Network File System Version 4 T. Haynes
Internet-Draft Hammerspace
Intended status: Standards Track 1 February 2026
Expires: 5 August 2026

Adding an Uncacheable Directory-Entry Metadata Attribute to NFSv4.2 draft-ietf-nfsv4-uncacheable-directories-04

Abstract

Network File System version 4.2 (NFSv4.2) clients commonly cache directory entries (dirents) to improve performance. While effective in many cases, such caching can prevent servers from enforcing per-user access controls on directory entries and up-to-date directory entry attributes such as size and timestamps. This document introduces a new uncacheable dirent metadata attribute for NFSv4.2 that allows servers to advise clients that caching of directory entry metadata is unsuitable. This enables servers to present directory contents based on user-specific access permissions while remaining compatible with existing NFSv4.2 clients.

Note to Readers

Discussion of this draft takes place on the NFSv4 working group mailing list (nfsv4@ietf.org), which is archived at https://mailarchive.ietf.org/arch/search/?email_list=nfsv4. Source code and issues list for this draft can be found at https://github.com/ietf-wg-nfsv4/uncacheable-directories.

Working Group information can be found at https://github.com/ietf-wg-nfsv4.

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 https://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 5 August 2026.

Table of Contents

1. Introduction

Clients of remote filesystems commonly cache directory entries (dirents) to improve performance. This caching is typically shared across users on the client and assumes that directory contents and access permissions are uniform across users.

In this document, the term directory is used to describe the context in which directory entries are retrieved. The uncacheable dirent metadata attribute applies to the caching of directory entry (dirent) metadata, including names and associated file object metadata such as size and timestamps. It does not prohibit caching of the directory object itself, nor does it affect caching of file data.

Access Based Enumeration (ABE) [MS-ABE], as implemented in the Server Message Block (SMB) [MS-SMB2] and deployed in implementations such as Samba [Samba], restricts directory visibility based on the access permissions of the requesting user. Implementing similar behavior in NFSv4.2 requires server involvement, as clients may not have sufficient information to evaluate permissions based on identity mappings, ACLs, or server-local policy.

While effective in environments with centralized identity and server-driven enumeration, the SMB ABE model tightly couples directory enumeration with authorization and requires per-user directory views that are not safely cacheable across users. This approach does not generalize well to NFS, where directory contents and metadata are traditionally shared and cached. The uncacheable dirent metadata attribute allows servers to ensure correctness of directory-entry metadata visibility and attributes without mandating a specific enumeration or authorization model.

Even in the absence of ABE, caching of directory entry metadata can result in incorrect size and timestamp information when files are modified concurrently, reducing the effectiveness of uncacheable file data semantics when directory entry metadata is stale. This can lead to applications observing inconsistent metadata and data views even when file data caching is disabled.

With a remote filesystem, the client typically caches directory entries (dirents) locally to improve performance. This cooperation succeeds because both the server and client operate under POSIX semantics ([POSIX.1]) and agree to interpretation of mode bits with respect to the uid and gid in NFSv3 [RFC1813]. For NFSv4.2, these would respectively be the mode, owner, and owner_group attributes defined in Section 5 of [RFC8881]. Note that this cooperation does not apply to Access Control List (ACLs) entries as NFSv4.2 does not implement a strict POSIX style ACL.

NFSv4.2 does implement NFSv4.1 ACLs, which are enforced on the server and not the client. As such, ACL enforcement requires the client to bypass the dirent cache to have checks done when a new user attempts to access the dirent.

Another consideration is that not all server implementations natively support SMB. Instead, they layer Samba on top of the NFSv4.2 service. The attributes of hidden, system, and offline have already been introduced in the NFSv4.2 protocol to support Samba. The Samba implementation can utilize these attributes to provide SMB semantics. While private protocols can supply these features, it is better to drive them into open standards.

Another concept that can be adapted from SMB is that of ABE, which is commonly used to control the visibility of directory entries. Under the POSIX model, this can be done on the client and not the server. However, that only works with uid, gid, and mode bits. If we consider identity mappings, ACLs, and server local policies, then the determination of ABE and directory entry visibility is best performed on the server.

Since cached dirents are shared by all users on a client, and the client cannot determine access permissions for individual dirents, all users are presented with the same set of attributes. To address this, this document introduces the uncacheable dirent metadata attribute. This attribute advises the client not to cache directory entry metadata for a file or directory object. Consequently, each time a client queries for these attributes, the server's response can be tailored to the specific user making the request.

This document introduces the uncacheable dirent metadata attribute to NFSv4.2 to allow servers to advise clients that caching of directory-entry metadata is unsuitable. Using the process detailed in [RFC8178], the revisions in this document become an extension of NFSv4.2 [RFC7862]. They are built on top of the external data representation (XDR) [RFC4506] generated from [RFC7863].

1.1. Definitions

Access Based Enumeration (ABE) When servicing a READDIR or GETATTR operation, the server provides results based on the access permissions of the user making the request.

dirent A directory entry representing a file or subdirectory and its associated attributes.

dirent caching A client-side cache of directory entry names and associated file object metadata, used to avoid repeated directory lookup and attribute retrieval.

uncacheable dirent metadata attribute An NFSv4.2 file attribute that

advises clients not to cache directory-entry metadata associated with file objects, including names, size, timestamps, and visibility.

This document assumes familiarity with NFSv4.2 operations, attributes, and error handling as defined in [RFC8881] and [RFC7862].

1.2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Caching of Directory-Entry Metadata

The fattr4_uncacheable_file_data attribute is a read-write file attribute and has a data type of boolean. The attribute is not set on individual file objects and applies only to directory-entry metadata returned from the directory on which it is set.

The uncacheable dirent metadata attribute enables correct presentation of directory entry visibility and attributes, including but not limited to Access Based Enumeration (ABE). As such, it is an OPTIONAL attribute to implement for NFSv4.2. If both the client and the server support this attribute, the client MUST to bypass caching of directory-entry metadata for directories marked as uncacheable.

This document specifies the required observable behavior rather than mandating a particular internal implementation strategy. Clients MAY employ more sophisticated mechanisms, such as per-user directory entry caching, provided that the externally visible behavior is equivalent to not caching directory-entry metadata across users.

Allowing clients to set this attribute provides a portable mechanism to request that directory-entry metadata not be cached, without requiring changes to application behavior or out-of-band administrative configuration.

A client can determine whether the uncacheable dirent metadata attribute is supported for a given directory by issuing a GETATTR request and examining the returned attribute list.

The only way that the server can determine that the client supports the attribute is if the client sends either a GETATTR or a SETATTR with the uncacheable dirent metadata attribute.

The uncacheable dirent metadata attribute governs caching behavior of directory-entry metadata returned by READDIR and related operations, not the directory object itself.

Suppressing caching of file data alone is insufficient to guarantee correct behavior if directory-entry metadata such as size and timestamps remains cached. The uncacheable dirent metadata attribute addresses a different aspect of client-side caching than fattr4_uncacheable_file_data ([I-D.ietf-nfsv4-uncacheable-files]). The file data attribute governs caching of file contents, while the dirent metadata attribute governs caching of directory-entry metadata. In some workloads, disabling only one form of caching may be insufficient to ensure correct behavior, but the attributes are independent and may be used separately.

This attribute does not define behavior for positive or negative name caching or for caching of LOOKUP results outside the scope of directory-entry metadata returned by READDIR and related operations.

Directory delegations do not address per-user directory-entry metadata visibility and therefore cannot replace the semantics defined by the uncacheable dirent metadata attribute.

2.1. Uncacheable Directory-Entry Metadata

The fattr4_uncacheable_file_data attribute is a read-write boolean attribute that applies on a per-file basis to regular files (NF4REG). Authorization to query or modify this attribute is governed by existing NFSv4.2 authorization mechanisms.

If a directory object has the uncacheable dirent metadata attribute set, the client is advised not to cache directory entry metadata. In such cases, the client retrieves directory entry attributes from the server for each request, allowing the server to evaluate access permissions based on the requesting user. Clients are advised not to share cached dirent attributes between different users.

The uncacheable dirent metadata attribute does not modify the semantics of the NFSv4.2 change attribute. Clients MUST continue to use the change attribute to detect directory modifications and to determine when directory contents may have changed, even when directory-entry metadata caching is suppressed. Suppressing caching of directory-entry metadata does not remove the need for change-based validation.

Servers SHOULD assume that clients which do not query or set this attribute may cache directory-entry metadata, and therefore SHOULD NOT rely on this attribute for correctness unless client support is confirmed.

Authorization to set or modify this attribute is governed by existing NFSv4.2 authorization mechanisms.

If a client holds a directory delegation for a directory that becomes marked with the uncacheable dirent metadata attribute, the server is expected to ensure that the client observes the updated attribute value. A server MAY recall an existing directory delegation in order to enforce the semantics of this attribute. Clients that observe the attribute set while holding a directory delegation MUST ensure that directory-entry metadata is not cached inconsistently with the attribute semantics.

Because this attribute provides advisory guidance rather than mandatory access control, servers cannot rely on client compliance for security enforcement in adversarial environments.

3. Example: Directory Enumeration With and Without Dirent Metadata Caching

This example illustrates the difference in client-visible behavior when directory-entry metadata caching is enabled versus when the uncacheable dirent metadata attribute is set on a directory.

3.1. Classic Directory Enumeration (Directory-Entry Metadata Cached)

In this scenario, the client caches directory-entry metadata obtained from the server and reuses it for subsequent users.

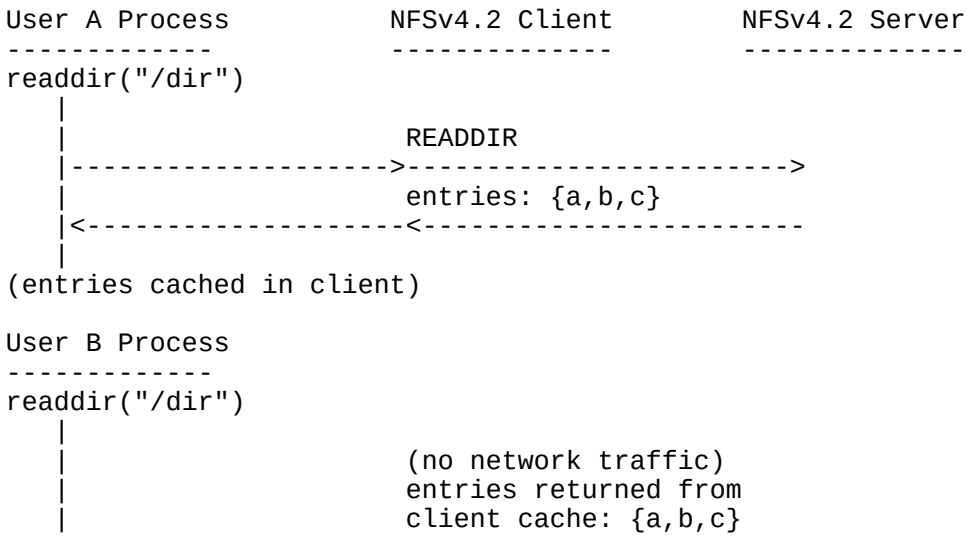

Figure 1: Directory-Entry Metadata Cached

In this case, Figure 1 shows directory-entry metadata retrieved on behalf of User A is reused to satisfy a directory read for User B. This behavior is typical of legacy NFS clients and maximizes performance, but it can result in incorrect or unauthorized directory views in multi-user or multi-protocol environments.

3.2. Directory Enumeration With Uncacheable Dirent Metadata

In this scenario, the directory has the uncacheable dirent metadata attribute set. The client does not retain directory-entry metadata across directory reads for different users.

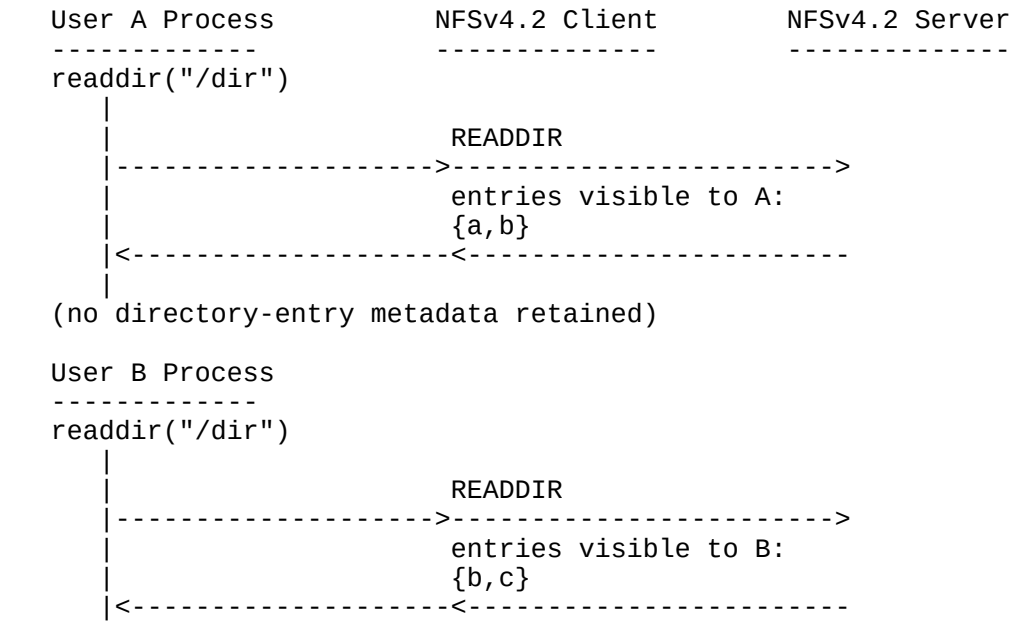

Figure 2: Directory-Entry Metadata Not Cached

In this case, Figure 2 shows each directory read results in a READDIR operation sent to the server, ensuring that directory-entry metadata reflects the current visibility and attributes appropriate to the requesting user. The client may still cache other information, provided the externally observable behavior is equivalent to not caching directory-entry metadata.

3.3. Discussion

This example demonstrates that the uncacheable dirent metadata attribute does not mandate a particular client implementation, but it does require that directory-entry metadata retrieved for one user MUST NOT be reused to satisfy directory reads for another user. The attribute ensures correctness and interoperability in environments where directory contents or visibility may differ across users, clients, or protocols.

4. XDR for Uncacheable Dirents Attribute

```
///
/// typedef bool            fattr4_uncacheable_dirent_metadata;
///
/// const FATTR4_UNCACHEABLE_DIRENT_METADATA   = 88;
///
```

5. Extraction of XDR

This document contains the external data representation (XDR) [RFC4506] description of the uncacheable dirent metadata attribute. The XDR description is presented in a manner that facilitates easy extraction into a ready-to-compile format. To extract the machine-readable XDR description, use the following shell script:

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

For example, if the script is named 'extract.sh' and this document is named 'spec.txt', execute the following command:

```
sh extract.sh < spec.txt > uncacheable_prot.x
```

This script removes leading blank spaces and the sentinel sequence '///' from each line. XDR descriptions with the sentinel sequence are embedded throughout the document.

Note that the XDR code contained in this document depends on types from the NFSv4.2 nfs4_prot.x file (generated from [RFC7863]). This includes both nfs types that end with a 4, such as offset4, length4, etc., as well as more generic types such as uint32_t and uint64_t.

While the XDR can be appended to that from [RFC7863], the code snippets should be placed in their appropriate sections within the existing XDR.

6. Security Considerations

This attribute is not intended to provide a security boundary or to replace server-enforced access control. Its primary purpose is to improve correctness and interoperability in environments where directory-entry metadata visibility varies across users or protocols. Servers MUST NOT rely on this mechanism alone to prevent unauthorized access to directory entries.

Authorization to set or modify the fattr4_uncacheable_file_data attribute is governed by existing NFSv4.2 authorization mechanisms. Servers MAY restrict modification of this attribute based on local policy, file ownership, or access control rules. This document does not define a new authorization model.

The discussion of users in this section is independent of the specific user identity representation employed by the client or server. This document does not distinguish between users identified via NFSv4.2 user@domain strings, RPC authentication identities, or

local operating system user identifiers. The uncacheable dirent metadata attribute does not alter NFSv4.2 authentication or authorization semantics and does not depend on any particular user identity model.

For a given user A, a client MUST NOT make access decisions for uncacheable dirents retrieved for another user B. These decisions MUST be made by the server. If the client is Labeled NFS aware ([RFC7204]), then the client MUST locally enforce the MAC security policies.

The concerns described above primarily apply to multi-user clients that cache directory-entry metadata on behalf of multiple users. Single-user clients may not be subject to these risks, but the attribute semantics remain the same regardless of client usage model.

The uncacheable dirent metadata attribute allows dirents to be annotated such that attributes are presented to the user based on the server’s access control decisions.

7. IANA Considerations

This document has no IANA actions.

8. References

8.1. Normative References

[I-D.ietf-nfsv4-uncacheable-files]
Haynes, T., "Adding an Uncacheable File Data Attribute to NFSv4.2", Work in Progress, Internet-Draft, draft-ietf-nfsv4-uncacheable-files-03, 13 January 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-nfsv4-uncacheable-files-03>.

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

[RFC4506] Eisler, M., Ed., "XDR: External Data Representation Standard", STD 67, RFC 4506, DOI 10.17487/RFC4506, May 2006, <https://www.rfc-editor.org/rfc/rfc4506>.

[RFC7204] Haynes, T., "Requirements for Labeled NFS", RFC 7204, DOI 10.17487/RFC7204, April 2014, <https://www.rfc-editor.org/rfc/rfc7204>.

[RFC7862] Haynes, T., "Network File System (NFS) Version 4 Minor Version 2 Protocol", RFC 7862, DOI 10.17487/RFC7862, November 2016, <https://www.rfc-editor.org/rfc/rfc7862>.

[RFC7863] Haynes, T., "Network File System (NFS) Version 4 Minor Version 2 External Data Representation Standard (XDR) Description", RFC 7863, DOI 10.17487/RFC7863, November 2016, <https://www.rfc-editor.org/rfc/rfc7863>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC8178] Noveck, D., "Rules for NFSv4 Extensions and Minor Versions", RFC 8178, DOI 10.17487/RFC8178, July 2017, <https://www.rfc-editor.org/rfc/rfc8178>.

[RFC8881] Noveck, D., Ed. and C. Lever, "Network File System (NFS) Version 4 Minor Version 1 Protocol", RFC 8881, DOI 10.17487/RFC8881, August 2020, <https://www.rfc-editor.org/rfc/rfc8881>.

8.2. Informative References

[MS-ABE] Microsoft, "Access-Based Enumeration (ABE) Concepts", May 2009, <https://techcommunity.microsoft.com/blog/askds/access-based-enumeration-abe-concepts-part-1-of-2/400435>.

[MS-SMB2] Microsoft Corporation, "Server Message Block (SMB) Protocol Versions 2 and 3", Microsoft MS-SMB2, n.d., <https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-smb2/>.

[POSIX.1] IEEE, "The Open Group Base Specifications Issue 7", IEEE Std 1003.1, 2013 Edition , 2013.

[RFC1813] Callaghan, B., Pawlowski, B., and P. Staubach, "NFS Version 3 Protocol Specification", RFC 1813, DOI 10.17487/RFC1813, June 1995, <https://www.rfc-editor.org/rfc/rfc1813>.

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2", FYI 36, RFC 4949, DOI 10.17487/RFC4949, August 2007, <https://www.rfc-editor.org/rfc/rfc4949>.

[RFC9754] Haynes, T. and T. Myklebust, "Extensions for Opening and Delegating Files in NFSv4.2", RFC 9754, DOI 10.17487/RFC9754, March 2025, <https://www.rfc-editor.org/rfc/rfc9754>.

[Samba] Samba Team, "Samba Project", n.d., <https://www.samba.org/>.

Acknowledgments

Trond Myklebust, Mike Snitzer, Jon Flynn, Keith Mannthey, and Thomas Haynes all worked on the prototype at Hammerspace.

Rick Macklem, Chuck Lever, and Dave Noveck reviewed the document.

Chris Inacio, Brian Pawlowski, and Gorry Fairhurst helped guide this process.

Author’s Address

Thomas Haynes
Hammerspace
Email: loghyr@gmail.com