

NFSv4  
Internet-Draft  
Intended status: Informational  
Expires: September 11, 2016

D. Noveck  
HPE  
March 10, 2016

NFS Protocol Extension: Retrospect and Prospect  
draft-dnoveck-nfs-extension-04

## Abstract

This document surveys the processes by which the NFS protocols have been extended in the past, including all of the NFS major and minor versions. It also looks forward to methods of protocol extension that might be used by NFS in the future.

A particular focus is on how the minor versioning model of NFSv4 has worked and what is being done to enhance version management within NFSv4. The work already done in the new NFSv4 version management document is summarized, and there is a discussion of further issues the working group will need to address in moving that work forward.

## 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 11, 2016.

## Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

Internet-Draft

nfs-extension

March 2016

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

<a href="#">1.</a>	<a href="#">Introduction . . . . .</a>	<a href="#">3</a>
<a href="#">1.1.</a>	<a href="#">Use of Key Words Defined in <a href="#">RFC2119</a> . . . . .</a>	<a href="#">3</a>
<a href="#">2.</a>	<a href="#">Protocol Extension . . . . .</a>	<a href="#">5</a>
<a href="#">3.</a>	<a href="#">Protocol Extension Mechanisms . . . . .</a>	<a href="#">5</a>
<a href="#">3.1.</a>	<a href="#">Specific Protocol Mechanisms Designed for Extension . . .</a>	<a href="#">6</a>
<a href="#">3.2.</a>	<a href="#">Protocol Extension by XDR Replacement . . . . .</a>	<a href="#">6</a>
<a href="#">3.3.</a>	<a href="#">Protocol Extension by XDR Extension . . . . .</a>	<a href="#">7</a>
<a href="#">3.4.</a>	<a href="#">Combination of Protocol Extension Mechanisms . . . . .</a>	<a href="#">8</a>
<a href="#">3.5.</a>	<a href="#">Switching Extension Mechanisms . . . . .</a>	<a href="#">8</a>
<a href="#">4.</a>	<a href="#">Pre-IETF NFS Versioning . . . . .</a>	<a href="#">9</a>
<a href="#">4.1.</a>	<a href="#">The Pre-IETF NFS Environment . . . . .</a>	<a href="#">9</a>
<a href="#">4.2.</a>	<a href="#">Transition from NFSv2 to NFSv3 . . . . .</a>	<a href="#">9</a>
<a href="#">5.</a>	<a href="#">Initial Work on NFSv4 Within IETF . . . . .</a>	<a href="#">10</a>
<a href="#">5.1.</a>	<a href="#">Transition from NFSv3 to NFSv4.0 . . . . .</a>	<a href="#">10</a>
<a href="#">5.2.</a>	<a href="#">NFSv4 and XDR Extensibility . . . . .</a>	<a href="#">12</a>
<a href="#">5.3.</a>	<a href="#">Initial Minor Versioning Model for NFSv4 . . . . .</a>	<a href="#">13</a>
<a href="#">5.4.</a>	<a href="#">Goals of Minor Versioning for NFSv4 . . . . .</a>	<a href="#">15</a>
<a href="#">6.</a>	<a href="#">Use of Minor Versioning in NFSv4 . . . . .</a>	<a href="#">16</a>
<a href="#">6.1.</a>	<a href="#">Transition from NFSv4.0 to NFSv4.1 . . . . .</a>	<a href="#">16</a>
<a href="#">6.2.</a>	<a href="#">Transition from NFSv4.1 to NFSv4.2 . . . . .</a>	<a href="#">22</a>
<a href="#">6.3.</a>	<a href="#">Evolution of Minor Versioning Model within NFSv4 . . . .</a>	<a href="#">23</a>
<a href="#">6.4.</a>	<a href="#">Review of NFSv4 Versioning through NFSv4.2 . . . . .</a>	<a href="#">24</a>
<a href="#">7.</a>	<a href="#">Inherited NFSv4 Versioning Approach . . . . .</a>	<a href="#">25</a>
<a href="#">7.1.</a>	<a href="#">Non-XDR-based Changes . . . . .</a>	<a href="#">25</a>
<a href="#">7.2.</a>	<a href="#">The Role of Minor Version Number in NFSv4 . . . . .</a>	<a href="#">25</a>
<a href="#">7.3.</a>	<a href="#">Inherited NFS Versioning Practices . . . . .</a>	<a href="#">27</a>
<a href="#">7.4.</a>	<a href="#">Problems with Inherited NFS Versioning Approach . . . .</a>	<a href="#">27</a>
<a href="#">8.</a>	<a href="#">Formulating a New NFSv4 Extension Approach . . . . .</a>	<a href="#">30</a>
<a href="#">9.</a>	<a href="#">Current Versioning-related Work . . . . .</a>	<a href="#">31</a>
<a href="#">9.1.</a>	<a href="#">Creation of New NFSv4 Version Management Document . . .</a>	<a href="#">31</a>
<a href="#">9.2.</a>	<a href="#">Related Changes to Other Documents . . . . .</a>	<a href="#">31</a>
<a href="#">9.3.</a>	<a href="#">Further work on New NFSv4 Versioning Document . . . . .</a>	<a href="#">32</a>
<a href="#">9.4.</a>	<a href="#">Issues Needing further discussion . . . . .</a>	<a href="#">34</a>
<a href="#">10.</a>	<a href="#">Looking Back . . . . .</a>	<a href="#">35</a>
<a href="#">11.</a>	<a href="#">Looking Forward . . . . .</a>	<a href="#">38</a>

<a href="#">12.</a>	Security Considerations . . . . .	<a href="#">39</a>
<a href="#">13.</a>	IANA Considerations . . . . .	<a href="#">39</a>
<a href="#">14.</a>	References . . . . .	<a href="#">39</a>
<a href="#">14.1.</a>	Normative References . . . . .	<a href="#">39</a>
<a href="#">14.2.</a>	Informative References . . . . .	<a href="#">39</a>

<a href="#">Appendix A.</a>	Acknowledgements . . . . .	<a href="#">41</a>
Author's Address	. . . . .	<a href="#">41</a>

## [1.](#) Introduction

This document examines the subject of protocol extension within the NFS family of protocols. In order to better understand the issues that exist going forward with NFSv4, we examine the history of protocol extension throughout the development of NFS including

- o The pre-IETF period
- o The development of successive NFSv4 minor versions, under minor versioning rules originally defined in [[RFC3530](#)] and subsequently modified based on the working group's then-current needs.
- o The consolidation of version management rules in [[NFSv4-vers](#)] along with the other changes made in that document to make NFSv4 version management more flexible.

With this history in mind, we discuss the issues that the working group needs to address in order to define and adopt a consistent and comprehensive approach to version management for the NFSv4 protocols.

### [1.1.](#) Use of Key Words Defined in [RFC2119](#)

It is intended that these key words be used in this document in a way that is consistent with their definition in [[RFC2119](#)].

However, because of the considerations noted below, there are some issues that readers need to be aware of.

It is not altogether clear whether these keywords are to be limited to requirements regarding protocol implementations, or whether they can reasonably be used in specifying guidance directed at future potential RFCs. In many cases, these terms are defined in ways that

strongly suggest that they are meant to apply to implementations and much of the discussion seems to make sense only in that context. On the other hand, there is no explicit statement to that effect.

As an example of the difficulties that can result in trying to resolve this question, consider the statement in [[RFC2119](#)] saying that these imperatives "MUST only be used where it is actually required for interoperation or to limit behavior which has potential for causing harm (e.g., limiting retransmissions)." On the one hand, this is a case in which these terms are directed at a future document, rather than implementations. However, it seems as if this use is itself not consistent with what it requires of others. While

the misuse of these terms is undesirable and it is best to avoid such misuse, it is hard to see exactly how that is "actually required for interoperation or to limit behavior which has potential for causing harm."

The nature of this document is such that it does not directly specify implementation requirements. In some cases it discusses potential requirements that an RFC derived from [[NFSv4-vers](#)]) might direct to implementations. In other cases, requirements are one step farther removed from protocol implementations. For example, the version management RFC might specify the kinds of requirements that future minor version definition documents and feature definition documents might specify.

The documents that are discussed herein, have at times used these key words in a way that seems to be at variance with the definitions in [[RFC2119](#)]. For details, see Sections [6.1](#) and [9.2](#).

The reader should be aware of our practices in this document regarding these terms. We only apply these upper-case key words to directions regarding protocol implementations, rather than to guidance intended for those writing RFCs. While it is possible that there could be cases in which directions targeted at future RFC's are necessary to assure interoperation etc., we know of no actual instances.

In this document, these keywords will often appear as quotations from existing documents, either direct or indirect. Readers should be aware that it is possible that some such uses might not be in accord

with [\[RFC2119\]](#).

In other cases, these keywords will be in the context of proposals/suggestions of what future RFCs could or might say regarding certain issues.

The use of these terms as part of minor versioning rules poses some troublesome issues in the context of this document. These rules have appeared in [\[RFC3010\]](#), [\[RFC3530\]](#), [\[RFC5661\]](#), and [\[NFSv42\]](#). They have also appeared in various drafts of [\[NFSv4-vers\]](#), and in various drafts leading up to [\[RFC7530\]](#). In presenting these rules, there have been multiple switches between the [RFC2119](#) keywords and corresponding lower-case terms.

Use of the terms "OPTIONAL" and "REQUIRED" as feature statuses would conform to our general practice since a feature status is essentially a way in which a minor version specification RFC might REQUIRE (or not) implementations to support a given feature. Unfortunately, "RECOMMENDED" would not fit that model very well. Also, it is

difficult to see how the same feature can be "OPTIONAL" in one minor version and "REQUIRED" in another given the meanings that [\[RFC2119\]](#) assigns to these terms.

In this document, in the interests of uniformity, we will use lower-case terms for feature statuses, except when we are referring to what is said in a particular document which used the upper-case keywords.

## [2.](#) Protocol Extension

Protocols often require means by which they can be extended. Such extensions may be needed to meet new requirements, to correct protocol weaknesses exposed by experience, or even to correct protocol bugs (as becomes more probable when protocols are published as RFC's without fully fleshed-out implementations).

We need to distinguish here between protocol "extension" and "versioning". Versioning is a form of protocol extension but not every form of protocol extension can be accommodated within a versioning paradigm.

When a versioning paradigm is in place, groups of extensions are

conceived of as ordered, allowing extensions in subsequent versions to build upon those in previous versions. When multiple extensions are combined into a single version, each of the extensions may be built assuming that the others will be present as well. In such cases, there can be the opportunity to make design changes in the protocol, allowing elements of the protocol to be restructured, sometimes in major ways.

When a versioning paradigm is in effect and extensions are optional, extensions cannot build upon one another, since the presence of any particular extension cannot be assumed. In such cases, the ability to restructure the protocol is reduced, but smaller changes may be introduced more easily.

In this latter case, it is not clear that the word "versioning" is appropriate. Nevertheless, in this document, we will, as in the phrase "NFSv4 minor versioning," use the existing terminology without necessarily subscribing to the view that "versioning" is the appropriate description.

### [3.](#) Protocol Extension Mechanisms

Some factors that are often relevant in deciding on the means by which a protocol will be extended:

Noveck	Expires September 11, 2016	[Page 5]
--------	----------------------------	----------

---

Internet-Draft	nfs-extension	March 2016
----------------	---------------	------------

- o Whether extensions are to be individually selectable (i.e. optional) or assumed to be always present, allowing one to build upon another earlier one.
- o The size and scope of extensions that will be made.
- o Compatibility issues with existing implementations.
- o Issues that relate to ensuring that when individual extensions, separately arrived at, are each optionally allowed, the extensions used are compatible and can be used effectively together.
- o The overall implementation framework. For example, RPC-based protocols may do extension by means of the RPC version mechanism.

Because NFS is layered on RPC and is described using an XDR description, the presentation of extension mechanisms below reflects those facts. Although XDR is mentioned in Sections [3.2](#) and [3.3](#) the reader should not assume that particular aspects of XDR itself are critical to the discussion. Similar extension approaches and analogous considerations would apply to other similar methods of protocol description.

### [3.1.](#) Specific Protocol Mechanisms Designed for Extension

Often, protocols will be designed with specific mechanisms, designed to allow protocol extension. An example is the provision for TCP options (see [[RFC0793](#)] and [[RFC2780](#)].) Most often, such mechanisms are designed to allow individual extensions to be designed and implemented independently, with any dependency relations between extensions specified separately and not enforced by the extension mechanism itself.

### [3.2.](#) Protocol Extension by XDR Replacement

RPC-based protocols may, and often do, provide for protocol extension by replacing the XDR for one version with that for another. In this case the new protocol version could be represented by a new RPC protocol number but the more common pattern is to use the RPC versioning mechanism to manage selection of the proper protocol variant. The use of the RPC versioning mechanism enforces a versioning paradigm of this sort on protocols using this extension mechanism.

This extension mechanism allows very extensive protocol changes, up to and including the replacement of one protocol by an entirely different one. For some kinds of protocol extensions, this seems the only way to effect the change.

This approach was used to effect a transition between NFSv2 and NFSv3 (See [Section 4.2](#)) and between NFSv3 and NFSv4.0 (See [Section 5.1](#)).

### [3.3.](#) Protocol Extension by XDR Extension

It is possible to replace an XDR definition by one which is an extension in the sense that,

- o The set of messages described by the second definition is a superset of that described by the first.
- o Each message within the set of messages described by the first definition is recognized as having exactly the same structure/interpretation by the second definition.
- o Each message within the set of messages described by the second definition but not the first definition must be recognized as part of an unsupported extension.

Within an XDR/RPC framework, extensions can be arrived at by:

- o Addition of previously unspecified RPC operation codes.
- o Addition of new, previously unused, values to existing enums.
- o Addition of previously unassigned bit values to a flag word.
- o Addition of new cases to existing switches, provided that the existing switch did not contain a default case.

Such an extension relation between XDR descriptions is reflexive, antisymmetric, and transitive and thus defines a partial order. In practice, provisions have to be made to make sure that two extensions of the same description are compatible (i.e. either one is an extension of the other, or there is a another description that is a valid extension of both).

To put things in concrete terms, such compatibility can be assured if measures are taken to ensure:

- o That the same operation code is not used for two different RPC operations.
- o That the same enum value is not assigned two different meanings.
- o That the same bit flag value is not assigned two different meanings.

- o That whenever the same case value is added to the same switch in



two different extensions, the content assigned to the two matching added cases is the same.

- o That default cases are never added to existing switches.

In contrast to XDR replacement discussed in [Section 3.2](#), use of XDR extension remains atypical and is not supported by the XDR/RPC infrastructure. It is important to keep this fact in mind, as we discuss the use of XDR extension by NFSv4 in [Section 5.2](#) and beyond.

#### [3.4.](#) Combination of Protocol Extension Mechanisms

It is possible to use multiple such means of protocol extension simultaneously. When this is done, the result is a composite extension mechanism. For example, if the XDR replacement or XDR extension mechanism is adopted, a protocol-specific mechanism can be added to it, if the protocol-specific mechanism is built on objects whose XDR definition is sufficiently generic. (e.g. opaque arrays or feature bitmasks).

It can be argued that the NFSv4 attribute model provides such an embedded protocol-specific extension mechanism, since sets of attribute values are specified as XDR opaque arrays and attribute sets are specified as variable-length arrays of 32-bit integers allowing new attribute bits to be defined outside of the XDR definition framework.

Note that there exists specification text that suggests that attributes are part of the XDR specification, making it hard to reach a firm conclusion on the matter. However, the resolution of this question does not affect the other matters discussed below, since, in either case, we have an extension mechanism that allows optional extensions.

#### [3.5.](#) Switching Extension Mechanisms

While it is possible to choose multiple extension mechanisms for different sorts of extensions, based on their characteristics, protocols typically do not take advantage of that flexibility.

On the other hand, protocols do, as NFS has done, change their preferred extension mechanisms in response to long-term changes in requirements. However, once having done so, they rarely switch back. Changing extension mechanisms is a big step, both conceptually and in implementation terms, and is not commonly repeated.

In the case of NFS, the possibility of returning to an XDR replacement model (as a major version change) has always been available, but there has never been any significant interest in taking this step. It remains unlikely, although not impossible, that this could happen in the future.

#### [4.](#) Pre-IETF NFS Versioning

##### [4.1.](#) The Pre-IETF NFS Environment

NFSv2 and NFSv3 were described by the informational RFC's [[RFC1094](#)] and [[RFC1813](#)]. These documents each described existing interoperating client and server implementations. Thus they started with running code. If there was a rough consensus in effect, it was that these were useful protocols to use and thus that someone building a client or server had to interoperate with the existing implementations.

The following characteristics of protocol development during that period are noteworthy.

- o Most client implementations were implemented on very similar systems, in terms of the API's supported and many specifics of the local filesystems exported by servers. In particular, clients exposed filesystem functionality to applications via a standards-compliant API (POSIX).
- o Often, the important client and server implementations were done by the same organization.

As a result of these commonalities, specifications tended to avoid a lot of detail that would have been required in a more diverse environment. New protocol features were thought of in terms of generally understood client and server implementation frameworks and it was generally clear which of those could be implemented without markedly changing that framework.

During this period, there was little perceived need for optional protocol features within NFS, in part because of the commonalities noted above. In some cases in which additional functionality was desired, the facility was added via an optional sideband protocol (e.g. NLM).

##### [4.2.](#) Transition from NFSv2 to NFSv3

There were a number of significant changes involved, but only the

first two were of major importance.

- o Converting file sizes and offsets from 32-bit to 64-bit unsigned integers.
- o Support for uncommitted WRITES and the COMMIT request.
- o Provision for WRITE to return atomic pre- and post-write file attributes, in order to allow a client to determine whether another client was writing the file.

Interestingly enough, this feature was not carried over into NFSv4.

- o The REaddirPLUS request.
- o The addition of NFS3ERR\_JUKEBOX (the precursor of NFS4ERR\_DELAY).

Of these only the first actually needed something as drastic as the XDR replacement model. The others could have been handled simply by adding new RPC requests and an enum value to an existing NFSv2 XDR. Since NFS's extension mechanism was then XDR replacement, such choices were not available.

## [5.](#) Initial Work on NFSv4 Within IETF

Control of the development of NFS was transferred to the IETF in connection with the development of NFSv4. In what follows, we will be distinguishing between "NFSv4" as a family of protocols and "NFSv4.0", the first minor version of NFSv4, also sometimes referred to as "NFSv4".

### [5.1.](#) Transition from NFSv3 to NFSv4.0

NFSv4.0 was the first NFS version published as a Standards track document. Although an initial [[RFC3010](#)], entitled "NFS version 4 protocol" was published as a Proposed Standard, it was never implemented due to issues with the design of locking.

Subsequently, [[RFC3530](#)], entitled "Network File System (NFS) version 4 Protocol" was published as a Proposed Standard, obsoleting

[[RFC3010](#)]. Later, the bis document [[RFC7530](#)] along with the XDR definition [[RFC7531](#)] were published as Proposed Standards.

The set of changes made to create NFSv4.0 was larger by far than that for NFSv3. A partial list follows.

- o Conversion to a stateful protocol, including support for locking. Locking facilities included support for OPENS (with share

Noveck

Expires September 11, 2016

[Page 10]

---

Internet-Draft

nfs-extension

March 2016

reservations), byte-range locking (optionally including mandatory byte-range locks) and delegations.

- o The COMPOUND operation. Although the main motivation for this mechanism was latency reduction, its main role has been to provide the basic framework for XDR extensibility (see [Section 5.2](#)).
- o Conversion to a bi-directional protocol, by the addition of (optional) callbacks.
- o Internationalization.
- o Support for filesystems doing case-insensitive name matching.
- o A new, extensible file attribute model, including support for acls, and conversion of user and group identifiers to a string model, opening the way for multi-domain support.
- o Support for named attributes.
- o Merger of ancillary protocols (e.g. MOUNT, NSM, NLM) into the NFS Protocol proper.
- o Support for crossing of filesystem boundaries within a server's file name space (originally done for the incorporation of MOUNT functionality).
- o Support for such multi-server operations as migration, replication, and referral.

Referrals were not explicitly mentioned in [[RFC3530](#)] and are explained in [[RFC7530](#)].

These features/extensions were implemented via the XDR replacement model. This was the only realistic alternative, not only because of the size of the list, but also because some of the changes undercut some of the central design elements of the pre-IETF NFS protocol.

Note that minor versioning, an important element of NFSv4, is not discussed above. This is partly because minor versioning was not part of NFSv4.0, even though it was discussed in [[RFC3530](#)]. Minor versioning only became an NFSv4 feature during the development of NFSv4.1, as discussed in [Section 6.1](#). We will discuss initial plans for minor versioning in [Sections 5.3](#) and [5.4](#).

## [5.2](#). NFSv4 and XDR Extensibility

Two of the elements within NFSv4.0 have had an important role in allowing NFSv4 to be modified using an XDR extension approach, which was initially used to implement minor versioning.

- o Since COMPOUND was the only RPC procedure, that aspect of NFSv4 never had to change. The COMPOUND request is a variable-length array, with each element being a switched union selected by an operation code. The case of COMPOUND results is similar.

New request operations could be added by extending the two switched unions, one for operations and one for results.

Because of the way XDR works in many implementations, it is often not possible to obtain a specific error code in response to an undefined operation code. Instead, the request containing this operation might be reported as undecipherable. To deal with this, clients would have to test for server awareness of particular protocol features before attempting to use the feature element in question.

- o The new attributes model, unlike COMPOUND, was originally designed to enable protocol extensibility. Sets of attributes are specified as XDR variable-length arrays containing attribute bit

masks and sets of attributes by nominally opaque arrays.

New attributes can be added by defining the numeric constant identifying the added attribute and specifying the XDR type of new attribute.

Because attribute sets are defined as opaque arrays, this has no effect on the code generated by XDR. Decoding of the collection of attributes within the nominally opaque array specifying a set of attributes is not part the task of the code corresponding to the XDR specification of NFSv4 protocol.

A number of other elements of the protocol were subject to extension:

- o Bits within flag words could be extended by defining the relevant constants. This does not result in any change in the code generated by XDR.
- o Enumerations, such as error codes, may be extended by added the new enumeration value to the existing XDR code. Unless the XDR code actually refers to the new values, this does not result in any change in the code generated by XDR.

- o The issues for addition of new cases to existing switched unions are similar to those for addition of operations to COMPOUND, as described above. Just as in that case the requester (i.e. client) needs to test to make sure the responder (i.e. server) can properly interpret the extended union before attempting to send it.

Issues regarding additional callback operations are similar to those for request operations. CB\_COMPOUND takes the place of COMPOUND and the client and server switch roles. The server is the requester and the client is the responder

In this document, we refer to the individual extensions listed above as "feature elements". For some previous documents defining minor versioning rules (e.g. [[RFC5661](#)] and [[NFSv42](#)]), it is unclear whether the rules are referring to individual feature elements or a set of such elements. Our use of the term "feature elements" is desirable for clarity because of the uncertainty just mentioned and

because ([\[NFSv4-vers\]](#)) uses the word "feature" in a different sense (i.e. to denote a set of feature elements which are documented together).

### [5.3.](#) Initial Minor Versioning Model for NFSv4

The minor versioning approach for NFSv4 uses an XDR extension model. It was presented within a versioning paradigm but the fact that all the added features were to be (at least initially) optional indicated that features were intended to be built independently, and that clients were expected to deal with their presence or absence. Note that the term "features" is not explicitly defined. We assume that the definition includes operations within COMPOUND or CB\_COMPOUND, attributes, added flag bits and enum values, and new cases of XDR switch definitions. As noted above, we refer to such items as "feature elements", even though the rules we are discussing may be using the term "features", and there has been a lack of clarity as to what precisely is meant (See [Section 6.1.](#))

Now let's look at some specifics of the minor version rules established for NFSv4 in [\[RFC3530\]](#). Note that some of these were significantly modified by [\[RFC5661\]](#) and [\[NFSv42\]](#), as discussed in [Section 6.4.](#)

- o No RPC requests may be added. Thus COMPOUND (and NULL) are to be the only requests within all NFSv4 minor versions.

Similarly for callbacks, CB\_COMPOUND and CB\_NULL are the only requests within the callback program.

- o The arguments for COMPOUND and CB\_COMPOUND contain a 32-bit minorversion field.

Although this field is part of the minor versioning paradigm, it is not clear how useful it is, as long as all extensions are optional. For a more detailed discussion of this issue, see [Section 7.2.](#)

- o Features may be defined as optional, recommended, or required.

Later, these designations were converted to use the corresponding

upper-case terms defined in [[RFC2119](#)]. See Sections [6.1](#) and [9.2](#) for details.

These designations apply to implementation by the server. For clients, no operations are defined as required, although it is hard to imagine an NFSv4.0 client that does not use PUTFH or SETCLIENTID, for example.

- o Features may be upgraded or downgraded along the optional/recommended/required scale.
- o Features may be declared "mandatory to not implement". This allows the deletion of a feature while retaining as reserved the value previously assigned.
- o Clients and servers that support a particular minor version must support all previous minor versions as well.
- o New features may not be designated as required in the minor version in which they are introduced.
- o Clients are not allowed to use stateids, filehandles, or similar returned objects from the COMPOUND procedure with one minor within another COMPOUND procedure with a different value of the minorversion field.

This model was subsequently modified in [[RFC5661](#)] and in [[NFSv42](#)]. See Sections [6.1](#) and [6.2](#) for details.

Many of the events anticipated in the model presented above have never been realized and it is likely that they never will be realized. See [Section 6.3](#) for some details. Examples are:

- o There have never been optional attributes.
- o Features have never been upgraded or downgraded in the transition between minor versions.

#### [5.4.](#) Goals of Minor Versioning for NFSv4

If we try to understand why the minor versioning model was adopted we can first look at [[RFC3530](#)], which has a number of relevant



statements.

Protocol extensibility (as opposed to versioning) is mentioned early in the RFC. In a short bulleted list of protocol goals, "Designed for protocol extensions" appears together with the following text:

The protocol is designed to accept standard extensions that do not compromise backward compatibility.

Later, the following text appears at the start of the section "Minor Versioning".

To address the requirement of an NFS protocol that can evolve as the need arises, the NFS version 4 protocol contains the rules and framework to allow for future minor changes or versioning.

The document does not explain why the goal of extensibility was constrained by the subsequent establishment of a strict versioning model. It is probably the case that the implications of this constraint were not really examined, and that the shift from major to minor versioning seemed to the working group like a sufficiently radical change to address foreseeable change management issues. It may also be the case that the fact that NFSv4 was moving outside the existing framework for RRC/XDR versioning made additional conceptual change difficult to consider.

Together with the rules that follow, which define an XDR extension model, we can draw the following conclusions.

- o That the use of XDR replacement (presumably to implement "major" versioning) was felt to be not helpful in the current circumstances and that a lighter-weight alternative was desirable.
- o That clients were, as a general rule, expected to deal with the presence or absence of particular extensions

If we look at how the various feature statuses are used, we have a basis to understand how the model was intended to work

- o Rule #12, in saying that features could not be mandatory at initial introduction, states that this rule "forces the use of implementation experience before designating a feature as mandatory." One can conclude from this that it was anticipated that many features might be introduced without implementation

experience and that features designated "optional" might be better thought of as experimental.

- o If it was expected that features might be introduced without implementation experience, there would need to be some way to correct those flaws that were found after introduction. As the restriction on changes to existing XDR structures would limit the ability of new minor versions to correct those flaws, it appears that the provision for declaring features mandatory to not implement was necessary not only to delete obsolete features, but also as a way to correct flawed features by replacing an existing feature by a similar one, with appropriate corrections.

Given the above it is hard to escape the conclusion that the ability to fix protocol bugs, in addition to the adding of entirely new features was intended, at least implicitly, to be part of the NFSv4 minor versioning model. Given that the lines between fixing bugs, correcting feature deficiencies, enhancing features, and providing new features are hard to draw in a precise way, there would have been no way to proceed on any other basis.

Despite the above, as things developed, issues explicitly involving protocol bug correction were not discussed during the period in which minor versions were being constructed. The issue of using NFSv4's extension model to correct protocol bugs was next addressed by [\[NFSv4-vers\]](#) as discussed in [Section 9](#).

## [6](#). Use of Minor Versioning in NFSv4

### [6.1](#). Transition from NFSv4.0 to NFSv4.1

NFSv4.1 was described in [\[RFC5661\]](#) and [\[RFC5662\]](#), each of which was published as a Proposed Standard.

NFSv4.1 made a major change to NFSv4.0. It was able to do so primarily using an XDR extension model although it did not follow the rules laid out in [Section 5.3](#). Instead, [\[RFC5661\]](#) presented its own set of minor versioning rules.

The following major changes in the rules were made.

- o Uses of the terms "recommended", "required", "should", etc. were switched to use the corresponding upper-case words defined in [\[RFC2119\]](#). This included conversion of "recommended" attributes to be "RECOMMENDED". The reasons for this change remain unclear.

Internet-Draft

nfs-extension

March 2016

Unlike the changes noted below, this change was incorporated in [\[RFC7530\]](#). For a discussion of how this situation was dealt with during the IESG review of that document, see [Section 9.2](#)

- o The rule requiring that features be introduced initially as optional was modified to enable features described as "infrastructural" to be required upon introduction.
- o Also, the requirement that clients and servers support previous minor versions changed from a "must" to a "SHOULD". Presumably, this change reflects the fact that a minor version with substantial infrastructural changes is essentially a new protocol, making the "must" seem dubious. Whether the "SHOULD" here is in line with [\[RFC2119\]](#) needs to be explored.

NFSv4.1 also made a change that affected the interpretation of the minor versioning rules, apart from any text changes to those rules. In [\[RFC3530\]](#), the term "feature" is not defined, leading one to conclude that it denotes what we have called "feature elements." By publishing a table showing the status of various operations and callbacks and their relationship to various specific features, [\[RFC5661\]](#) opened the way to a different interpretation. However, this shift was incomplete, leaving room for continued ambiguity since:

- o Only a small set of features are listed, leaving such operations as OPENATTR as orphan feature elements. They are listed as "OPTIONAL", in which case their status of optional features implies that (some) feature elements are features as well.
- o Many operations are listed as "REQUIRED" with no assigned feature named. This poses no immediate problem since most of these are sessions-related. However, given the provision that features can be downgraded, it is not clear what is being referred to here.
- o Feature elements such as attributes are not listed, although they clearly have a status as "OPTIONAL" or "RECOMMENDED", and in some cases a relationship to a broader feature.

NFSv4.1 also bypassed the versioning rules by making non-XDR-based

protocol changes. See [Section 7.1](#) for a discussion of the logic behind such changes.

A large set of changes was associated with the following two structural modifications in the protocol. Each of these involved multiple XDR-based feature elements and some non-XDR-based semantic change.

Noveck

Expires September 11, 2016

[Page 17]

---

Internet-Draft

nfs-extension

March 2016

- o Support for a sessions model including support for EOS (exactly-once semantics).
- o A new set of operations were added which enable the client and server to identify themselves to one another. Although these were implemented to support the sessions model and are often thought of as part of the sessions model, they are best thought of as logically distinct.

The session model involved the following set of protocol changes:

- o The new operation SEQUENCE and CB\_SEQUENCE were defined to allow per-request session-related information to be specified while avoiding changes to existing XDR structures.

SEQUENCE is specified as required in [[RFC5661](#)], while CB\_SEQUENCE is specified as optional.

- o The new callback operation CB\_RECALL\_SLOT was introduced as required, although servers are allowed to escape the requirement by not creating a callback channel.

Since each use of CB\_RECALL\_SLOT requires an initial CB\_SEQUENCE in the CB\_CPMPOUND, it seems that it is an error to designate CB\_RECALL\_SLOT as required while CB\_SEQUENCE is optional.

- o Many semantic changes were made to existing operations to support this arrangement. Since SEQUENCE was supposed to be the first operation of each request, the semantics of each request was modified to make it invalid to specify that operation if SEQUENCE had not appeared first.
- o The semantics of each of the operations which previously had a

clientid in their arguments was modified since the associated clientid was now inferable from the session on which the associated request was issued.

- o The semantics of each operation which used owner-based seqids was modified since these seqids were no longer required to support replay detection.
- o Because of the deletion of owner-based replay detection, OPEN\_CONFIRM was not needed and it became mandatory to not implement.
- o Also, related to the deletion of owner-based replay detection was the fact that RELEASE\_LOCKOWNER became mandatory to not implement. Because lockowners no longer held replay detection state, servers

Noveck

Expires September 11, 2016

[Page 18]

---

Internet-Draft

nfs-extension

March 2016

were able to delete them at will, eliminating the need for the client to be involved in deciding when it was valid to release them.

- o Stateids became tied to the specific session on which they were created, and from the session to the particular client with which they were associated.
- o RENEW became mandatory to not implement, since every request made on a specific session had the effect of renewing the lease of the associated client.

As a result of these changes, no COMPOUND request that contains any operation which is valid in NFSv4.0 is also valid in NFSv4.1. Either it contains an operation which has become mandatory to not implement, or it contains an operation which requires a preceding SEQUENCE operation, which did not exist in NFSv4.0. The only COMPOUND valid in both protocols is one which consists a zero-length operation array.

As mentioned previously, the sequence of operations by which clients and servers connected to one another was expanded and reorganized. This had a major role in enabling a number of functional enhancements to the protocol.

- o Providing for the creation and management of sessions, in

connection with implementing exactly-once semantics, as discussed above.

- o Proving a way for the server to identify himself to the client, allowing the client to determine and use appropriate forms of trunking in clustered-server situations.
- o Restructuring the methods by which connections are associated to clients, allowing callbacks to be assigned to connection separate from that used in the forward direction.

The above set of changes involved multiple operations.

- o EXCHANGE\_ID was introduced as required, replacing SETCLIENTID which became mandatory to not implement.
- o CREATE\_SESSION was introduced as required. Since it had the role of confirming the original EXCHANGE\_ID, it effectively replaced SETCLIENTID\_CONFIRM which became mandatory to not implement.
- o DESTROY\_CLIENTID and DESTROY\_SESSION were introduced as required, providing cleanup facilities missing from NFSv4.0

- o BIND\_CONN\_TO\_SESSION and BACKCHANNEL\_CTL were introduced as required, in order to regularize the management of bindings between connections and sessions.

The following additional feature elements were added as required at initial introduction:

- o TEST\_STATEID and FREE\_STATEID were added to allow better management of lock revocation and lost-lease situations.
- o RECLAIM\_COMPLETE was added to allow better server sequencing of lock reclaim operations.
- o suppattr\_exclcreat was added as a REQUIRED attribute to give clients information about attributes that might be validly specified as part of an exclusive create.

Given the above list, one is forced to the conclusion that these feature elements were not included as required at initial

introduction because there was anything particularly "infrastructural" about them. Rather, there were various feature elements within NFSv4.1 that it was convenient to make required at initial introduction. As a result, their presumed infrastructural character arises from their inclusion as required protocol elements. See [Section 9.4](#) for a discussion of how such feature elements might relate to the proposed reformulation of NFSv4 version management.

The addition of such feature elements did not give rise to any difficulties despite the fact that the rules makes their inclusion problematic. Rather, as discussed below, the problems that NFSv4.1 created for the NFSv4 versioning model arose from the features that were most clearly of an infrastructural character.

In addition to these required feature elements, there were a number of non-XDR-based changes made in NFSv4.1. Although not thought of as "features" at the time, they are described in [[RFC5661](#)], which gives no indication that support is optional. These include:

- o Addition of a new special stateid to represent the current stateid.
- o New rules for the interpretation of a stateid seqid of zero.
- o New rules regarding the interaction of the MODE and acl-related attributes.

There also a number of non-required feature elements. While such feature elements are optional in the sense that a server may or may

not support them, it is not clear that all are optional in terms of the existing minor versioning rules. Given that all attributes are specified as REQUIRED OR RECOMMENDED, it may be that attributes that may or may not be supported are considered recommended from the viewpoint of the minor versioning rules. Here and in the future we will use "non-required" instead of "optional or recommended".

The following non-required feature elements were added.

- o Feature elements to support Parallel NFS
- o WANT\_DELEGATION to allow delegations to be obtained apart from

opens.

- o SECINFO\_NO\_NAME was introduced as "RECOMMENDED", although it is "REQUIRED" if the pNFS file layout type is supported.
- o Feature elements to support directory delegations and notifications.
- o The MODE\_SET\_MASKED, SACL, DACL attributes.
- o The FS\_LOCATIONS\_INFO and FS\_STATUS attributes.

Note that there has been little implementation work so far on the last three of these items.

Parallel NFS created an alternate protocol extension mechanism for NFS. New pNFS mapping types could be added, without any change to the existing XDR or change in the minor version number. Existing mapping types might have their own extension mechanisms. There also exists the possibility that features might be added within the NFSv4 protocol proper, designed to, or capable of, interacting with particular mapping types. This document will not these issues but they will have to be addressed by the NFSv4 Versioning RFC.

The set of changes introduced by NFSv4.1 was very large, and essentially made NFSv4.1 a different protocol from NFSV4.0, albeit one accomplished using the XDR extension model, and following the modified minor versioning rules in [[RFC5661](#)]

As a result of this bifurcation, the incremental enhancement provided by minor versioning became unavailable to NFSv4.0, while it still was available to NFSv4.1. While there was some discussion in the working group regarding the use of "micro-versioning" to address this gap, it was not followed up on and work to reform NFSv4 version management followed a different path.

While there was a general sense within the working group that versions as large as NFSv4.1 should be avoided in the future, there was no effort to modify the versioning rules to make this less likely.



## [6.2.](#) Transition from NFSv4.1 to NFSv4.2

While NFSv4.2 has not been defined in an RFC, the associated specifications have been approved by the IESG and are being prepared for publication. It is thus highly unlikely to experience additions to or deletions from its feature set before publication as a Proposed Standard. [\[NFSv42\]](#) and [\[NFSv42-dotx\]](#) can serve as useful references for this analysis.

Because of the lengthy process involved in producing NFSv4.1, the working group decided that NFSv4.2 was to be a relatively small minor version consisting only of optional features which would be documented by specifying changes from NFSv4.1.

The following features (all optional) are provided for in NFSv4.2:

- o Support for labeled NFS.
- o Server-side copy features, including intra-server copy, inter-server copy, and clone (a variant of intra-server copy).
- o An operation fence option on EXCHANGE\_ID.
- o Application data holes (formerly application data blocks).
- o Disk-space reservation (nominally "recommended" since it is implemented by an attribute and attributes have never been declared "optional").
- o Hole-punching operations.
- o READ\_PLUS.

Note that there are two pieces of infrastructure that are used by multiple features above. These are not "infrastructural" in the sense mentioned in [Section 6.1](#) (i.e. they are not required), but they do serve an infrastructural role in that are required to be present if one of the optional features that use them are supported.

- o WRITE\_PLUS used to implement (ordinary) hole punching and application data holes.

- o OFFLOAD operations/callbacks used to support WRITE\_PLUS and server-side copy.

### 6.3. Evolution of Minor Versioning Model within NFSv4

As noted above, there have been changes made by [[RFC5661](#)] and [[NFSv42](#)] in the NFSV4 minor versioning model.

- o NFSv4.1 (in [[RFC5661](#)]) introduced the concept of "infrastructural" feature elements (i.e. those allowed to be required at initial introduction).
- o NFSV4.1 made additional non-XDR-based changes, which, although not explicitly defined as such, were effectively required.

Taking note of these changes, we can classify potential minor versions, starting with those that currently exist, based on how they affect inter-version compatibility.

- o Minor version zero which introduced a new (major) version of the NFS protocol. All of the operations within it are new and a subset are effectively required.
- o Minor versions which make required changes in the protocol that affect all implementations of the previous minor version.

Such changes may include addition of required/infrastructural feature elements, incorporation of an effectively required non-XDR-based change, or the deletion of a required feature element by making it mandatory to not implement.

Currently, the only such version is minor version one, although it is possible that there could be others in the future.

- o Minor versions that only add optional/recommended feature elements, each present or absent on the server with clients needing to be able to deal individually with their presence or absence.

Currently, the only such version is minor version two. It is likely there will be others in the future and it is possible that all future minor versions will be of this character.

- o Minor versions which make required changes in the protocol that will affect some implementations of the previous minor version.

These can result from feature element status changes. Changing a non-required feature element to required affects only

Internet-Draft

nfs-extension

March 2016

implementations that do not support the feature element. Conversely, deleting a non-required feature element by making it mandatory to not implement affects only implementations that do support the feature element.

No such minor versions currently exist.

#### [6.4.](#) Review of NFSv4 Versioning through NFSv4.2

To summarize protocol extension as it has been applied to the NFSv4 protocols:

- o NFSV4.0 was implemented using the XDR replacement approach inherited from NFSv2 and NFSv3. As was to be expected given the nature and scope of the changes, its development took considerable time.

It defined a protocol extension approach based on the XDR extension mechanism which specified in framework built around the concept of minor versions. However, this mechanism was not used as part of the implementation of NFSv4.0.

- o NFSV4.1 was primarily implemented using the XDR extension mechanism. To implement sessions, it was forced to modify the extension approach in the only way that seemed viable in the circumstances. As a result, the specification process took a long time, since it made significant structural changes to the protocol and also because it specified the entire protocol in a new RFC, rather than documenting a set of extensions.

NFSv4.1 also made a number of modifications to the NFSv4 protocol which did not involve any XDR-based changes at all. These are discussed in [Section 7.1](#). While not considered infrastructural, or even thought of as "features" at the time, these changes are effectively required and the possibility of such changes needs to be considered as the working group formulates its approach to the problems of NFSv4 version management.

- o NFSV4.2 returned to the original XDR extension mechanism and was intended to be a small incremental update with a one-hundred page (or less) specification. The fact that this turned out to be a multi-year effort has occasioned concern and we will discuss how

the process can be streamlined and otherwise made more effective.

The history of NFSv4.2 development serves to illustrate some of the inherent problems in the then-current approach to minor versioning. Since similar problems can be expected to recur

unless that approach is changed, attention needs to be paid to understanding why such difficulties were experienced.

It is important to note that NFSv4.0 (in [[RFC3530](#)] and [[RFC7530](#)]), both about 300 pages and NFSv4.1 (in [[RFC5661](#)]), about 600 pages documented the entire minor version. On the other hand, the NFSv4.2 specification document (in [[NFSv42](#)]) simply specified the changes from the previous minor version, in about 100 pages.

Given the difficulties of writing very large specifications, it has to be considered extremely unlikely that any future minor versions will be documented other than as a set of changes from the previous minor version.

## [7.](#) Inherited NFSv4 Versioning Approach

### [7.1.](#) Non-XDR-based Changes

Although not recognized at the time, the existence of non-XDR-based protocol changes is part of the existing NFSv4 versioning approach and must be addressed in any revision of NFSv4 version management.

Such changes have been of two types:

- o Changes in field interpretation and use. Although the intention has always been that all such matters were to be defined in XDR, there are areas where this is not true. The interpretation of certain opaque fields and of strings are cases in which the field interpretation and use is defined in protocol specification text.
- o Changes in operation semantics. These may apply to only a few operations or to most or all operations.

All such changes have been implemented as required with implementations using the minor version field of the COMPOUND to

determine whether the change applies to a particular request.

## [7.2.](#) The Role of Minor Version Number in NFSv4

Minor versions which may affect inter-version compatibility form an ascending sequence in which we also have a versioning paradigm, implemented principally using XDR extension.

Optional-feature-only minor versions are fundamentally different. Each NFSv4.2 server implements the same protocol as NFSv4.1 with a particular set of optional or recommended feature elements beyond those that are required. This set may range from the null set all the way to all of the non-required feature elements. Here, it

appears that the versioning paradigm is not appropriate to the reality of the extension mechanism.

As a way of illustrating the basic point, let us consider two servers each of which only supports operations within NFSv4.1:

- o The first server "supports" NFSv4.2 but none of the non-required feature elements added in [[NFSv4.2](#)]. In this case, any attempt by a client to use one of those features will result in an NFS4ERR\_OPNOTSUPP being returned.
- o Let us say that the second server does not support NFSv4.2 and supports precisely the same set of feature elements. In this case, a request will be rejected (with error NFS4ERR\_MINOR\_VERSION\_MISMATCH) if its COMPOUND minorversion field is two and if the field is one, any unsupported NFSv4.2 operation will be rejected with NFS4ERR\_OP\_ILLEGAL.

Although this obeys the rules as they stand, there is no obvious value for the client, the server, or the protocol in making these artificial distinctions. Optional-feature-only minor versions such as NFSv4.2 are not minor versions in the same sense that NFSv4.0 and NFSv4.1 are. In this case the minorversion field is not providing much useful information, while the set of operations supported is the important thing that the server implementer chooses and that the client needs to know.

The role of the minorversion field needs to be better understood.

This is particularly so in light of the fact that it appears to be that the original intention was that all or most minor versions would be optional-feature-only minor versions, where, as we have seen, it has no useful role.

While this field is useful in distinguishing NFsv4.0 from NFSv4.1, such transitions were never anticipated when the NFSv4 minor versioning model was first created. A plausible hypothesis is that the switch from "extension" to "versioning" led to a perceived requirement for a version number without much thought about whether it was needed and for what purpose.

Nevertheless this field has proved useful despite the weaknesses noted above. It appears that we have the following ironic situation:

- o The most likely original motivation, to get agreement on a common XDR, appears to be unnecessary because the XDR extension approach implies that a common XDR can be determined if different XDRs are extensions of a common base XDR.

Noveck

Expires September 11, 2016

[Page 26]

---

Internet-Draft

nfs-extension

March 2016

- o The actual uses for this field involve changes in feature statuses, which have only happened in ways that were strongly discouraged by the original definition of minor versioning, and non-XDR-based changes (see [Section 7.1](#)), which were never seriously considered as being related to versioning.

### [7.3](#). Inherited NFS Versioning Practices

The following pattern was followed for NFSv4.2, and, unless a new version management approach is adopted, seems likely to persist.

- o Various features are sketched out in individual drafts.
- o The working group reaches by rough consensus as to the extensions/features to be included in the minor version. At the time consensus is reached, the features may vary as to their maturity. Some have individual drafts which sketch them out while others do not.
- o Any existing individual drafts are combined into a draft of a working group document intended to eventually evolve into an RFC

describing the new minor version. These are then supplemented with descriptions of the other features chosen for inclusion.

- o This document goes through further refinement and cycles of working group document review. At some point a companion -dot-x document is prepared and reviewed by the working group as well.
- o The two documents go through working group last call, IESG review, and RFC publication.

This pattern of development is not a good fit for the kind of minor version that NFSv4.2 is and many future such minor versions will be. Such versions consist of a set of mostly unrelated features, each individually selectable or not by implementers, artificially yoked together. In essence, we have a "feature batch" rather than a minor version.

#### [7.4.](#) Problems with Inherited NFS Versioning Approach

A number of issues have been noted with the existing process for NFSv4.2, leading to the conclusion that the process needs to be revised in some way, for future minor versions, of the same sort.

- o It takes too long to get a minor version drafted and through working group and IESG review. Despite the fact that NFSv4.2 was intended to be a fairly minimal minor version, describable in a one-hundred-page spec, it looks like the pace of development is

such that there will be nearly a five-year delay from the time that the first NFSv4.2 draft was submitted to the time at which the corresponding RFCs are published.

- o The timeline for some of the features within NFSv4.2 is even longer. It will have taken nearly seven years for the inter-server copy feature to go from initial draft to RFC publication of the minor version of which it is a part.
- o Only quite late in the development of the version have there been significant active implementations in which proposed last-minute protocol changes could be tested for validity. As an example of the problem, consider the decision to pass source stateids to the COPY op. If there had been prototype implementations of inter-

server copy, the problems that this created (since stateids are tied to clientids in NFSv4.1 and beyond) would have quickly become manifest.

- o Many features within NFSv4.2 had not received the kind of searching review appropriate to later stages of specification development, until very late in the process.

Some instances of problems/issues ascribable to a lack of searching document review at an early stage are described below. Rather than requiring the necessary review prior to feature acceptance, a common pattern has been that important issues are only discovered on those occasions in which it appears that work on the minor version is coming to a close and that there are issues that have to be addressed before they create difficulties for prospective implementers.

- o The state of the IO hints feature is most unsatisfactory. It is not clear how, or even if, it is possible to specify this in a way that interoperable clients and servers can be written which respond appropriately to such hints so that useful performance improvements can be demonstrated.
- o It was the general understanding within the group that labeled NFS required use of RPCSEC\_GSSv3, when in fact, only one case of labelled NFS, the server-enforced one, required it. When it became clear that RPCSEC\_GSSv3 had not been worked on for a long time, the working group had to address that issue seriously.
- o The security for inter-server copy was specified to be dependent on RPCSEC\_GSSv3, yet, when it was found that RPCSEC\_GSSv3 seemed not to be on the horizon, it turned out that a simple alternative was available. After a great deal of uncertainty about whether the functionality needed for inter-server copy would really be

doable in RPCSEC\_GSSv3, the working group had a range of choices for inter-server copy security.

Later, after much work was done on RPCSEC\_GSSv3, the working group had the option of going back to an approach dependent on RPCSEC\_GSSv3.



- o Application data blocks and related features have had a complicated history within NFSv4.2. Application data blocks were superseded by application data holes. There was little interest in implementing the feature since, as specified, a server implementation would require extensive filesystem extensions. Also, client-side API's were lacking, meaning that the only possible client-side implementations would be in special-purpose clients tightly bound to a particular implementation. Nevertheless, the feature continued in this unsatisfactory state for a long while.

Eventually, it became clear that, as the feature was defined, it imposed significant implementation constraints even on NFSv4.2 clients not implementing the feature. At this point, it became clear that significant changes had to be made.

This issue was resolved by limiting the scope of the proposed feature so that servers were not required to remember the parameters relevant to application data holes, but only the data that they represent.

If we look at the problems above, we can understand better how such problems can arise. In short, the decision as to what features to include within a minor version was not a good use of the rough consensus model. In proceeding on that basis, the group created a set of perverse incentives that undercut the process. Also, as the process goes on for a long time, as is likely, these perverse incentives are intensified. Consider the following points:

- o It is not clear exactly what the consensus as to proposed minor version contents actually means. Working group members might interpret it as meaning "These features are worth pursuing and they should be pursued". However, if they thought the definition was more like, "each of these features is so important that, if it is not ready, any other feature, including the one I'm interested in, should be delayed also", then it is hard to imagine any such rough consensus existing. Note that, given the minor versioning implementation laid out in [Section 7.3](#), the latter definition is, for functional purposes, the effective meaning, of the minor version content consensus.

- o Given that many features are linked together, any delay in one feature, once it is accepted as part of the feature batch, delays all of the features, making it hard for people to comment forthrightly on any significant specification inadequacies. Not only will it delay your preferred feature, but if the problems are not fixed, the only recourse is an extreme penalty. As a result, it often seems not worth pursuing these sorts of issues.
- o As the version turnaround cycle is so long, it is very difficult to remove a feature from a minor version feature batch. Given that these are all features that have enough interest to be in the minor version, it is hard to transfer the feature into the next minor version, given that doing so will certainly result in a multi-year delay, at a minimum.

As a result, features, once accepted, have an implicit guarantee of inclusion in the minor version, undercutting the motivation of the proposer and others to work to move the feature forward.

On the other hand, uncertainty about the time of specification completion often makes it hard to plan for and allocate resources to development of client and server prototypes and implementations.

- o Given that responsibility for a minor version is transferred to the editor of the minor version definition document at an early stage, the process is such that it is not clear who has responsibility to follow up on the work necessary to make the feature happen. Although formally this is the responsibility of the minor version editor, he may not have the required time and expertise in all cases. The lack of designated feature owner makes it hard to follow up on things that require follow-up to progress at an appropriate pace.

## 8. Formulating a New NFSv4 Extension Approach

The following issues led the working group to consider formulation of a new approach to NFSv4 versioning:

- o The experience of NFSv4.2 showed that, although there was a need for shorter documents, addressing that need without other changes in how things were done left important issues unaddressed.
- o The lack of implementation experience for many features led to a general feeling that the working group needed to do more to encourage/require development of feature implementations before they were published as Proposed Standards.

Internet-Draft

nfs-extension

March 2016

- o The publication of multiple minor versioning rules came to seem at variance with the idea of a rule-based approach. Eventually, it was decided that a single version management document was needed for NFSv4 as a whole.

## [9.](#) Current Versioning-related Work

### [9.1.](#) Creation of New NFSv4 Version Management Document

The working group is now working on a standards-track document ([\[NFSv4-vers\]](#)) defining an approach to version management that is intended to apply to NFSv4 as a whole. The major advances that formed the basis for that new document include

- o Creation of a set of minor versioning rules to form a basis for a unified set of versioning rules. Such a set of rules would supersede the per-minor-version rules that had previously been present.
- o Creation of the concept of extensible minor version with an associated workflow that avoids many of the problems noted above with regard to the batching of features.
- o Explicitly allowing XDR changes to correct protocol errors. This addresses a situation in which there was sufficient uncertainty about such changes to prevent them from being considered, even though they were not explicitly disallowed.

### [9.2.](#) Related Changes to Other Documents

During this period, changes were made to some related documents, as they moved toward publication.

Some of these changes were made to simplify handling of the transition in versioning models that would occur upon publication of [\[NFSv4-vers\]](#) as an RFC.

- o During the IESG review process, drafts of [\[RFC7530\]](#) were modified to eliminate specific minor versioning rules. As a result these rules did not appear in [\[RFC7530\]](#)

This made sense because the rules in question did not apply to NFSv4.0, and were, with regard to NFSv4.1 and beyond, superseded

by those in [[RFC5661](#)].

- o The restatement of minor versioning rules in [[NFSv42](#)] was eliminated. In its place was left a short statement of v4.2-relevant exceptions from existing rules. The text is

compatible with the base rules being taken from either [[RFC5661](#)] or [[NFSv4-vers](#)].

As a result of these changes, upon publication of an NFSv4 version management RFC, it will only need to be marked as updating [[RFC5661](#)].

Another relevant change concerned the use of the term "RECOMMENDED" regarding attributes which are not REQUIRED. As it appeared that this usage is inconsistent with [[RFC2119](#)], the draft of [[RFC7530](#)] was modified and [[RFC7530](#)] was published including the following statement:

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 [RFC 2119](#) except where "REQUIRED" and "RECOMMENDED" are used as qualifiers to distinguish classes of attributes as described in [Section 1.3.3.2](#) and [Section 5](#).

A reasonable conclusion to draw from this is that while certain attributes are indeed REQUIRED, the other ones cannot be designated as "RECOMMENDED" as that term is defined in [[RFC2119](#)].

Work still needs to be done to deal with the analogous issue in [[RFC5661](#)].

This change relates to the NFSv4 feature status model. Although there has been discussion of feature elements moving within a status hierarchy including OPTIONAL, RECOMMENDED, and REQUIRED statuses, with the exception of attributes, "RECOMMENDED" has never been used. Given that all feature elements are, in [RFC2219](#) terms, either OPTIONAL or REQUIRED opens the way to a simpler feature status model.

### [9.3](#). Further work on New NFSv4 Versioning Document

Previously the versioning document incorporated two conceptual models

which needed to be better integrated to give us a more coherent treatment of version management within NFSv4.

- o The versioning rules inherited from [[RFC5661](#)] are focused on minor versions and individual XDR changes (in our terms these are "feature elements" although the existing rules refer to them as "features"). Even though the working group expects the addition of incremental features to be at the core of our prospective versioning practice, the rules relate to that approach only by implication.

- o The version extensibility and protocol fix work had focused on (non-required) features, but had no real place for minor versions other than as a passive recipient of those features. It was decided to focus on this aspect of version management since it is likely to be the primary means by which changes will be introduced into NFSv4 in the future. Nevertheless, there was a need to examine the possibility of minor versions that might refactor things or otherwise allow other sorts of minor-version-level change, that are outside the scope of things that can be done via the simple extension model explored so far.

Recent work has restructured the treatment of NFSv4 version management into a layered structure.

- o Changes, including XDR changes and non-XDR changes.
- o Features, which are groups of protocol changes specified together.
- o Minor versions which combine a set of features which may be optional or required.

One way of dealing with the issues surrounding the minor versioning rules is to re-organize them so as to move away from a single set of minor versioning rules, while adapting them to the evolving version management model in [[NFSv4-vers](#)].

Although the result may not have the format of a numbered list of rules, there will be sets of guidelines that serve the needed functions. One possible organization is the following:

- o Rules defining the types of protocol changes that may be made. These would include the XDR extensions mentioned in the existing rules, but there would also have to be provision for at least some of the non-XDR-based changes that have been useful in the past. A large part of the original rules wound up in this group.
- o Rules governing the construction and organization of features. This is a new area.
- o Rules governing the incorporation of features in the protocol, whether this as part of a published minor version, an extension to a published minor version, or as a correction to a published minor version. This is also a new area.
- o Rules governing the changes of feature statuses in a minor version. These are the only rules regarding minor versions that are still directed at minor versions. In the same class are rules

which would deal with possibility of post facto changes of inter-feature compatibility constraints.

- o Rules that deal with the interaction of multiple minor versions, on the model of rules 11 and 13 in [[RFC5661](#)]. These are the only rules directed to implementers, as opposed to those defining new features and minor versions.

#### [9.4.](#) Issues Needing further discussion

While there has been general acceptance of the idea that version management needs to become more flexible and incremental, the details need to be more fully discussed.

One particular area of discussion is to get more clarity on the role of minor versions within the new version management approach. It appears that there is a range of opinions about how often minor versions will be needed. Some people are of the opinion that all required protocol change can be done using the extension model, making an provisions for additional minor versions redundant. Others feel that there may well be situations in which changes which cannot be performed using the existing model will be required.

Fortunately, it is not necessary to get agreement on this issue to proceed with this document. As long as people are in agreement as to the situations that would require a new minor version, the fact that they have different expectations as to whether and when those will occur is not relevant at this point. Those are matters best left for discussion when the relevant situations might arise.

Currently [[NFSv4-vers](#)] defines the following events as requiring a minor version to effect. These need to be discussed to make sure the group has a general understanding of our versioning options going forward.

- o Addition of required features.
- o Changes of features from optional to required or the reverse.
- o Conversion of features to mandatory to not implement.
- o Any non-XDR-based semantic changes.
- o Any change to the status of feature elements within existing features.

- o Any change that affect constraints regarding what existing features may be present together, including feature dependence and compatibility rules.

It might also be useful for the working group to have a discussion of situations in which it might choose to create a new minor version, even if not obliged to do so.

- o To simplify the task of clients in determining what features servers might be aware of.
- o To logically isolate a previous epoch of protocol extension preparatory to moving the protocol in a new direction.

Another issue that the working group might profitably discuss is the

question of versions that make larger sets of protocol changes, on the order of NFSv4.0 or NFSv4.1. While most people feel that such changes are quite unlikely in the foreseeable future, there is likely to be a range of opinions about how a version management RFC should deal with the matter.

- o The text might make it clear that when introducing a feature as required, the scope of related changes that follow from it (i.e. what [[RFC5661](#)] presumably means by its "infrastructural" character) is a reason to make it optional at initial introduction, rather than the reverse.
- o The text might simply ban introducing a feature as required, although some of these would not pose problems if they are small enough, as was the case with some of those introduced in NFSv4.1. (See [Section 6.1](#)).
- o Stating that such changes be made using the XDR replacement model, implicitly indicating that such versions should be part of NFSv5.x, and thus out of scope for an NFSv4 document.

## [10](#). Looking Back

We can summarize the history of protocol extension within the NFS family as protocols as follows:

- o At first, NFS used the XDR replacement paradigm in order to effect protocol extension, following standard practice for XDR-based protocols.
- o With the development of NFSv4.0, the basic mechanisms were in place to adopt an XDR extension paradigm as a means of protocol extension. Despite this, for reasons that are not very clear, a

minor versioning approach was established, even though the basic goal, to enable optional features, was at variance with this.

- o A number of efforts were made to implement this minor versioning approach. While protocol improvements were made, the results were troubling and various ad hoc adjustments were made. For example, when NFSv4.1 broke with the optional-feature-only idea and produced a 900-page spec, it was decided that NFSv4.2 was to be a



small minor version with a spec shorter than 100 pages.

- o When it turned out NFSv4.2, despite being so much smaller, took over four years to go from -00 to IESG consideration (as opposed to three years for NFSv4.1) did it become clear that serious reform was in order.

A question that has been asked is why NFSv4 has had such difficulty arriving at a workable approach to protocol extension, compared to other protocols, that have embraced incremental extension without difficulty. This is a question for which a definitive answer is not possible. Nevertheless, it is a reasonable question, and the author has attempted to answer it.

In part, the problem stems from an early confusion of extensibility and versioning, as we have seen in [Section 5.4](#). While this sheds light on the origin of the problem, in a way it compounds our difficulties. Given that this same confusion first appeared in [\[RFC3010\]](#), it appears that it has taken around fifteen years to provide a version management framework appropriate to the initial goals for NFSv4.

In part, this extended hiatus derives from the history cited above.

- o It is difficult to focus properly on change management in the absence of actual change.
- o Since NFSv4.1 did not really implement the version management approach initially specified for NFSv4, there was no experience to see how well that worked until NFSv4.2 was fairly far along.
- o The unexpected complexity and difficulty of NFSv4.2 may have forced attention on completing that specification, as opposed to investigating how/why those difficulties arose.

Despite the plausibility of the above, the author feels that it doesn't fully explain the length of the gap and thinks that looking for a more systemic explanation is worthwhile. The following suggestions are worth considering as an explanation for NFSv4's difficulties compared to other protocols:

- o When other protocols implement extension mechanisms, they

generally do so in a specialized way. This is opposed to the case of NFSv4 in which a very large part of the protocol is, at least theoretically, subject to change. It would not be surprising if this situation occasioned concern about the possibility of uncontrolled change causing a reduction of effective interoperability.

- o The history of NFS created expectations that numbered versions would be produced. The initial definition of the minor versioning model reinforced those expectations and created institutional barriers that strengthened them. For example, the working group charter would at times mention numbered minor versions and discussed the items that they were expected to contain.
- o Because NFSv4 was an XDR-based protocol, there were further conceptual barriers to change that developed. This was in part because RPC version negotiation is built around the concept of numbered versions but a more fundamental issue derives from the fundamentals of XDR, which, by seeking a full definition of the protocol to compile, seems to foreclose the idea of extensibility by focusing on the need for equivalence/identity between two protocol definitions. As a result partial order defined by the XDR extension relation may have been outside the natural conceptual framework for an XDR-based protocol.

In part, these are problems which derive from the identification of a protocol with the contents of an XDR file. This has had two unfortunate consequences.

- o If the XDR file had not changed, it was assumed that the protocol had not changed. As a result, there has been no awareness of the version management implications of non-XDR-based semantic changes made in NFSv4.1, simply because there was no corresponding change to the XDR.
- o It is often the case that it is felt that a new error cannot be added simply because the error code would be added to an existing enum and thus change the XDR code, which is felt to be inherently dangerous, even though the code generated by XDR would not change at all.

The fact that some of the necessary extensions did affect the code generated by XDR led to further difficulties. While it might seem straightforward to design a mechanism to allow extensions, and NFSv4 in [[RFC3530](#)] defined such a mechanism, the information hiding that is part of the XDR interface could have caused the details of the compatibility issues to be obscured. Instead, the natural tendency

was only to see that the XDR's were different, and thus presumably incompatible. As a result, the virtues of the NFSv4 extensibility infrastructure were difficult to see, and so we were unable to take full advantage of them.

## 11. Looking Forward

Once this document gets through working group last call, there are a number of events that must occur before implementation of a new version management approach can begin.

- o The new version management document [[NFSv4-vers](#)] needs to be approved for publication as a Proposed Standard.
- o The NFSv4.2 documents ([\[NFSv42\]](#) and [\[NFSv42-dotx\]](#)) need to be approved for publication as a Proposed Standards. Since [\[NFSv4-vers\]](#) states that each minor version beyond NFSv4.1 will be extensible by default, these approvals and that of [\[NFSv4-vers\]](#) may happen in any order.
- o Some extension needs to be accepted by the working group as a feature specification document, as described in [\[NFSv4-vers\]](#). A likely candidate for this role is [\[xattr\]](#) which introduces a feature that allows user-specified extended attributes. Another possible candidate feature is discussed below. As work proceeds on that possible feature specifications and on [\[NFSv4-vers\]](#), reviewers would verify that they meet the requirements for a feature specification document.

Another possible extension involves the `mode_umask` attribute described in [\[mode\\_umask\]](#), which is currently an individual draft. This attribute allows ACL inheritance to avoid problems that have arisen in adapting to the POSIX-based `umask` concept. This extension's purpose is to address an oversight in the design of NFSv4's access control attributes, present in all existing minor versions, including NFSv4.0. Because of that fact, it is likely to be turned into a working group item and it might be considered as a protocol correction, possibly updating [\[RFC7530\]](#), after [\[NFSv4-vers\]](#) is published as a Proposed Standard.

Once [\[xattr\]](#) and [\[NFSv4-vers\]](#) are published, the extended attributes feature would become a valid optional extension to NFSv4.2 and clients and server would be able to support it. If [\[mode\\_umask\]](#) and [\[NFSv4-vers\]](#) are published the same thing would happen with the `mode_umask` attribute.

Given that a dramatic change will be involved, it seems that a discussion of risk mitigation is in order. The most important

component of addressing the risks associated with a new process is active concern about problems that arise. This needs to be combined with a willingness to make appropriate adjustments if problems arise. Clearly, we need to avoid a situation in which an extension model is not working and the problems go on for years without being addressed.

The principal means by which the risk of change may be mitigated would involve care in approving extensions. Both the working group and the IESG should exercise the requisite care, to make sure that the extensions are clearly documented, have an appropriate scope, and address real problems. Further, having a clear requirement for interoperable prototype implementations should have the effect of limiting excessive feature creation activity.

## [12.](#) Security Considerations

Since no substantive protocol changes are proposed here, no security considerations apply.

As features and minor versions designed and specified in standards-track documents, their security issues will be addressed and each RFC candidate will receive the appropriate security review from the NFSv4 working group and IESG.

## [13.](#) IANA Considerations

The current document does not require any actions by IANA.

Depending on decisions that the working group makes about how to address the issues raised in this document, future documents may require actions by IANA.

## [14.](#) References

### [14.1.](#) 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>>.

## 14.2. Informative References

[mode-umask]

Fields, J. and A. Gruenbacher, "Allowing inheritable NFSv4 ACLs to override the umask", February 2016, <<http://www.ietf.org/id/draft-bfields-nfsv4-umask-01.txt>>.

Noveck

Expires September 11, 2016

[Page 39]

---

Internet-Draft

nfs-extension

March 2016

Work in progress.

[NFSv4-vers]

Noveck, D., "NFSv4 Version Management", January 2016, <<http://www.ietf.org/id/draft-ietf-nfsv4-versioning-03.txt>>.

Work in progress.

[NFSv42] Haynes, T., Ed., "NFS Version 4 Minor Version 2", January 2016, <<http://www.ietf.org/id/draft-ietf-nfsv4-minorversion2-41.txt>>.

Work in progress.

[NFSv42-dotx]

Haynes, T., Ed., "NFS Version 4 Minor Version 2 Protocol External Data Representation Standard (XDR) Description", January 2016, <<http://www.ietf.org/id/draft-ietf-nfsv4-minorversion2-dot-x-41.txt>>.

Work in progress.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, [RFC 793](#), DOI 10.17487/RFC0793, September 1981, <<http://www.rfc-editor.org/info/rfc793>>.

[RFC1094] Nowicki, B., "NFS: Network File System Protocol specification", [RFC 1094](#), DOI 10.17487/RFC1094, March 1989, <<http://www.rfc-editor.org/info/rfc1094>>.

[RFC1813] Callaghan, B., Pawlowski, B., and P. Staubach, "NFS

Version 3 Protocol Specification", [RFC 1813](#), DOI 10.17487/RFC1813, June 1995, <<http://www.rfc-editor.org/info/rfc1813>>.

- [RFC2780] Bradner, S. and V. Paxson, "IANA Allocation Guidelines For Values In the Internet Protocol and Related Headers", [BCP 37](#), [RFC 2780](#), DOI 10.17487/RFC2780, March 2000, <<http://www.rfc-editor.org/info/rfc2780>>.
- [RFC3010] Shepler, S., Callaghan, B., Robinson, D., Thurlow, R., Beame, C., Eisler, M., and D. Noveck, "NFS version 4 Protocol", [RFC 3010](#), DOI 10.17487/RFC3010, December 2000, <<http://www.rfc-editor.org/info/rfc3010>>.

Noveck

Expires September 11, 2016

[Page 40]

---

Internet-Draft

nfs-extension

March 2016

- [RFC3530] Shepler, S., Callaghan, B., Robinson, D., Thurlow, R., Beame, C., Eisler, M., and D. Noveck, "Network File System (NFS) version 4 Protocol", [RFC 3530](#), DOI 10.17487/RFC3530, April 2003, <<http://www.rfc-editor.org/info/rfc3530>>.
- [RFC5661] Shepler, S., Ed., Eisler, M., Ed., and D. Noveck, Ed., "Network File System (NFS) Version 4 Minor Version 1 Protocol", [RFC 5661](#), DOI 10.17487/RFC5661, January 2010, <<http://www.rfc-editor.org/info/rfc5661>>.
- [RFC5662] Shepler, S., Ed., Eisler, M., Ed., and D. Noveck, Ed., "Network File System (NFS) Version 4 Minor Version 1 External Data Representation Standard (XDR) Description", [RFC 5662](#), DOI 10.17487/RFC5662, January 2010, <<http://www.rfc-editor.org/info/rfc5662>>.
- [RFC7530] Haynes, T., Ed. and D. Noveck, Ed., "Network File System (NFS) Version 4 Protocol", [RFC 7530](#), DOI 10.17487/RFC7530, March 2015, <<http://www.rfc-editor.org/info/rfc7530>>.
- [RFC7531] Haynes, T., Ed. and D. Noveck, Ed., "Network File System (NFS) Version 4 External Data Representation Standard (XDR) Description", [RFC 7531](#), DOI 10.17487/RFC7531, March 2015, <<http://www.rfc-editor.org/info/rfc7531>>.

[xattr] Naik, M., "File System Extended Attributes in NFSv4",  
March 2016,  
<<http://www.ietf.org/id/draft-ietf-nfsv4-xattrs-02.txt>>.

Work in progress.

#### Appendix A. Acknowledgements

The author wishes to thank Chuck Lever of Oracle for his comprehensive document review and his many important suggestions, which helped to significantly improve this document.

#### Author's Address

David Noveck  
Hewlett Packard Enterprise  
165 Dascomb Road  
Andover, MA 01810  
US

Phone: +1 978 474 2011  
Email: [davenoveck@gmail.com](mailto:davenoveck@gmail.com)