Network File System Version 4 C. Lever
Internet-Draft Oracle
Intended status: Standards Track S. Sorce
Expires: November 3, 2016 Red Hat
May 2, 2016

A Simpler Mechanism For Organizing FedFS NSDBs
draft-cel-nfsv4-federated-fs-nce-05

Abstract

This document describes a new, simpler mechanism for searching FedFS NSDBs (Name Space Data Bases) for FedFS records. This mechanism replaces the mechanism described in existing FedFS Proposed Standards.

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

Copyright Notice

Table of Contents

1. Introduction

A FedFS junction is an object stored on a fileserver that redirects file-access clients to other fileserver shares. Using junctions, multiple fileserver shares on many fileservers can be joined into a single filename namespace. Think of a junction as a symlink that points to the root directory of a share stored on another fileserver.

The target locations of these remote shares are not stored in FedFS junctions. Instead, a junction contains a reference to a set of location information stored in an LDAP directory. The LDAP directory is referred to as a Name Space Data Base, or NSDB.

The fundamental and most frequent operation performed with an NSDB is "FSN resolution," described in section 5.2.2 of [RFC7532]. This is the act of reading the reference stored in a junction, retrieving the referenced location information from an NSDB, and forming a filesystem referral to send to file-access clients using a native filesystem protocol. In this scenario, a fileserver acts as an NSDB client.

FedFS records in any NSDB are stored as descendants of an NSDB Container Entry, or NCE, for short. An NSDB client specifies a search base when forming an LDAP query to search an NSDB for FedFS records. The Distinguished Name of an NCE is that search base.

FedFS junctions contain only the hostname and port of the NSDB service and a UUID. NSDB clients use a standard procedure for probing NSDBs for their NCEs. For the purposes of this discussion, we refer to procedures which retrieve putative NCEs from an NSDB as "NCE discovery mechanisms."

Implementation experience has shown that the NCE discovery mechanism currently defined in sections 4.1 and 5.2.1 of [RFC7532] can be problematic for some LDAP server implementations or deployments. In particular, altering the rootDSE of LDAP servers in any way is an onerous requirement. We present a new mechanism for expressing NCEs that does not require alteration of the rootDSE.

1.1. Altering rootDSEs Considered Hazardous

As described in Section 4.1 of [RFC7532], the LDAP naming context that is superior to an NCE is modified to include the fedfsNsdbContainerInfo object class. This object class carries the mandatory fedfsNceDN attribute, whose value is the Distinguished Name of the NCE that resides under that naming context.

To discover all NCEs on an NSDB, NSDB clients use the procedure described in Section 5.2.1 of [RFC7532]. NSDB clients look for naming context entries that include the fedfsNsdbContainerInfo object class. The fedfsNceDN attribute points directly to the NCE under that root suffix.

Difficulty arises because there is often a high bar to altering an LDAP server's rootDSE.

o Because rootDSE's typically contain read-mostly or even read-only data, an LDAP server implementation may simulate its rootDSE, rather than storing it as regular LDAP records. Code changes would be necessary for such an LDAP server implementation to act as an NSDB.

o For security reasons, some LDAP server deployments may not wish to grant unauthenticated access to their rootDSE information.

o Typically, a separate NSDB administrator account is used to manage NSDB entries under an NCE, but LDAP server administrator privileges (essentially, a root password for the LDAP server) are required if an NCE needs to be added, moved, or deleted. Some

LDAP server administrators may prefer using Access Control Lists (ACLs) to manage access to all FedFS record types, including NCEs.

- The schema specified in [RFC7532] limits the number of NCEs per naming context to one. There does not appear to be any other architected limit in the NSDB protocol that requires a one-to-one relationship between naming context and NCE.

Neither [RFC7532] nor [RFC5716] discuss the motivation for referring to an NCE record via the rootDSE. The use of the current NCE discovery mechanism is simply a convenience that enables the contents of junctions to exclude the LDAP search base when specifying the LDAP record containing the junction's target locations.

Originally, an NSDB Container Entry DN was stored in each junction, along with an FSN UUID and the name of the NSDB where the FSN was stored.

To simplify the contents of junctions, the NCE DN was removed from junctions as the design of the NSDB protocol was finalized. NSDB clients were tasked with discovering the correct LDAP search base to use by searching the target NSDB's rootDSE for the NCE DN.

The current approach is described in Section 5.2.2 of [RFC7532]:

1. The NSDB client obtains a list of the target NSDB's naming contexts.

2. The NSDB client obtains a list of NCEs that reside on the target NSDB by searching each of the NSDB's naming contexts for the fedfsNsdbContainerInfo object class.

3. Finally, the NSDB client forms a series of LDAP queries using each NCE as the search base. The client supplies a filter to retrieve only objects in the fedfsFsl object class.

The search stops when one or more FSLs are returned or when all NCEs have been searched.

Without the fedfsNsdbContainerInfo object class, a root suffix DN is an adequate LDAP search base to use when searching for FSL records. For administrative purposes, however, the construct of NCE is maintained.

2. Simplifying NCE Discovery

Rather than relying on a Distinguish Name stored in an attribute by an administrative tool, an NSDB client might instead use a search query that is natural to LDAP to obtain that DN.

Currently an NSDB Container Entry record is entirely unremarkable, except for its position in the DIT. It includes no special object class, Distinguished Name, or attribute that defines it as an NCE. It becomes an NCE only because its parent naming context points to it via the context's fedfsNceDN attribute.

If each NCE were instead tagged with an object class specific only to NCEs, NSDB clients could discover NCEs simply by filtering on that object class. Then no change to the LDAP server's rootDSE is required.

The following sections provide a protocol specification, similar to [RFC7532], for the new NCE discovery mechanism.

2.1. NSDB Configuration

An NSDB is constructed using an LDAP Directory. This LDAP Directory can have multiple naming contexts. The LDAP Directory's DSA-specific entry (its rootDSE) has a multi-valued namingContext attribute. Each value of the namingContext attribute is the DN of the naming context's root entry (see [RFC4512]).

For each naming context that contains federation entries (e.g., FSNs and FSLs):

1. Typically, all of a naming context's federation entries are descended from one LDAP entry in the Directory Information Tree (DIT). This entry is termed an NSDB Container Entry (NCE).

2. NSDB clients filter with "(objectClass=fedfsNsdbContainerEntry)" to locate the naming context's NCEs. Therefore, every NCE MUST include fedfsNsdbContainerEntry as one of its object classes.

3. NSDB clients search for FSNs with a scope ONE search, based on an NCE. Therefore, all FSN entries under the naming context MUST be immediate children of an NCE.

4. NSDB clients search for FSLs with a scope ONE search, based on an FSN entry. Therefore, all FSL entries under the naming context MUST be immediate children of an FSN entry.

An NCE may have no children. When an NSDB is first created, there are no federation entries aside from the NCE, which therefore has no federation-related descendents.

NSDB administrative operations, such as those described in Section 5.1 of [RFC7532], may use NCEs to demark the position of repositories for federation entries.

If a naming context does not host an NSDB DIT, it MUST NOT contain an entry that includes the fedfsNsdbContainerEntry object class. For example, an LDAP directory might have the following entries:

```
        -+ [root DSE]
         |  namingContext: o=fedfs
         |  namingContext: dc=example,dc=com
         |  namingContext: ou=system
         |
         |
         +---- [o=fedfs]
         |       objectClass: fedfsNsdbContainerEntry
         |
         |
         +---- [dc=example,dc=com]
         |
         +-------- [ou=corp-it,dc=example,dc=com]
         |
         +------------ [ou=fedfs,ou=corp-it,dc=example,dc=com]
         |               objectClass: fedfsNsdbContainerEntry
         |
         |
         +---- [ou=system]
```

In this case, the o=fedfs naming context is also an NSDB Container Entry; the dc=example,dc=com naming context has an NSDB Container Entry at ou=fedfs,ou=corp-it,dc=example,dc=com, and the ou=system naming context 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. To perform the operations described in Section 5.1 of [RFC7532], a user 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 NSDB data. A fileserver performs the operations described in Section 5.2 of [RFC7532] as an unprivileged user.

All NSDB implementations SHOULD use the same schema. At minimum, each MUST use a schema that includes all attributes and objects named in Section 4.2 of [RFC7532] and in Section 2.2. If it is necessary for an implementation to privately extend the schema defined there, consider using one of the following ways:

o Define a fedfsAnnotation key and values (see Section 4.2.1.6 of [RFC7532]). Register the new key and values with IANA.

o Define additional attribute types and object classes, then have entries inherit from a class defined in Section 4.1 of [RFC7532] and in Section 2.2, and from the implementation-defined ones.

2.2. fedfsNsdbContainerEntry

This section is an addendum to the FedFS NSDB schema specified in Section 4.1 of [RFC7532].

The fedfsNsdbContainerEntry object class signifies that an LDAP record acts as an NSDB Container Entry (NCE).

A fedfsNsdbContainerEntry's has no mandatory attributes. A fedfsNsdbContainerEntry's fedfsAnnotation and fedfsDescr attributes are OPTIONAL.

<CODE BEGINS>

```
      ///
      /// objectclass (
      ///     a.b.c.d.e.f.g.h.i NAME 'fedfsNsdbContainerEntry'
      ///     DESC 'Denotes an NSDB Container Entry'
      ///     SUP top AUXILIARY
      ///     MAY (
      ///             fedfsAnnotation
      ///             $ fedfsDescr
      ///     ))
      ///
```

<CODE ENDS>

2.3. NSDB Container Entry (NCE) Enumeration

To find NCEs residing on the NSDB nsdb.example.com, an NSDB client SHOULD do the following:

<CODE BEGINS>

```
      nce_list = empty
      connect to the LDAP directory at nsdb.example.com
      for each namingContext value $BAR in the root DSE
          /* $BAR is a DN */
          query for fedfsNsdbContainerEntry objects under $BAR
          include the DN of such objects on the nce_list
```

<CODE ENDS>

The [RFC4516] LDAP URL for the search query inside the loop would be:

```
      ldap://nsdb.example.com:389/$BAR??sub?
              (objectClass=fedfsNsdbContainerEntry)
```

3. Backwards Compatibility

3.1. NSDB server compatibility

NSDBs might serve NSDB client populations that contain clients that comply only with [RFC7532] as well as clients that use the query described in Section 2.3.

In this case, both the mechanism described in Section 2.1 and the mechanism described in [RFC7532] can be concurrently implemented in the NSDB to support both types of NSDB client.

To remain compatible with NSDB clients that comply with [RFC7532], each naming context on such an NSDB MUST contain either zero or one NCE records.

3.2. NSDB client compatibility

During a transition period to the new NCE discovery mechanism, NSDB clients may have to resolve FSNs on NSDBs that comply with [RFC7532] as well as NSDBs that have deployed the mechanism described in Section 2.1.

In this case, an NSDB client can try the query described in Section 2.3 first. If no NCE is found by this method, the NSDB client can try the query described in Section 5.2.1 of [RFC7532].

An NSDB client that encounters more than one NCE record under a naming context MUST return FEDFS_ERR_NSDB_NONCE if it does not support multiple NCEs per naming context.

4. Security Considerations

To maintain separation of privileges, privileged users that are permitted to perform NSDB administrative operations should not be permitted access to other areas of the LDAP server's DIT. The use of LDAP ACLs is recommended to permit unauthenticated access to FedFS records (FSNs, FSLs, and NCEs) while restricting the ability to change these records to one or a few privileged user accounts.

5. IANA Considerations

5.1. LDAP Descriptor Deprecation

IANA is to update the registry entitled "FedFS Object Identifiers" for the purpose of recording the change in status of existing FedFS Object Identifiers (OIDs) mentioned by this document. This includes the fedfsNceDN entry (OID 1.3.6.1.4.1.31103.1.14) and the fedfsNsdbContainerInfo entry (OID 1.3.6.1.4.1.31103.1.1001). The designation of "historic" is to be added to the "Reference" column of both entries.

In accordance with Section 5.2 of [RFC4520], object identifier descriptors for the fedfsNsdbContainerInfo object class and the fedfsNceDN attribute are to be marked as "historic in nature" in IANA's LDAP parameters "Object Identifier Descriptors" table, after Expert Review.

5.2. LDAP Descriptor Registration

In accordance with Section 3.4 and Section 4 of [RFC4520], object identifier descriptors for the new fedfsNsdbContainerEntry object class defined in this document will be assigned and 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-cel-nfsv4-federated-fs-nce
Author/Change Controller: IESG (iesg@ietf.org)

Object Identifier: TBD
Descriptor (short name): fedfsNsdbContainerEntry
Usage: object class

6. Acknowledgements

The author of this document gratefully acknowledges the contributions and patience of Rob Thurlow, Tom Haynes, and David Noveck. This work would not have been possible without the efforts of the authors and contributors to [RFC7532].

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC4512] Zeilenga, K., Ed., "Lightweight Directory Access Protocol (LDAP): Directory Information Models", RFC 4512, DOI 10.17487/RFC4512, June 2006, <http://www.rfc-editor.org/info/rfc4512>.

[RFC4516] Smith, M., Ed. and T. Howes, "Lightweight Directory Access Protocol (LDAP): Uniform Resource Locator", RFC 4516, DOI 10.17487/RFC4516, June 2006, <http://www.rfc-editor.org/info/rfc4516>.

[RFC4520] Zeilenga, K., "Internet Assigned Numbers Authority (IANA) Considerations for the Lightweight Directory Access Protocol (LDAP)", BCP 64, RFC 4520, DOI 10.17487/RFC4520, June 2006, <http://www.rfc-editor.org/info/rfc4520>.

[RFC7532] Lentini, J., Tewari, R., and C. Lever, Ed., "Namespace Database (NSDB) Protocol for Federated File Systems", RFC 7532, DOI 10.17487/RFC7532, March 2015, <http://www.rfc-editor.org/info/rfc7532>.

7.2. Informative References

[RFC5716] Lentini, J., Everhart, C., Ellard, D., Tewari, R., and M. Naik, "Requirements for Federated File Systems", RFC 5716, DOI 10.17487/RFC5716, January 2010, <http://www.rfc-editor.org/info/rfc5716>.

Authors' Addresses

Charles Lever
Oracle Corporation
1015 Granger Avenue
Ann Arbor, MI 48104
USA

Phone: +1 734 274 2396
Email: chuck.lever@oracle.com

Simo Sorce
Red Hat, Inc.

Email: simo.sorce@redhat.com