CODE BEGINS>

```
        ///
        /// attributetype (
        ///     1.3.6.1.4.1.31103.1.10 NAME 'fedfsFslPort'
        ///     DESC 'The file service transport port number'
        ///     SUP fedfsNetPort
        ///     SINGLE-VALUE
        ///     )
        ///
```

<CODE ENDS>

4.2.1.11. fedfsFslTTL

A fedfsFslTTL is the amount of time in seconds an FSL SHOULD be cached by a fileserver. A fedfsFslTTL MUST be encoded as an Integer syntax value [RFC4517].

This attribute is single-valued.

<CODE BEGINS>

```
          ///
          /// attributetype (
          ///     1.3.6.1.4.1.31103.1.11 NAME 'fedfsFslTTL'
          ///     DESC 'Time to live of an FSL'
          ///     EQUALITY integerMatch
          ///     SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
          ///     SINGLE-VALUE
          ///     )
          ///
```

<CODE ENDS>

OID 1.3.6.1.4.1.1466.115.121.1.27 is the Integer syntax [RFC4517].

4.2.1.12. fedfsAnnotation

A fedfsAnnotation contains an object annotation formatted as a key/value pair.

This attribute is multi-valued; an object type that permits annotations may have any number of annotations per instance.

A fedfsAnnotation attribute is a human-readable sequence of UTF-8 characters with no non-terminal NUL characters. The value MUST be formatted according to the following ABNF [RFC5234] rules:

```
ANNOTATION = KEY EQUALS VALUE
KEY = ITEM
VALUE = ITEM
ITEM = BLANK DQUOTE STR DQUOTE BLANK
BLANK = 0*EMPTY
EMPTY = SPACE / HTAB
HTAB = %x09 ; horizontal tab
STR = 0*UTF8
```

The DQUOTE, EQUALS, UTF8, and SPACE rules are defined in [RFC4512].

The following escape sequences are allowed:

escape sequence	replacement
\\	\
\"	"

A fedfsAnnotation value SHOULD be processed as follows:

1. Parse the attribute value according to the ANNOTATION rule, ignoring the escape sequences above.

2. Scan through results of the previous step and replace the escape sequences above.

A fedfsAnnotation attribute that does not adhere to this format SHOULD be ignored.

The following are examples of valid fedfsAnnotation attributes:

```
"key1" = "foo"
"another key" = "x=3"
"key-2" = "A string with \" and \\ characters."
```

which correspond to the following key/value pairs:

key	value
key1	foo
another key	x=3
key-2	A string with " and \ characters.

<CODE BEGINS>

```
///
/// attributetype (
///     1.3.6.1.4.1.31103.1.12 NAME 'fedfsAnnotation'
///     DESC 'Annotation of an object'
///     SUP name
///     )
///
```

<CODE ENDS>

4.2.1.13. fedfsDescr

A fedfsDescr stores an object description. The description MUST be encoded as a UTF-8 string.

This attribute is multi-valued which permits any number of descriptions per entry.

<CODE BEGINS>

```
        ///
        /// attributetype (
        ///     1.3.6.1.4.1.31103.1.13 NAME 'fedfsDescr'
        ///     DESC 'Description of an object'
        ///     SUP name
        ///     )
        ///
```

<CODE ENDS>

4.2.1.14. fedfsNfsPath

A fedfsNfsPath is the path attribute of an FSL. The path MUST be the XDR encoded NFS path as defined by the NFS pathname4 XDR type of the fs_location's rootpath [3530bis] and the fs_locations_item's fli_rootpath [RFC5661]. The NFS pathname4 XDR type is a variable length array of component4 elements. The NFS component4 XDR type is a variable length array of opaque data. A fedfsNfsPath attribute's component4 elements SHOULD be prepared using the component4 rules defined in Chapter 12 "Internationalization" of [3530bis].

This attribute is single-valued.

<CODE BEGINS>

```
        ///
        /// attributetype (
        ///     1.3.6.1.4.1.31103.1.100 NAME 'fedfsNfsPath'
        ///     DESC 'Server-local path to a fileset'
        ///     EQUALITY octetStringMatch
        ///     SYNTAX 1.3.6.1.4.1.1466.115.121.1.40
        ///     SINGLE-VALUE
        ///     )
        ///
```

<CODE ENDS>

OID 1.3.6.1.4.1.1466.115.121.1.40 is the Octet String syntax [RFC4517].

4.2.1.15. fedfsNfsMajorVer

A fedfsNfsMajorVer contains the NFS major version of the associated NFS FSL. A fedfsNfsMajorVer MUST be encoded as an Integer syntax value [RFC4517].

For example if the FSL was exported via NFS 4.1, the contents of this attribute would be the value 4.

This attribute is single-valued.

<CODE BEGINS>

```
        ///
        /// attributetype (
        ///     1.3.6.1.4.1.31103.1.101 NAME 'fedfsNfsMajorVer'
        ///     DESC 'NFS major version'
        ///     EQUALITY integerMatch
        ///     SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
        ///     SINGLE-VALUE
        ///     )
        ///
```

<CODE ENDS>

OID 1.3.6.1.4.1.1466.115.121.1.27 is the Integer syntax [RFC4517].

4.2.1.16. fedfsNfsMinorVer

A fedfsNfsMinorVer contain the NFS minor version of the associated NFS FSL. A fedfsNfsMinorVer MUST be encoded as an Integer syntax value [RFC4517].

For example if the FSL was exported via NFS 4.1, the contents of this attribute would be the value 1.

This attribute is single-valued.

<CODE BEGINS>

```
        ///
        /// attributetype (
        ///     1.3.6.1.4.1.31103.1.102 NAME 'fedfsNfsMinorVer'
        ///     DESC 'NFS minor version'
        ///     EQUALITY integerMatch
        ///     SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
        ///     SINGLE-VALUE
        ///     )
        ///
```

<CODE ENDS>

OID 1.3.6.1.4.1.1466.115.121.1.27 is the Integer syntax [RFC4517].

4.2.1.17. fedfsNfsCurrency

A fedfsNfsCurrency stores the NFSv4.1 fs_locations_server's fls_currency value [RFC5661]. A fedfsNfsCurrency MUST be encoded as an Integer syntax value [RFC4517] in the range [-2147483648, 2147483647].

This attribute is single-valued.

<CODE BEGINS>

```
        ///
        /// attributetype (
        ///     1.3.6.1.4.1.31103.1.103 NAME 'fedfsNfsCurrency'
        ///     DESC 'up-to-date measure of the data'
        ///     EQUALITY integerMatch
        ///     SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
        ///     SINGLE-VALUE
        ///     )
        ///
```

<CODE ENDS>

OID 1.3.6.1.4.1.1466.115.121.1.27 is the Integer syntax [RFC4517].

4.2.1.18. fedfsNfsGenFlagWritable

A fedfsNfsGenFlagWritable stores the value of an FSL's NFSv4.1 FSLI4GF_WRITABLE bit [RFC5661]. A value of "TRUE" indicates the bit is true. A value of "FALSE" indicates the bit is false.

<CODE BEGINS>

```
       ///
       /// attributetype (
       ///     1.3.6.1.4.1.31103.1.104 NAME 'fedfsNfsGenFlagWritable'
       ///     DESC 'Indicates if the filesystem is writable'
       ///     EQUALITY booleanMatch
       ///     SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
       ///     SINGLE-VALUE
       ///     )
       ///
```

<CODE ENDS>

OID 1.3.6.1.4.1.1466.115.121.1.7 is the Boolean syntax [RFC4517].

4.2.1.19. fedfsNfsGenFlagGoing

A fedfsNfsGenFlagGoing stores the value of an FSL's NFSv4.1 FSLI4GF_GOING bit [RFC5661]. A value of "TRUE" indicates the bit is true. A value of "FALSE" indicates the bit is false.

<CODE BEGINS>

```
        ///
        /// attributetype (
        ///     1.3.6.1.4.1.31103.1.105 NAME 'fedfsNfsGenFlagGoing'
        ///     DESC 'Indicates if the filesystem is going'
        ///     EQUALITY booleanMatch
        ///     SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
        ///     SINGLE-VALUE
        ///     )
        ///
```

<CODE ENDS>

OID 1.3.6.1.4.1.1466.115.121.1.7 is the Boolean syntax [RFC4517].

4.2.1.20. fedfsNfsGenFlagSplit

A fedfsNfsGenFlagSplit stores the value of an FSL's NFSv4.1 FSLI4GF_SPLIT bit [RFC5661]. A value of "TRUE" indicates the bit is true. A value of "FALSE" indicates the bit is false.

<CODE BEGINS>

```
        ///
        /// attributetype (
        ///     1.3.6.1.4.1.31103.1.106 NAME 'fedfsNfsGenFlagSplit'
        ///     DESC 'Indicates if there are multiple filesystems'
        ///     EQUALITY booleanMatch
        ///     SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
        ///     SINGLE-VALUE
        ///     )
        ///
```

<CODE ENDS>

OID 1.3.6.1.4.1.1466.115.121.1.7 is the Boolean syntax [RFC4517].

4.2.1.21. fedfsNfsTransFlagRdma

A fedfsNfsTransFlagRdma stores the value of an FSL's NFSv4.1 FSLI4TF_RDMA bit [RFC5661]. A value of "TRUE" indicates the bit is

true. A value of "FALSE" indicates the bit is false.

<CODE BEGINS>

```
        ///
        /// attributetype (
        ///     1.3.6.1.4.1.31103.1.107 NAME 'fedfsNfsTransFlagRdma'
        ///     DESC 'Indicates if the transport supports RDMA'
        ///     EQUALITY booleanMatch
        ///     SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
        ///     SINGLE-VALUE
        ///     )
        ///
```

<CODE ENDS>

OID 1.3.6.1.4.1.1466.115.121.1.7 is the Boolean syntax [RFC4517].

4.2.1.22. fedfsNfsClassSimul

A fedfsNfsClassSimul contains the FSL's NFSv4.1 FSLI4BX_CLSIMUL [RFC5661] value. A fedfsNfsClassSimul MUST be encoded as an Integer syntax value [RFC4517] in the range [0, 255].

<CODE BEGINS>

```
        ///
        /// attributetype (
        ///     1.3.6.1.4.1.31103.1.108 NAME 'fedfsNfsClassSimul'
        ///     DESC 'The simultaneous-use class of the filesystem'
        ///     EQUALITY integerMatch
        ///     SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
        ///     SINGLE-VALUE
        ///     )
        ///
```

<CODE ENDS>

OID 1.3.6.1.4.1.1466.115.121.1.27 is the Integer syntax [RFC4517].

4.2.1.23. fedfsNfsClassHandle

A fedfsNfsClassHandle contains the FSL's NFSv4.1 FSLI4BX_CLHANDLE [RFC5661] value. A fedfsNfsClassHandle MUST be encoded as an Integer syntax value [RFC4517] in the range [0, 255].

<CODE BEGINS>

```
        ///
        /// attributetype (
        ///     1.3.6.1.4.1.31103.1.109 NAME 'fedfsNfsClassHandle'
        ///     DESC 'The handle class of the filesystem'
        ///     EQUALITY integerMatch
        ///     SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
        ///     SINGLE-VALUE
        ///     )
        ///
```

<CODE ENDS>

OID 1.3.6.1.4.1.1466.115.121.1.27 is the Integer syntax [RFC4517].

4.2.1.24. fedfsNfsClassFileid

A fedfsNfsClassFileid contains the FSL's NFSv4.1 FSLI4BX_CLFILEID [RFC5661] value. A fedfsNfsClassFileid MUST be encoded as an Integer syntax value [RFC4517] in the range [0, 255].

<CODE BEGINS>

```
        ///
        /// attributetype (
        ///     1.3.6.1.4.1.31103.1.110 NAME 'fedfsNfsClassFileid'
        ///     DESC 'The fileid class of the filesystem'
        ///     EQUALITY integerMatch
        ///     SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
        ///     SINGLE-VALUE
        ///     )
        ///
```

<CODE ENDS>

OID 1.3.6.1.4.1.1466.115.121.1.27 is the Integer syntax [RFC4517].

4.2.1.25. fedfsNfsClassWritever

A fedfsNfsClassWritever contains the FSL's NFSv4.1 FSLI4BX_CLWRITEVER [RFC5661] value. A fedfsNfsClassWritever MUST be encoded as an Integer syntax value [RFC4517] in the range [0, 255].

<CODE BEGINS>

```
        ///
        /// attributetype (
        ///     1.3.6.1.4.1.31103.1.111 NAME 'fedfsNfsClassWritever'
        ///     DESC 'The write-verifier class of the filesystem'
        ///     EQUALITY integerMatch
        ///     SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
        ///     SINGLE-VALUE
        ///     )
        ///
```

<CODE ENDS>

OID 1.3.6.1.4.1.1466.115.121.1.27 is the Integer syntax [RFC4517].

4.2.1.26. fedfsNfsClassChange

A fedfsNfsClassChange contains the FSL's NFSv4.1 FSLI4BX_CLCHANGE [RFC5661] value. A fedfsNfsClassChange MUST be encoded as an Integer syntax value [RFC4517] in the range [0, 255].

<CODE BEGINS>

```
        ///
        /// attributetype (
        ///     1.3.6.1.4.1.31103.1.112 NAME 'fedfsNfsClassChange'
        ///     DESC 'The change class of the filesystem'
        ///     EQUALITY integerMatch
        ///     SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
        ///     SINGLE-VALUE
        ///     )
        ///
```

<CODE ENDS>

OID 1.3.6.1.4.1.1466.115.121.1.27 is the Integer syntax [RFC4517].

4.2.1.27. fedfsNfsClassReaddir

A fedfsNfsClassReaddir contains the FSL's NFSv4.1 FSLI4BX_CLREADDIR [RFC5661] value. A fedfsNfsClassReaddir MUST be encoded as an Integer syntax value [RFC4517] in the range [0, 255].

<CODE BEGINS>

```
        ///
        /// attributetype (
        ///     1.3.6.1.4.1.31103.1.113 NAME 'fedfsNfsClassReaddir'
        ///     DESC 'The readdir class of the filesystem'
        ///     EQUALITY integerMatch
        ///     SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
        ///     SINGLE-VALUE
        ///     )
        ///
```

<CODE ENDS>

OID 1.3.6.1.4.1.1466.115.121.1.27 is the Integer syntax [RFC4517].

4.2.1.28. fedfsNfsReadRank

A fedfsNfsReadRank contains the FSL's NFSv4.1 FSLI4BX_READRANK [RFC5661] value. A fedfsNfsReadRank MUST be encoded as an Integer syntax value [RFC4517] in the range [0, 255].

<CODE BEGINS>

```
        ///
        /// attributetype (
        ///     1.3.6.1.4.1.31103.1.114 NAME 'fedfsNfsReadRank'
        ///     DESC 'The read rank of the filesystem'
        ///     EQUALITY integerMatch
        ///     SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
        ///     SINGLE-VALUE
        ///     )
        ///
```

<CODE ENDS>

OID 1.3.6.1.4.1.1466.115.121.1.27 is the Integer syntax [RFC4517].

4.2.1.29. fedfsNfsReadOrder

A fedfsNfsReadOrder contains the FSL's NFSv4.1 FSLI4BX_READORDER [RFC5661] value. A fedfsNfsReadOrder MUST be encoded as an Integer syntax value [RFC4517] in the range [0, 255].

<CODE BEGINS>

```
        ///
        /// attributetype (
        ///     1.3.6.1.4.1.31103.1.115 NAME 'fedfsNfsReadOrder'
        ///     DESC 'The read order of the filesystem'
        ///     EQUALITY integerMatch
        ///     SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
        ///     SINGLE-VALUE
        ///     )
        ///
```

<CODE ENDS>

OID 1.3.6.1.4.1.1466.115.121.1.27 is the Integer syntax [RFC4517].

4.2.1.30. fedfsNfsWriteRank

A fedfsNfsWriteRank contains the FSL's FSLI4BX_WRITERANK [RFC5661] value. A fedfsNfsWriteRank MUST be encoded as an Integer syntax value [RFC4517] in the range [0, 255].

<CODE BEGINS>

```
        ///
        /// attributetype (
        ///     1.3.6.1.4.1.31103.1.116 NAME 'fedfsNfsWriteRank'
        ///     DESC 'The write rank of the filesystem'
        ///     EQUALITY integerMatch
        ///     SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
        ///     SINGLE-VALUE
        ///     )
        ///
```

<CODE ENDS>

OID 1.3.6.1.4.1.1466.115.121.1.27 is the Integer syntax [RFC4517].

4.2.1.31. fedfsNfsWriteOrder

A fedfsNfsWriteOrder contains the FSL's FSLI4BX_WRITEORDER [RFC5661] value. A fedfsNfsWriteOrder MUST be encoded as an Integer syntax value [RFC4517] in the range [0, 255].

<CODE BEGINS>

```
        ///
        /// attributetype (
        ///     1.3.6.1.4.1.31103.1.117 NAME 'fedfsNfsWriteOrder'
        ///     DESC 'The write order of the filesystem'
        ///     EQUALITY integerMatch
        ///     SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
        ///     SINGLE-VALUE
        ///     )
        ///
```

<CODE ENDS>

OID 1.3.6.1.4.1.1466.115.121.1.27 is the Integer syntax [RFC4517].

4.2.1.32. fedfsNfsVarSub

A fedfsNfsVarSub stores the value of an FSL's NFSv4.1 FSLI4F_VAR_SUB bit [RFC5661]. A value of "TRUE" indicates the bit is true. A value of "FALSE" indicates the bit is false.

<CODE BEGINS>

```
        ///
        /// attributetype (
        ///     1.3.6.1.4.1.31103.1.118 NAME 'fedfsNfsVarSub'
        ///     DESC 'Indicates if variable substitution is present'
        ///     EQUALITY booleanMatch
        ///     SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
        ///     SINGLE-VALUE
        ///     )
        ///
```

<CODE ENDS>

OID 1.3.6.1.4.1.1466.115.121.1.7 is the Boolean syntax [RFC4517].

4.2.1.33. fedfsNfsValidFor

A fedfsNfsValidFor stores an FSL's NFSv4.1 fs_locations_info fli_valid_for value [RFC5661]. A fedfsNfsValidFor MUST be encoded as an Integer syntax value [RFC4517] in the range [-2147483648, 2147483647].

An FSL's fedfsFslTTL value and fedfsNfsValidFor value MAY be different.

This attribute is single-valued.

<CODE BEGINS>

```
        ///
        /// attributetype (
        ///     1.3.6.1.4.1.31103.1.19 NAME 'fedfsNfsValidFor'
        ///     DESC 'Valid for time'
        ///     EQUALITY integerMatch
        ///     SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
        ///     SINGLE-VALUE
        ///     )
        ///
```

OID 1.3.6.1.4.1.1466.115.121.1.27 is the Integer syntax [RFC4517].

<CODE ENDS>

4.2.2. LDAP Objects

4.2.2.1. fedfsNsdbContainerInfo

A fedfsNsdbContainerInfo describes the location of the NCE.

A fedfsFsn's fedfsNcePrefix attribute is REQUIRED.

A fedfsFsn's fedfsAnnotation and fedfsDescr attributes are OPTIONAL.

<CODE BEGINS>

```
       ///
       /// objectclass (
       ///     1.3.6.1.4.1.31103.1.1001 NAME 'fedfsNsdbContainerInfo'
       ///     DESC 'Describes NCE location'
       ///     SUP top AUXILIARY
       ///     MUST (
       ///             fedfsNcePrefix
       ///     )
       ///     MAY (
       ///             fedfsAnnotation
       ///             $ fedfsDescr
       ///     ))
       ///
```

<CODE ENDS>

4.2.2.2. fedfsFsn

A fedfsFsn represents an FSN.

A fedfsFsn's fedfsNsdbName and fedfsFsnUuid attributes are REQUIRED.

A fedfsFsn's fedfsNsdbPort, fedfsAnnotation, and fedfsDescr attributes are OPTIONAL.

If the fedfsNsdbPort is omitted, the standard LDAP port number, 389, SHOULD be assumed.

The DN of an FSN is REQUIRED to take the following form: "fedfsFsnUuid=$FSNUUID,$NCE", where $FSNUUID is the UUID of the FSN and $NCE is the DN of the NCE ("o=fedfs" by default). Since LDAP requires a DN to be unique, this ensures that each FSN entry has a unique UUID value within the LDAP directory.

A fedfsFsn MAY also have additional attributes, but these attributes MUST NOT be referenced by any part of this document.

<CODE BEGINS>

```
        ///
        /// objectclass (
        ///     1.3.6.1.4.1.31103.1.1002 NAME 'fedfsFsn'
        ///     DESC 'Represents a fileset'
        ///     SUP top STRUCTURAL
        ///     MUST (
        ///             fedfsFsnUuid
        ///             $ fedfsNsdbName
        ///     )
        ///     MAY (
        ///             fedfsNsdbPort
        ///             $ fedfsAnnotation
        ///             $ fedfsDescr
        ///     ))
        ///
```

<CODE ENDS>

4.2.2.3. fedfsFsl

The fedfsFsl object class represents an FSL.

The fedfsFsl is an abstract object class. Protocol specific subtypes of this object class are used to store FSL information. The fedfsNfsFsl object class defined below is used to record an NFS FSL's location. Other subtypes MAY be defined for other protocols (e.g. CIFS).

A fedfsFsl's fedfsFslUuid, fedfsFsnUuid, fedfsNsdbName, fedfsFslHost,

and fedfsFslTTL attributes are REQUIRED.

A fedfsFsl's fedfsNsdbPort, fedfsFslPort, fedfsAnnotation, and fedfsDescr attributes are OPTIONAL.

If the fedfsNsdbPort is omitted, the standard LDAP port number, 389, SHOULD be assumed.

If the fedfsFslPort is omitted, a standard port number based on the type of FSL should be assumed. For an NFS FSL, the standard NFS port number, 2049, SHOULD be assumed.

The DN of an FSL is REQUIRED to take the following form: "fedfsFslUuid=$FSLUUID,fedfsFsnUuid=$FSNUUID,$NCE" where $FSLUUID is the FSL's UUID, $FSNUUID is the FSN's UUID, and $NCE is the DN of the NCE ("o=fedfs" by default). Since LDAP requires a DN to be unique, this ensures that each FSL entry has a unique UUID value within the LDAP directory.

<CODE BEGINS>

```
        ///
        /// objectclass (
        ///     1.3.6.1.4.1.31103.1.1003 NAME 'fedfsFsl'
        ///     DESC 'A physical location of a fileset'
        ///     SUP top ABSTRACT
        ///     MUST (
        ///             fedfsFslUuid
        ///             $ fedfsFsnUuid
        ///             $ fedfsNsdbName
        ///             $ fedfsFslHost
        ///             $ fedfsFslTTL
        ///     )
        ///     MAY (
        ///             fedfsNsdbPort
        ///             $ fedfsFslPort
        ///             $ fedfsAnnotation
        ///             $ fedfsDescr
        ///     ))
        ///
```

<CODE ENDS>

4.2.2.4. fedfsNfsFsl

A fedfsNfsFsl is used to represent an NFS FSL. The fedfsNfsFsl inherits all of the attributes of the fedfsFsl and extends the fedfsFsl with information specific to the NFS protocol.

The DN of an NFS FSL is REQUIRED to take the following form: "fedfsFslUuid=$FSLUUID,fedfsFsnUuid=$FSNUUID,$NCE" where $FSLUUID is the FSL's UUID, $FSNUUID is the FSN's UUID, and $NCE is the DN of the NCE ("o=fedfs" by default). Since LDAP requires a DN to be unique, this ensures that each NFS FSL entry has a unique UUID value within the LDAP directory.

<CODE BEGINS>

```
        ///
        /// objectclass (
        ///     1.3.6.1.4.1.31103.1.1004 NAME 'fedfsNfsFsl'
        ///     DESC 'An NFS location of a fileset'
        ///     SUP fedfsFsl STRUCTURAL
        ///     MUST (
        ///             fedfsNfsPath
        ///             $ fedfsNfsMajorVer
        ///             $ fedfsNfsMinorVer
        ///             $ fedfsNfsCurrency
        ///             $ fedfsNfsGenFlagWritable
        ///             $ fedfsNfsGenFlagGoing
        ///             $ fedfsNfsGenFlagSplit
        ///             $ fedfsNfsTransFlagRdma
        ///             $ fedfsNfsClassSimul
        ///             $ fedfsNfsClassHandle
        ///             $ fedfsNfsClassFileid
        ///             $ fedfsNfsClassWritever
        ///             $ fedfsNfsClassChange
        ///             $ fedfsNfsClassReaddir
        ///             $ fedfsNfsReadRank
        ///             $ fedfsNfsReadOrder
        ///             $ fedfsNfsWriteRank
        ///             $ fedfsNfsWriteOrder
        ///             $ fedfsNfsVarSub
        ///             $ fedfsNfsValidFor
        ///     ))
        ///
```

<CODE ENDS>

5. NSDB Operations

The operations defined by the protocol can be described as several sub-protocols that are used by entities within the federation to perform different roles.

The first of these sub-protocols defines how the state of an NSDB

node can be initialized and updated. The primary use of this sub-protocol is by an administrator to add, edit, or delete filesets, their properties, and their fileset locations.

The second of these sub-protocols defines the queries that are sent to an NSDB node in order to perform resolution (or to find other information about the data stored within that NSDB node) and the responses returned by the NSDB node. The primary use of this sub-protocol is by a fileserver in order to perform resolution, but it may also be used by an administrator to query the state of the system.

The first and second sub-protocols are defined as LDAP operations, using the schema defined in the previous section. If each NSDB node is a standard LDAP server, then, in theory, it is unnecessary to describe the LDAP operations in detail, because the operations are ordinary LDAP operations to query and update records. However, we do not require that an NSDB node implement a complete LDAP service, and therefore we define in these sections the minimum level of LDAP functionality required to implement an NSDB node.

The NSDB sub-protocols are defined in the next two sub-sections. The descriptions of LDAP messages in these sections use the LDAP Data Interchange Format (LDIF) [RFC2849]. In order to differentiate constant and variable strings in the LDIF specifications, variables are prefixed by a $ character and use all upper case characters. For example, a variable named FOO would be specified as $FOO.

This document uses the term NSDB client to refer to an LDAP client that uses either of the NSDB sub-protocols

The third sub-protocol defines the queries and other requests that are sent to a fileserver in order to get information from it or to modify the state of the fileserver in a manner related to the federation protocols. The primary purpose of this protocol is for an administrator to create or delete a junction or discover related information about a particular fileserver.

The third sub-protocol is defined as an ONC RPC protocols. The reason for using ONC RPC instead of LDAP is that all fileservers support ONC RPC but some do not support an LDAP Directory server.

The ONC RPC administration protocol is defined in [FEDFS-ADMIN].

5.1. NSDB Operations for Administrators

The admin entity initiates and controls the commands to manage fileset and namespace information. The admin entity, however, is

stateless. All state is maintained at the NSDB nodes or at the fileserver.

We require that each NSDB node be able to act as an LDAP server and that the protocol used for communicating between the admin entity and each NSDB node is LDAP.

The names we assign to these operations are entirely for the purpose of exposition in this document, and are not part of the LDAP dialogs.

5.1.1. Create an FSN

This operation creates a new FSN in the NSDB by adding a new fedfsFsn entry in the NSDB's LDAP directory.

A fedfsFsn entry contains a fedfsFsnUuid and fedfsNsdbName. The administrator chooses the fedfsFsnUuid and fedfsNsdbName. The process for choosing the fedfsFsnUuid is described in Section 4.2.1.1). The fedfsNsdbName is the name of the NSDB node that will serve as the source of definitive information about the FSN for the life of the FSN.

The NSDB node that receives the request SHOULD check that fedfsNsdbName value matches its own value and return an error if it does not. This is to ensure that an FSN is always created by the NSDB node encoded within the FSN as its owner.

The NSDB node that receives the request SHOULD check all of the attributes for validity and consistency, but this is not generally possible for LDAP servers because the consistency requirements cannot be expressed in the LDAP schema (although many LDAP servers can be extended, via plug-ins or other mechanisms, to add functionality beyond the strict definition of LDAP).

5.1.1.1. LDAP Request

This operation is implemented using the LDAP ADD request described by the LDIF below.

```
dn: fedfsFsnUuid=$FSNUUID,$NCE
changeType: add
objectClass: fedfsFsn
fedfsFsnUuid: $FSNUUID
fedfsNsdbName: $NSDBNAME
```

For example, if the $FSNUUID is "f81d4fae-7dec-11d0-a765-00a0c91e6bf6", the $NSDBNAME is "nsdb.example.com", and the $NCE is "o=fedfs" the operation would be:

```
dn: fedfsFsnUuid=f81d4fae-7dec-11d0-a765-00a0c91e6bf6,o=fedfs
changeType: add
objectClass: fedfsFsn
fedfsFsnUuid: f81d4fae-7dec-11d0-a765-00a0c91e6bf6
fedfsNsdbName: nsdb.example.com
```

5.1.2. Delete an FSN

This operation deletes an FSN by removing a fedfsFsn entry in the NSDB's LDAP directory.

If the FSN entry being deleted has child FSL entries, this function MUST return an error. This ensures that the NSDB will not contain any orphaned FSL entries. A compliant LDAP implementation will meet this requirement since Section 4.8 of [RFC4511] defines the LDAP delete operation to only be capable of removing leaf entries.

Note that the FSN delete function only removes the fileset from the namespace (by removing the records for that FSN from the NSDB node that receives this request). The fileset and its data are not deleted. Any junction that has this FSN as its target may continue to point to this non-existent FSN. A dangling reference may be detected when a client tries to resolve the target of a junction that refers to the deleted FSN and the NSDB returns an error.

5.1.2.1. LDAP Request

This operation is implemented using the LDAP DELETE request described by the LDIF below.

```
dn: fedfsFsnUuid=$FSNUUID,$NCE
changeType: delete
```

For example, if the $FSNUUID is "f81d4fae-7dec-11d0-a765-00a0c91e6bf6" and $NCE is "o=fedfs", the operation would be:

```
dn: fedfsFsnUuid=f81d4fae-7dec-11d0-a765-00a0c91e6bf6,o=fedfs
changeType: delete
```

5.1.3. Create an FSL

This operation creates a new FSL for the given FSN by adding a new fedfsFsl entry in the NSDB's LDAP directory.

A fedfsFsl entry contains a fedfsFslUuid, fedfsFsnUuid, fedfsNsdbName, fedfsFslHost, and fedfsFslTTL. The administrator chooses the fedfsFslUuid. The process for choosing the fedfsFslUuid is described in Section 4.2.1.1. The fedfsFsnUuid is the UUID of the

FSL's FSN. The fedfsNsdbName is the name of the NSDB node that stores definitive information about the FSL's FSN. The fedfsFslHost value is the network location of the fileserver that stores the FSL. The fedfsFslTTL is chosen by the administrator as described in Section 2.8.2.

The administrator will also set additional attributes depending on the FSL type.

5.1.3.1. LDAP Request

This operation is implemented using the LDAP ADD request described by the LDIF below (NOTE: the LDIF shows the creation of an NFS FSL)

```
dn:fedfsFslUuid=$FSLUUID,fedfsFsnUuid=$FSNUUID,$NCE
changeType: add
objectClass: fedfsNfsFsl
fedfsFslUuid: $FSLUUID
fedfsFsnUuid: $FSNUUID
fedfsNsdbName: $NSDBNAME
fedfsFslHost: $HOST
fedfsFslPort: $PORT
fedfsFslTTL: $TTL
fedfsNfsPath: $PATH
fedfsNfsMajorVer: $MAJOR
fedfsNfsMinorVer: $MINOR
fedfsNfsCurrency: $CURRENCY
fedfsNfsGenFlagWritable: $WRITABLE
fedfsNfsGenFlagGoing: $GOING
fedfsNfsGenFlagSplit: $SPLIT
fedfsNfsTransFlagRdma: $RDMA
fedfsNfsClassSimul: $CLASS_SIMUL
fedfsNfsClassHandle:$CLASS_HANDLE
fedfsNfsClassFileid:$CLASS_FILEID
fedfsNfsClassWritever:$CLASS_WRITEVER
fedfsNfsClassChange: $CLASS_CHANGE
fedfsNfsClassReaddir: $CLASS_READDIR
fedfsNfsReadRank: $READ_RANK
fedfsNfsReadOrder: $READ_ORDER
fedfsNfsWriteRank: $WRITE_RANK
fedfsNfsWriteOrder: $WRITE_ORDER
fedfsNfsVarSub: $VAR_SUB
fedfsNfsValidFor: $TIME
fedfsAnnotation: $ANNOTATION
fedfsDescr: $DESCR
```

For example, if the $FSNUUID is "f81d4fae-7dec-11d0-a765-00a0c91e6bf6", the $FSLUUID is "84f775a7-8e31-14ae-b39d-10eeee060d2c", the $NSDBNAME is "nsdb.example.com", the $HOST is "server.example.com", $PORT is "2049", the $TTL is "300" seconds, the $PATH is stored in the file "/tmp/fsl_path", fileset is exported via NFSv4.1 ($MAJOR is "4" and $MINOR is "1"), $CURRENCY is "0" (an up to date copy), the FSL is writable, but not going, split, or accessible via RDMA, the simultaneous-use class is "1", the handle class is "0", the fileid class is "1", the write-verifier class is "1", the change class is "1", the readdir class is "9", the read rank is "7", the read order is "8", the write rank is "5", the write order is "6", variable substitution is false, $TIME is "300" seconds, $ANNOTATION is ""foo" = "bar"", $DESC is "This is a description.", and the $NCE is "o=fedfs", the operation would be (for readability the DN is split into two lines):

```
dn:fedfsFslUuid=84f775a7-8e31-14ae-b39d-10eeee060d2c,
    fedfsFsnUuid=f81d4fae-7dec-11d0-a765-00a0c91e6bf6,o=fedfs
changeType: add
objectClass: fedfsNfsFsl
fedfsFslUuid: 84f775a7-8e31-14ae-b39d-10eeee060d2c
fedfsFsnUuid: f81d4fae-7dec-11d0-a765-00a0c91e6bf6
fedfsNsdbName: nsdb.example.com
fedfsFslHost: server.example.com
fedfsFslPort: 2049
fedfsFslTTL: 300
fedfsNfsPath:< file:///tmp/fsl_path
fedfsNfsMajorVer: 4
fedfsNfsMinorVer: 1
fedfsNfsCurrency: 0
fedfsNfsGenFlagWritable: TRUE
fedfsNfsGenFlagGoing: FALSE
fedfsNfsGenFlagSplit: FALSE
fedfsNfsTransFlagRdma: FALSE
fedfsNfsClassSimul: 1
fedfsNfsClassHandle: 0
fedfsNfsClassFileid: 1
fedfsNfsClassWritever: 1
fedfsNfsClassChange: 1
fedfsNfsClassReaddir: 9
fedfsNfsReadRank: 7
fedfsNfsReadOrder: 8
fedfsNfsWriteRank: 5
fedfsNfsWriteOrder: 6
fedfsNfsVarSub: FALSE
fedfsNfsValidFor: 300
fedfsAnnotation: "foo" = "bar"
```

```
        fedfsDescr: This is a description.
```

5.1.3.2. Selecting fedfsNfsFsl Values

The fedfsNfsFSl object class is used to describe NFSv4 and NFSv4.1 accessible filesets. For the reasons described in Section 2.8.3, administrators SHOULD choose reasonable values for all LDAP attributes of an NFSv4 accessible fedfsNfsFsl even though some of these LDAP attributes are not explicitly contained in the NFSv4 fs_locations attribute returned to an NFSv4 client.

When the administrator is unable to choose reasonable values for the LDAP attributes not explicitly contained in a NFSv4 fs_locations attribute, the values in the following table are RECOMMENDED.

LDAP attribute	LDAP value	Notes
fedfsNfsCurrency	negative value	Indicates that the server does not know the currency (see 11.10.1 of [RFC5661]).
fedfsNfsGenFlagWritable	FALSE	Leaving unset is not harmful (see 11.10.1 of [RFC5661]).
fedfsNfsGenFlagGoing	FALSE	NFS client will detect a migration event if the FSL becomes unavailable.
fedfsNfsGenFlagSplit	TRUE	Safe to assume that the FSL is split.
fedfsNfsTransFlagRdma	TRUE	NFS client will detect if RDMA access is available.
fedfsNfsClassSimul	0	0 (zero) is treated as non-matching (see 11.10.1 of [RFC5661]).
fedfsNfsClassHandle	0	See fedfsNfsClassSimul note.
fedfsNfsClassFileid	0	See fedfsNfsClassSimul note.
fedfsNfsClassWritever	0	See fedfsNfsClassSimul note.
fedfsNfsClassChange	0	See fedfsNfsClassSimul note.
fedfsNfsClassReaddir	0	See fedfsNfsClassSimul note.
fedfsNfsReadRank	0	Highest value ensures FSL will be tried.
fedfsNfsReadOrder	0	See fedfsNfsReadRank note.
fedfsNfsWriteRank	0	See fedfsNfsReadRank note.
fedfsNfsWriteOrder	0	See fedfsNfsReadRank note.
fedfsNfsVarSub	FALSE	NFSv4 does not define variable substituion in paths.

fedfsNfsValidFor	0	Indicates no appropriate refetch interval (see 11.10.2 of [RFC5661]).

5.1.4. Delete an FSL

This operation deletes the given Fileset location. The admin requests the NSDB node storing the fedfsFsl to delete it from its database. This operation does not result in the fileset location's data being deleted at the fileserver.

5.1.4.1. LDAP Request

The admin sends an LDAP DELETE request to the NSDB node to remove the FSL.

```
dn: fedfsFslUuid=$FSLUUID,fedfsFsnUuid=$FSNUUID,$NCE
changeType: delete
```

For example, if the $FSNUUID is "f81d4fae-7dec-11d0-a765-00a0c91e6bf6", the $FSLUUID is "84f775a7-8e31-14ae-b39d-10eeee060d2c", and the $NCE is "o=fedfs", the operation would be (for readability the DN is split into two lines):

```
dn: fedfsFslUuid=84f775a7-8e31-14ae-b39d-10eeee060d2c,
     fedfsFsnUuid=f81d4fae-7dec-11d0-a765-00a0c91e6bf6,o=fedfs
changeType: delete
```

5.1.5. Update an FSL

This operation updates the attributes of a given FSL. This command results in a change in the attributes of the fedfsFsl at the NSDB node maintaining this FSL. The attributes that must not change are the fedfsFslUuid and the fedfsFsnUuid of the fileset this FSL implements.

5.1.5.1. LDAP Request

The admin sends an LDAP MODIFY request to the NSDB node to update the FSL.

```
dn: fedfsFslUuid=$FSLUUID,fedfsFsnUuid=$FSNUUID,$NCE
```

```
changeType: modify
replace: $ATTRIBUTE-TYPE
```

For example, if the $FSNUUID is "f81d4fae-7dec-11d0-a765-00a0c91e6bf6", the $FSLUUID is "84f775a7-8e31-14ae-b39d-10eeee060d2c", the $NCE is "o=fedfs", and the administrator wished to change the TTL to 10 minutes, the operation would be (for readability the DN is split into two lines):

```
dn: fedfsFslUuid=84f775a7-8e31-14ae-b39d-10eeee060d2c,
     fedfsFsnUuid=f81d4fae-7dec-11d0-a765-00a0c91e6bf6,o=fedfs
changeType: modify
replace: fedfsFslTTL
fedfsFslTTL: 600
```

5.2. NSDB Operations for Fileservers

5.2.1. NSDB Container Entry (NCE) Enumeration

To find the NCEs for the NSDB foo.example.com, a fileserver would do the following:

```
nce_list = empty
connect to the LDAP directory at foo.example.com
for each namingContext value $BAR in the root DSE
    /* $BAR is a DN */
    query for a fedfsNcePrefix value at $BAR
    /*
     * The LDAP URL for this search would be
     *
     *  ldap://foo.example.com:389/$BAR?fedfsNcePrefix??
     *                      (objectClass=fedfsNsdbContainerInfo)
     *
     */
    if a fedfsNcePrefix value is found
        nce = prepend the fedfsNcePrefix value to $BAR
        add nce to the nce_list
```

5.2.2. Lookup FSLs for an FSN

Using an LDAP search, the fileserver can obtain all of the FSLs for a given FSN. The FSN's fedfsFsnUuid is used as the search key. The following examples use the LDAP Universal Resource Identifier (URI) format defined in [RFC4516].

To obtain a list of all FSLs for $FSNUUID on the NSDB named $NSDBNAME, the following search can be used (for readability the URI is split into two lines):

```
for each $NCE in nce_list
    ldap://$NSDBNAME/fsnUuid=$FSNUUID,$NCE??one?
        (objectClass=fedfsFsl)
```

This search is for the children of the object with DN "fedfsFsnUuid=$FSNUUID,$NCE" with a filter for "objectClass=fedfsFsl". The scope value of "one" restricts the search to the entry's children (rather than the entire subtree below the entry) and the filter ensures that only FSL entries are returned.

For example if $NSDBNAME is "nsdb.example.com", $FSNUUID is "f81d4fae-7dec-11d0-a765-00a0c91e6bf6", and $NCE is "o=fedfs", the search would be (for readability the URI is split into three lines):

```
ldap://nsdb.example.com/
        fsnUuid=f81d4fae-7dec-11d0-a765-00a0c91e6bf6,o=fedfs
        ??one?(objectClass=fedfsFsl)
```

The following search can be used to obtain only the NFS FSLs for $FSNUUID on the NSDB named $NSDBNAME (for readability the URI is split into two lines):

```
for each $NCE in nce_list
    ldap://$NSDBNAME/fsnUuid=$FSNUUID,$NCE??one?
        (objectClass=fedfsNfsFsl)
```

This also searches for the children of the object with DN "fedfsFsnUuid=$FSNUUID,$NCE", but the filter for "objectClass = fedfsNfsFsl" restricts the results to only NFS FSLs.

For example if $NSDBNAME is nsdb.example.com, $FSNUUID is f81d4fae-7dec-11d0-a765-00a0c91e6bf6, and $NCE is "o=fedfs",the search would be (for readability the URI is split into three lines):

```
ldap://nsdb.example.com/
        fsnUuid=f81d4fae-7dec-11d0-a765-00a0c91e6bf6,o=fedfs
        ??one?(objectClass=fedfsNfsFsl)
```

The fileserver will generate a referral based on the set of FSLs returned by these queries using the process described in Section 2.8.3.

5.3. NSDB Operations and LDAP Referrals

The LDAPv3 protocol defines an LDAP referral mechanism that allows an LDAP server to redirect an LDAP client. LDAPv3 defines two types of LDAP referrals: the Referral type defined in Section 4.1.10 of [RFC4511] and the SearchResultReference type defined in Section 4.5.3 of [RFC4511]. In both cases, the LDAP referral lists one or more URIs for services that can be used to complete the operation. In the remainder of this document, the term LDAP referral is used to indicate either of these types.

If an NSDB operation results in an LDAP referral, the NSDB client MAY follow the LDAP referral. An NSDB client's decision to follow an LDAP referral is implementation and configuration dependent. For example, an NSDB client might be configured to follow only those LDAP referrals that were received over a secure channel, or only those that target an NSDB that supports encrypted communication. If an NSDB client chooses to follow an LDAP referral, the NSDB client MUST process the LDAP referral and prevent looping as described in Section 4.1.10 of [RFC4511].

6. Security Considerations

Both NFSv4/NFSv4.1 and LDAP provide security mechanisms. When used in conjunction with the federated filesystem protocols described in this document, the use of these mechanisms is RECOMMENDED. Specifically, the use of RPCSEC_GSS [RFC2203], which is built on the GSS-API [RFC2743], is RECOMMENDED on all NFS connections between a client and fileserver. The "Security Considerations" sections of the the NFSv4 [3530bis] and NFSv4.1 [RFC5661] specifications contain special considerations for the handling of GETATTR operations for the fs_locations and fs_locations_info attributes. For all LDAP connections established by the federated filesystem protocols, the use of TLS [RFC5246], as described in [RFC4513], is RECOMMENDED.

If an NSDB client chooses to follow an LDAP referral, the NSDB client SHOULD authenticate the LDAP referral's target NSDB using the target NSDB's credentials (not the credentials of the NSDB that generated the LDAP referral). The NSDB client SHOULD NOT follow an LDAP referral that targets an NSDB for which it does not know the NSDB's credentials.

Within a federation, there are two types of components an attacker

may compromise: a fileserver and an NSDB.

If an attacker compromises a fileserver, the attacker can interfere with the client's filesystem I/O operations (e.g. by returning fictitious data in the response to a read request) or fabricating a referral. The attacker's abilities are the same regardless of whether or not the federation protocols are in use. While the federation protocols do not give the attacker additional capabilities, they are additional targets for attack. The LDAP protocol described in Section 5.2 SHOULD be secured using the methods described above to defeat attacks on a fileserver via this channel.

If an attacker compromises an NSDB, the attacker will be able to forge FSL information and thus poison the fileserver's referral information. Therefore an NSDB should be as secure as the fileservers which query it. The LDAP operations described in Section 5 SHOULD be secured using the methods described above to defeat attacks on an NSDB via this channel.

It should be noted that the federation protocols do not directly provide access to filesystem data. The federation protocols only provide a mechanism for building a namespace. All data transfers occur between a client and server just as they would if the federation protocols were not in use. As a result, the federation protocols do not require new user authentication and authorization mechanisms or require a fileserver to act as a proxy for a client.

7. IANA Considerations

7.1. Registry for the fedfsAnnotation Key Namespace

This document defines the fedfsAnnotation key in Section 4.2.1.12. The fedfsAnnotation key namespace is to be managed by IANA. IANA is to create and maintain a new registry entitled "FedFS Annotation Keys". Future registrations are to be administered by IANA using the "First Come First Served" policy defined in [RFC5226]. Registration requests MUST include the key (a valid UTF-8 string of any length), a brief description of the key's purpose, and an email contact for the registration. For viewing, the registry should be sorted lexicographically by key. There are no initial assignments for this registry.

7.2. Registry for FedFS Object Identifiers

Using the process described in [RFC2578], one of the authors was assigned the Internet Private Enterprise Numbers range 1.3.6.1.4.1.31103.x. Within this range, the subrange

1.3.6.1.4.1.31103.1.x is permanently dedicated for use by the federated file system protocols.

IANA is to create and maintain a new registry entitled "FedFS Object Identifiers" for the purpose of administering the FedFS Object Identifier (OID) range. Future allocations from the 1.3.6.1.4.1.31103.1.x range are to be assigned by IANA using the "RFC Required" policy defined in [RFC5226]. Registration requests MUST include an OID value from the 1.3.6.1.4.1.31103.1.x range, a short description of the OID, and a reference to the specification that defines the OID's usage. For viewing, the registry should be sorted numerically by OID value. The initial contents of the FedFS Object Identifiers registry are given in Table 1.

OID	Description	Reference
1.3.6.1.4.1.31103.1.1	fedfsUuid	RFC-TBD1
1.3.6.1.4.1.31103.1.2	fedfsNetAddr	RFC-TBD1
1.3.6.1.4.1.31103.1.3	fedfsNetPort	RFC-TBD1
1.3.6.1.4.1.31103.1.4	fedfsFsnUuid	RFC-TBD1
1.3.6.1.4.1.31103.1.5	fedfsNsdbName	RFC-TBD1
1.3.6.1.4.1.31103.1.6	fedfsNsdbPort	RFC-TBD1
1.3.6.1.4.1.31103.1.7	fedfsNcePrefix	RFC-TBD1
1.3.6.1.4.1.31103.1.8	fedfsFslUuid	RFC-TBD1
1.3.6.1.4.1.31103.1.9	fedfsFslHost	RFC-TBD1
1.3.6.1.4.1.31103.1.10	fedfsFslPort	RFC-TBD1
1.3.6.1.4.1.31103.1.11	fedfsFslTTL	RFC-TBD1
1.3.6.1.4.1.31103.1.12	fedfsAnnotation	RFC-TBD1
1.3.6.1.4.1.31103.1.13	fedfsDescr	RFC-TBD1
1.3.6.1.4.1.31103.1.100	fedfsNfsPath	RFC-TBD1
1.3.6.1.4.1.31103.1.101	fedfsNfsMajorVer	RFC-TBD1
1.3.6.1.4.1.31103.1.102	fedfsNfsMinorVer	RFC-TBD1
1.3.6.1.4.1.31103.1.103	fedfsNfsCurrency	RFC-TBD1
1.3.6.1.4.1.31103.1.104	fedfsNfsGenFlagWritable	RFC-TBD1
1.3.6.1.4.1.31103.1.105	fedfsNfsGenFlagGoing	RFC-TBD1
1.3.6.1.4.1.31103.1.106	fedfsNfsGenFlagSplit	RFC-TBD1
1.3.6.1.4.1.31103.1.107	fedfsNfsTransFlagRdma	RFC-TBD1
1.3.6.1.4.1.31103.1.108	fedfsNfsClassSimul	RFC-TBD1
1.3.6.1.4.1.31103.1.109	fedfsNfsClassHandle	RFC-TBD1
1.3.6.1.4.1.31103.1.110	fedfsNfsClassFileid	RFC-TBD1
1.3.6.1.4.1.31103.1.111	fedfsNfsClassWritever	RFC-TBD1
1.3.6.1.4.1.31103.1.112	fedfsNfsClassChange	RFC-TBD1
1.3.6.1.4.1.31103.1.113	fedfsNfsClassReaddir	RFC-TBD1
1.3.6.1.4.1.31103.1.114	fedfsNfsReadRank	RFC-TBD1
1.3.6.1.4.1.31103.1.115	fedfsNfsReadOrder	RFC-TBD1
1.3.6.1.4.1.31103.1.116	fedfsNfsWriteRank	RFC-TBD1
1.3.6.1.4.1.31103.1.117	fedfsNfsWriteOrder	RFC-TBD1
1.3.6.1.4.1.31103.1.118	fedfsNfsVarSub	RFC-TBD1
1.3.6.1.4.1.31103.1.119	fedfsNfsValidFor	RFC-TBD1
1.3.6.1.4.1.31103.1.1001	fedfsNsdbContainerInfo	RFC-TBD1
1.3.6.1.4.1.31103.1.1002	fedfsFsn	RFC-TBD1
1.3.6.1.4.1.31103.1.1003	fedfsFsl	RFC-TBD1
1.3.6.1.4.1.31103.1.1004	fedfsNfsFsl	RFC-TBD1

Table 1

7.3. LDAP Descriptor Registration

In accordance with Section 3.4 and Section 4 of [RFC4520], the object identifier descriptors defined in this document (listed below) will be registered via the Expert Review process.

Subject: Request for LDAP Descriptor Registration
Person & email address to contact for further information: See "Author/Change Controller"
Specification: draft-ietf-nfsv4-federated-fs-protocol
Author/Change Controller: [document authors]

Object Identifier: 1.3.6.1.4.1.31103.1.1
Descriptor (short name): fedfsUuid
Usage: attribute type

Object Identifier: 1.3.6.1.4.1.31103.1.2
Descriptor (short name): fedfsNetAddr
Usage: attribute type

Object Identifier: 1.3.6.1.4.1.31103.1.3
Descriptor (short name): fedfsNetPort
Usage: attribute type

Object Identifier: 1.3.6.1.4.1.31103.1.4
Descriptor (short name): fedfsFsnUuid
Usage: attribute type

Object Identifier: 1.3.6.1.4.1.31103.1.5
Descriptor (short name): fedfsNsdbName
Usage: attribute type

Object Identifier: 1.3.6.1.4.1.31103.1.6
Descriptor (short name): fedfsNsdbPort
Usage: attribute type

Object Identifier: 1.3.6.1.4.1.31103.1.7
Descriptor (short name): fedfsNcePrefix
Usage: attribute type

Object Identifier: 1.3.6.1.4.1.31103.1.8
Descriptor (short name): fedfsFslUuid
Usage: attribute type

Object Identifier: 1.3.6.1.4.1.31103.1.9
Descriptor (short name): fedfsFslHost
Usage: attribute type

Object Identifier: 1.3.6.1.4.1.31103.1.10
Descriptor (short name): fedfsFslPort
Usage: attribute type

Object Identifier: 1.3.6.1.4.1.31103.1.11
Descriptor (short name): fedfsFslTTL
Usage: attribute type

Object Identifier: 1.3.6.1.4.1.31103.1.12
Descriptor (short name): fedfsAnnotation
Usage: attribute type

Object Identifier: 1.3.6.1.4.1.31103.1.13
Descriptor (short name): fedfsDescr
Usage: attribute type

Object Identifier: 1.3.6.1.4.1.31103.1.100
Descriptor (short name): fedfsNfsPath
Usage: attribute type

Object Identifier: 1.3.6.1.4.1.31103.1.101
Descriptor (short name): fedfsNfsMajorVer
Usage: attribute type

Object Identifier: 1.3.6.1.4.1.31103.1.102
Descriptor (short name): fedfsNfsMinorVer
Usage: attribute type

Object Identifier: 1.3.6.1.4.1.31103.1.103
Descriptor (short name): fedfsNfsCurrency
Usage: attribute type

Object Identifier: 1.3.6.1.4.1.31103.1.104
Descriptor (short name): fedfsNfsGenFlagWritable
Usage: attribute type

Object Identifier: 1.3.6.1.4.1.31103.1.105
Descriptor (short name): fedfsNfsGenFlagGoing
Usage: attribute type

Object Identifier: 1.3.6.1.4.1.31103.1.106
Descriptor (short name): fedfsNfsGenFlagSplit
Usage: attribute type

Object Identifier: 1.3.6.1.4.1.31103.1.107
Descriptor (short name): fedfsNfsTransFlagRdma
Usage: attribute type

Object Identifier: 1.3.6.1.4.1.31103.1.108
Descriptor (short name): fedfsNfsClassSimul
Usage: attribute type

Object Identifier: 1.3.6.1.4.1.31103.1.109
Descriptor (short name): fedfsNfsClassHandle
Usage: attribute type

Object Identifier: 1.3.6.1.4.1.31103.1.110
Descriptor (short name): fedfsNfsClassFileid
Usage: attribute type

Object Identifier: 1.3.6.1.4.1.31103.1.111
Descriptor (short name): fedfsNfsClassWritever
Usage: attribute type

Object Identifier: 1.3.6.1.4.1.31103.1.112
Descriptor (short name): fedfsNfsClassChange
Usage: attribute type

Object Identifier: 1.3.6.1.4.1.31103.1.113
Descriptor (short name): fedfsNfsClassReaddir
Usage: attribute type

Object Identifier: 1.3.6.1.4.1.31103.1.114
Descriptor (short name): fedfsNfsReadRank
Usage: attribute type

Object Identifier: 1.3.6.1.4.1.31103.1.115
Descriptor (short name): fedfsNfsReadOrder
Usage: attribute type

Object Identifier: 1.3.6.1.4.1.31103.1.116
Descriptor (short name): fedfsNfsWriteRank
Usage: attribute type

Object Identifier: 1.3.6.1.4.1.31103.1.117
Descriptor (short name): fedfsNfsWriteOrder
Usage: attribute type

Object Identifier: 1.3.6.1.4.1.31103.1.118
Descriptor (short name): fedfsNfsVarSub
Usage: attribute type

Object Identifier: 1.3.6.1.4.1.31103.1.119
Descriptor (short name): fedfsNfsValidFor
Usage: attribute type

Object Identifier: 1.3.6.1.4.1.31103.1.1001
Descriptor (short name): fedfsNsdbContainerInfo
Usage: object class

Object Identifier: 1.3.6.1.4.1.31103.1.1002
Descriptor (short name): fedfsFsn
Usage: object class

Object Identifier: 1.3.6.1.4.1.31103.1.1003
Descriptor (short name): fedfsFsl
Usage: object class

Object Identifier: 1.3.6.1.4.1.31103.1.1004
Descriptor (short name): fedfsNfsFsl
Usage: object class

8. Glossary

Administrator: user with the necessary authority to initiate administrative tasks on one or more servers.

Admin Entity: A server or agent that administers a collection of fileservers and persistently stores the namespace information.

Client: Any client that accesses the fileserver data using a supported filesystem access protocol.

Federation: A set of server collections and singleton servers that use a common set of interfaces and protocols in order to provide to their clients a federated namespace accessible through a filesystem access protocol.

Fileserver: A server exporting a filesystem via a network filesystem access protocol.

Fileset: The abstraction of a set of files and the directory tree that contains them. A fileset is the fundamental unit of data management in the federation.

Note that all files within a fileset are descendants of one directory, and that filesets do not span filesystems.

Filesystem: A self-contained unit of export for a fileserver, and the mechanism used to implement filesets. The fileset does not need to be rooted at the root of the filesystem, nor at the export point for the filesystem.

A single filesystem MAY implement more than one fileset, if the client protocol and the fileserver permit this.

Filesystem Access Protocol: A network filesystem access protocol such as NFSv2 [RFC1094], NFSv3 [RFC1813], NFSv4 [3530bis], or CIFS (Common Internet File System) [MS-SMB] [MS-SMB2] [MS-CIFS].

FSL (Fileset Location): The location of the implementation of a fileset at a particular moment in time. An FSL MUST be something that can be translated into a protocol-specific description of a resource that a client can access directly, such as an fs_location (for NFSv4), or share name (for CIFS). Note that not all FSLs need to be explicitly exported as long as they are contained within an exported path on the fileserver.

FSN (Fileset Name): A platform-independent and globally unique name for a fileset. Two FSLs that implement replicas of the same fileset MUST have the same FSN, and if a fileset is migrated from one location to another, the FSN of that fileset MUST remain the same.

Junction: A filesystem object used to link a directory name in the current fileset with an object within another fileset. The server-side "link" from a leaf node in one fileset to the root of another fileset.

Namespace: A filename/directory tree that a sufficiently authorized client can observe.

NSDB (Namespace Database) Service: A service that maps FSNs to FSLs. The NSDB may also be used to store other information, such as annotations for these mappings and their components.

NSDB Node: The name or location of a server that implements part of the NSDB service and is responsible for keeping track of the FSLs (and related info) that implement a given partition of the FSNs.

Referral: A server response to a client access that directs the client to evaluate the current object as a reference to an object at a different location (specified by an FSL) in another fileset, and possibly hosted on another fileserver. The client re-attempts the access to the object at the new location.

Replica: A replica is a redundant implementation of a fileset. Each replica shares the same FSN, but has a different FSL.

Replicas may be used to increase availability or performance. Updates to replicas of the same fileset MUST appear to occur in the same order, and therefore each replica is self-consistent at any moment.

We do not assume that updates to each replica occur simultaneously. If a replica is offline or unreachable, the other replicas may be updated.

Server Collection: A set of fileservers administered as a unit. A server collection may be administered with vendor-specific software.

The namespace provided by a server collection could be part of the federated namespace.

Singleton Server: A server collection containing only one server; a stand-alone fileserver.

9. References

9.1. Normative References

[3530bis] Haynes, T. and D. Noveck, "NFS Version 4 Protocol", draft-ietf-nfsv4-rfc3530bis (Work In Progress), 2010.

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

[RFC2203] Eisler, M., Chiu, A., and L. Ling, "RPCSEC_GSS Protocol Specification", RFC 2203, September 1997.

[RFC2578] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Structure of Management Information

Version 2 (SMIv2)", STD 58, RFC 2578, April 1999.

[RFC2743] Linn, J., "Generic Security Service Application Program Interface Version 2, Update 1", RFC 2743, January 2000.

[RFC2849] Good, G., "The LDAP Data Interchange Format (LDIF) - Technical Specification", RFC 2849, June 2000.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, July 2005.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.

[RFC4510] Zeilenga, K., "Lightweight Directory Access Protocol (LDAP): Technical Specification Road Map", RFC 4510, June 2006.

[RFC4511] Sermersheim, J., "Lightweight Directory Access Protocol (LDAP): The Protocol", RFC 4511, June 2006.

[RFC4512] Zeilenga, K., "Lightweight Directory Access Protocol (LDAP): Directory Information Models", RFC 4512, June 2006.

[RFC4513] Harrison, R., "Lightweight Directory Access Protocol (LDAP): Authentication Methods and Security Mechanisms", RFC 4513, June 2006.

[RFC4516] Smith, M. and T. Howes, "Lightweight Directory Access Protocol (LDAP): Uniform Resource Locator", RFC 4516, June 2006.

[RFC4517] Legg, S., "Lightweight Directory Access Protocol (LDAP): Syntaxes and Matching Rules", RFC 4517, June 2006.

[RFC4519] Sciberras, A., "Lightweight Directory Access Protocol (LDAP): Schema for User Applications", RFC 4519, June 2006.

[RFC4520] Zeilenga, K., "Internet Assigned Numbers Authority (IANA) Considerations for the Lightweight Directory Access Protocol (LDAP)", BCP 64, RFC 4520, June 2006.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC5661] Shepler, S., Eisler, M., and D. Noveck, "Network File System (NFS) Version 4 Minor Version 1 Protocol", RFC 5661, January 2010.

9.2. Informative References

[AFS] Howard, J., "An Overview of the Andrew File System", Proceeding of the USENIX Winter Technical Conference , 1988.

[FEDFS-ADMIN] Lentini, J., Everhart, C., Ellard, D., Tewari, R., and M. Naik, "Administration Protocol for Federated Filesystems", draft-ietf-nfsv4-federated-fs-admin (Work In Progress), 2010.

[FEDFS-DNS-SRV] Everhart, C., Adamson, W., and J. Zhang, "Using DNS SRV to Specify a Global File Name Space with NFS version 4", draft-ietf-nfsv4-federated-fs-dns-srv-namespace (Work In Progress), 2010.

[MS-CIFS] Microsoft Corporation, "Common Internet File System (CIFS) Protocol Specification", MS-CIFS 2.0, November 2009.

[MS-SMB] Microsoft Corporation, "Server Message Block (SMB) Protocol Specification", MS-SMB 17.0, November 2009.

[MS-SMB2] Microsoft Corporation, "Server Message Block (SMB) Version 2 Protocol Specification", MS-SMB2 19.0, November 2009.

[RFC1094] Nowicki, B., "NFS: Network File System Protocol specification", RFC 1094, March 1989.

[RFC1813] Callaghan, B., Pawlowski, B., and P. Staubach, "NFS Version 3 Protocol Specification", RFC 1813, June 1995.

[RFC3254] Alvestrand, H., "Definitions for talking about

directories", RFC 3254, April 2002.

[RFC5662] Shepler, S., Eisler, M., and D. Noveck, "Network File System (NFS) Version 4 Minor Version 1 External Data Representation Standard (XDR) Description", RFC 5662, January 2010.

[RFC5716] Lentini, J., Everhart, C., Ellard, D., Tewari, R., and M. Naik, "Requirements for Federated File Systems", RFC 5716, January 2010.

Appendix A. Acknowledgments

We would like to thank Andy Adamson of NetApp, Paul Lemahieu of EMC, Robert Thurlow of Sun Microsystems, and Mario Wurzl of EMC for helping to author this document.

We would also like to thank George Amvrosiadis, Chuck Lever, Trond Myklebust, and Nicolas Williams for their comments.

The extract.sh shell script and formatting conventions were first described by the authors of the NFSv4.1 XDR specification [RFC5662].

Authors' Addresses

James Lentini
NetApp
1601 Trapelo Rd, Suite 16
Waltham, MA 02451
US

Phone: +1 781-768-5359
Email: jlentini@netapp.com

Craig Everhart
NetApp
800 Cranberry Woods Drive
Cranberry Township, PA 16066
US

Phone: +1 724-741-5101
Email: Craig.Everhart@netapp.com

Daniel Ellard
Raytheon BBN Technologies
10 Moulton Street
Cambridge, MA 02138
US

Phone: +1 617-873-8004
Email: dellard@bbn.com

Renu Tewari
IBM Almaden
650 Harry Rd
San Jose, CA 95120
US

Email: tewarir@us.ibm.com

Manoj Naik
IBM Almaden
650 Harry Rd
San Jose, CA 95120
US

Email: manoj@almaden.ibm.com