NFSv4 D. Noveck
Internet-Draft HPE
Updates: 5661 (if approved) January 15, 2016
Intended status: Standards Track
Expires: July 18, 2016

NFSv4 Version Management
draft-ietf-nfsv4-versioning-03

Abstract

This document describes the management of versioning within the NFSv4 family of protocols. It covers the creation of minor versions, the addition of optional features to existing minor versions, and the correction of flaws in features already published as Proposed Standards. The rules relating to the construction of minor versions and the interaction of minor version implementations that appear in this document supersede the minor versioning rules in RFC5661.

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 July 18, 2016.

Table of Contents

1. Introduction 3
1.1. Existing Minor Versions 4
1.2. Updated NFSv4 Version Management Framework 4
2. Terminology 5
2.1. Use of Keywords Defined in RFC2119 6
2.2. Use of Feature Statuses 7
2.3. NFSv4 Versions 8
3. Consolidation of Version Management Rules 9
4. XDR Considerations 11
4.1. XDR Extension 11
4.1.1. XDR Extension in General 11
4.1.2. Particulars of XDR Extension within NFSv4 12
4.1.3. Rules for XDR Extension within NFSv4 13
4.2. Handling of Protocol Elements 13
4.3. Organization of Protocol Elements 15
4.4. Inter-version Interoperability 15
4.4.1. Requirements for Knowledge of Protocol Elements . . . 15
4.4.2. Establishing Interoperability 16
4.4.3. Determining Knowledge of Protocol Elements 18
4.4.4. Interoperability Between Version Groups 19
5. Other NFSv4 Protocol Changes 19
5.1. Non-XDR Protocol Changes 19
5.1.1. Field Interpretation and Use 20
5.1.2. Behavioral Changes 21
5.1.3. Rules for non-XDR changes 21
5.2. Specification of Associated Protocols 22
5.2.1. Associated Protocols via pNFS Mapping Types 22
5.2.2. Additional Forms of Associated Protocols 23
6. NFSv4 Protocol Features 24
6.1. Previous Uses of the Feature Concept 25
6.2. Rules for Protocol Feature Construction 25
6.3. Statuses of Features 26
6.4. Statuses of Protocol Elements Within Features 27
6.5. Determining Protocol Element Support 29
6.6. Feature Discovery 30
6.7. Feature Incorporation 31
7. Extensions within Minor Versions 32
7.1. Adding Features to Extensible Minor Versions 32
7.2. Use of Feature Specification Documents 32
7.3. Compatibility Issues 33
7.3.1. Compatibility Issues for Messages Sent to Servers . . 34
7.3.2. Compatibility Issues for Messages Sent to Clients . . 35

7.4. Relationship Between Minor Versioning and Extensions within a Minor Version 36
8. Minor Versions 36
8.1. Creation of New Minor Versions 36
8.1.1. New Minor Versions within an Existing Group 37
8.1.2. New Minor Version Groups 37
8.1.3. Limits on Minor Version Groups 40
8.2. Role of Minor Versions 41
8.3. Minor Version Interaction Rules 41
8.3.1. Minor Version Identifier Transfer Issues 42
8.3.2. Minor Version Intra-Group Compatibility 42
8.3.3. Minor Version Inter-Group Compatibility 43
9. Correction of Existing Minor Versions and Features 44
9.1. XDR Changes to Implement Protocol Corrections 45
10. Documentation of Features, Extensions, Minor Versions, and Protocol Corrections 46
10.1. Documentation Approach 47
10.2. Indexing material 47
10.3. Feature Specification Documents 48
10.4. XDR File Considerations 50
10.5. Additional Documents to Support Protocol Extension . . . 51
10.5.1. Minor Version Indexing Document 51
10.5.2. Consolidated XDR Document 52
10.5.3. XDR Assignment Document 52
10.5.4. Transition of Documents to RFC's 53
10.6. Documentation of New Minor Versions 54
10.7. Documentation of XDR Changes for Corrections 54
11. Security Considerations 55
12. IANA Considerations 55
13. References 55
13.1. Normative References 55
13.2. Informative References 55
Appendix A. Acknowledgements 57
Author's Address 57

1. Introduction

To address the requirement for an NFS protocol that can evolve as the need arises, the Network File System (NFS) version 4 (NFSv4) protocol provides a framework to allow for future changes via the creation of new protocol versions including minor versions and certain forms of modification of existing minor versions. The version management rules contained in this document allow extensions and other changes to be implemented in a way that maintains compatibility with existing clients and servers.

1.1. Existing Minor Versions

Previously, all protocol changes had been part of new minor versions. The COMPOUND procedure (see Section 14.2 of [RFC7530]) specifies the minor version being used by the client in making requests. The CB_COMPOUND procedure (see Section 15.2 of [RFC7530]) specifies the minor version being used by the server on callback requests.

Each existing minor version has been specified by one or more standards track RFCs:

- Minor version 0 (NFSv4.0) is specified by [RFC7530] with the XDR description appearing in [RFC7531].

- Minor version 1 (NFSv4.1) is specified by [RFC5661] with the XDR description appearing in [RFC5662].

- Minor version 2 (NFSv4.2) is specified by [NFSv42] (in terms of changes from [RFC5661]). The XDR description appears in [NFSv42-dot-x]

Existing minor versions can be divided into two groups, based on compatibility considerations. NFSv4.0 is one group, while NFSv4.1, NFSv4.2, and potentially other minor versions, form a second group. The definition of NFSv4 minor version groups is explained in more detail in Section 2.3, as is the concept of variants within minor versions and version groups.

1.2. Updated NFSv4 Version Management Framework

A number of significant changes from previous version management practices should be noted here:

- Creation of a new minor version is no longer the only way in which protocol changes may be made. Added optional features and protocol corrections can be proposed, specified and implemented within the context of a single minor version. Creation of new minor versions remains available to make other sorts of changes.

- Specification of future minor versions in the way that was done for NFSv4.0 and NFSv4.1 (i.e. as a single document defining the entire protocol) is no longer practical and should not be attempted. All future minor versions will be documented by specifying the differences between the minor version being documented and the previous minor version. The documentation framework discussed in Section 10 should be used.

After dealing with some preliminary matters, this document focuses on presenting the conceptual framework on which NFSv4 versioning is built.

- First we discuss (in Section 4) how the XDR descriptions for various NFSv4 versions can be extended to produce the XDR descriptions for other versions while allowing clients and servers using the XDR descriptions associated with different versions to communicate.

- We then complete the discussion (in Section 5) of the range of protocol changes that NFSv4 versioning is to deal with.

- Then we discuss (in Section 6) how those changes are organized into features and feature packages.

Using this framework, we look at the ways that those changes can be incorporated into the NFSv4 protocol.

- The addition of new feature packages to existing minor versions is discussed in Section 7.

- New Minor versions can be constructed, as described in Section 8.

- Issues relating to the correction of protocol errors in existing features and minor versions are discussed in Section 9.

We then discuss (in Section 10) how features, minor versions, and protocol corrections will be documented.

2. Terminology

A basic familiarity with NFSv4 terminology is assumed in this document and the reader is pointed to [RFC7530].

In this document, the term "version" is not limited to minor versions. When minor versions are meant, the term "minor version" is used explicitly. For more discussion of this and related terms, see Section 2.3

In this document, the word "feature" is used , except in the case of quotations, to denote a key structuring concept. By organizing changes into features, defining RFCs can clearly specify what protocol elements a server must be able to recognize and what protocol elements a server must support. See Section 6 for more which allows the defining RFCs to clearly specify what protocol elements must be supported together by the server and when a given

server must be able to correctly interpret the corresponding associated protocol constructs. See Section 6 for more details.

A feature contains one or more "feature elements". Often, at least one feature element will be a protocol extension that can help a sender determine whether the receiver supports a given feature. See Section 4.1.3 for more details. A feature element may also be one of a set of other types of protocol change as described in Section 5.

A "feature package" is a set of features that are defined together, either as part of a minor version or as part of the same protocol extension.

We also need to introduce our vocabulary regarding specification of features and minor versions. Given the ongoing shift to a finer-grained documentation model, it is important to be clear here.

- The term "minor version definition document" denotes the principal document defining a specific NFSv4 minor version. It may be in the form of a complete protocol definition (e.g. [RFC7530], [RFC5661]), a specification of changes relative to the previous minor version (e.g. [NFSv42]), or in a document that specifies the features to be included, either by referencing their definition document normatively (see Section 10.6) or implicitly (see Section 7.1).

- The term "minor version documentation" includes the minor version definition document but also includes any corresponding XDR definition documents if they are published separately (e.g. [RFC7531], [RFC5662], [NFSv42-dot-x]). Also included are documents separately specifying features newly incorporated in the minor version and the ancillary documents described in Section 10.5.

- The term "feature definition document" denotes a document describing a single feature or a set of closely related features, forming a feature package.

- The term "protocol definition document" denotes a minor version definition document, a feature definition document or any standards-track document updating one of these.

2.1. Use of Keywords Defined in RFC2119

The keywords defined by [RFC2119] have special meanings which this document intends to adhere to. However, due to the nature of this document and some special circumstances, there are some complexities to take note of:

- Where this document does not directly specify implementation requirements, use of these capitalized terms is often not appropriate, since the guidance given in this document does not directly affect interoperability.

- In this document, what authors of RFCs defining features and minor versions need to do is stated without these specialized terms. Although it is necessary to follow this guidance to provide successful NFSv4 version management, that sort of necessity is not of the sort defined as applicable to the use of the keywords defined in [RFC2119].

  The fact that these capitalized terms are not used should not be interpreted as indicating that this guidance does not need to be followed or is somehow not important.

- In speaking of the possible statuses of features and feature elements, the terms "OPTIONAL" and "REQUIRED" are used. For further discussion, see Section 2.2.

- When one of these upper-case keywords defined in [RFC2119] is used in this document, it is in the context of a rule directed to an implementer of NFSv4 minor versions, the status of a feature or protocol element, or in a quotation, sometimes indirect, from another document.

2.2. Use of Feature Statuses

There has been some confusion, during the history of NFSv4, about the correct use of these terms, and instances in which the keywords defined in [RFC2119] were used in ways that appear to be at variance with the definitions in that document.

- In [RFC3530], the lower-case terms "optional", "recommended", and "required" were used as feature statuses, Later, in [RFC5661] and [RFC7530], the corresponding upper-case keywords were used. However, it is not clear why this change was made.

- In the case of "RECOMMENDED", its use as a feature status is inconsistent with [RFC2119] and it will not be used for this purpose in this document.

- The word "RECOMMENDED" to denote the status of attributes in [RFC3530] and [RFC5661] raises similar issues. This has been recognized in [RFC7530] with regard to NFSV4.0, although the situation with regard to NFSv4.1 remains unresolved.

In this document, the keywords "OPTIONAL" and "REQUIRED" and the phrase "mandatory to not implement" are used to denote the status of features and individual protocol elements within a given minor version. In using these terms, RFCs which specify the status of features or protocol elements inform:

- client implementations whether they need to deal with the absence of support for the protocol elements
- server implementations whether they need to provide support for the protocol elements

When the status of a protocol feature is specified, the support requirements for associated protocol elements are defined by the status of the protocol elements with regard to the feature in question as described in Section 6.4.

The fact that such statuses and the organization of protocol features may change between minor version groups may raise interoperability issues which the authors of minor version RFCs and the working group need to carefully consider. See Section 8.1.2 for guidance in this regard.

2.3. NFSv4 Versions

The term "version" denotes any valid protocol variant constructed according to the rules in this document. It includes minor versions, but there are situations which allow multiple variant versions to be associated with and co-exist within a single minor version:

- When there are feature specification documents published as Proposed Standards extending a given minor version, then the protocol defined by the minor version specification document, when combined with any subset (not necessarily proper) of the feature specification documents, is a valid NFSv4 version variant which is part of the minor version in question.
- When there are protocol corrections published which update a given minor version, each set of published updates, up to the date of publication of the update, is a valid NFSv4 version variant which is part of the minor version in question.

Because of the above, there can be multiple version variants that are part of a given minor version. Two of these are worthy of special terms:

- The term "base minor version" denotes the version variant that corresponds to the minor version as originally defined, including

all protocol elements specified in the minor version definition document but not incorporating any extensions or protocol corrections published subsequently.

- At any given time, the term "current minor version" denotes the minor version variant including all extensions of and corrections to the minor version made by standard-track documents published subsequently.

Each version variant which is part of a given minor version is a subset of the current minor version and a superset of the base minor version. When the term "minor version" is used without either of these qualifiers, it should refer to something which is true of all variants within that minor version. For example, one may refer the set of REQUIRED features in a given minor version since it is the same for all variants within the minor version.

Each client and server which implements a specific minor version will implement some particular variant of that minor version. Each of these will be a superset of the appropriate base minor version.

A minor version group is defined as a set of minor versions having exactly the same set of REQUIRED and mandatory to not implement protocol elements. The union of the sets of variants for all these minor versions provides a high degree of inter-variant compatibility. Clients and servers which implement variants within this group should be compatible as long as each takes proper care, as it should, to properly deal with the case in which the other party does not know of or has no support for OPTIONAL protocol elements.

3. Consolidation of Version Management Rules

In the past, the only existing version management rules were the minor versioning rules that had been being maintained and specified in the Standards Track RFCs which defined the individual minor versions. In the past, these minor versioning rules were modified on an ad hoc basis for each new minor version.

More recently, minor versioning rules were specified in [RFC5661] while modifications to those rules were allowed in subsequent minor versions.

This document defines a set of version management rules, including rules for minor version construction. These rules apply to all future changes to the NFSv4 protocol. The rules are subject to change but any such change should be part of a standards track RFC obsoleting or updating this document.

Rather than a single list of minor versioning rules, as in [RFC5661], this document defines multiple sets of rules that deal with the various forms of versioning provided for in the NFSv4 version management framework.

- The kinds of changes that may be made are addressed in the rules in Sections 4.1.3, 5.1.3, 5.2.1, and 5.2.2.

- Rules relating to the composition of changes into protocol features are addressed in Section 6.2

- Rules limiting the protocol features which may be effected as an extension to an existing minor version appear in Section 7.

- Minor version construction, including rules applicable to protocol features which cannot be used as extensions to existing minor versions are addressed in Sections 8.1.1 and 8.1.2.

- Minor version interaction rules are discussed in Sections 8.3.2, 8.3.3, and 8.3.1.

This document supersedes minor versioning rules appearing in the minor version specification RFC's, including those in [RFC5661]. As a result, potential conflicts among these documents should be addressed as follows:

- The specification of the actual protocols for minor versions previously published as Proposed Standards take precedence over minor versioning rules in either this document or in the minor version specification RFC's. In other words, if the transition from version A to version B violates a minor versioning rule, the version B protocol stays as it is. In particular, many of the changes made for NFSV4.1 would not be allowed in the version management framework defined here. See Section 5.1.3 for details.

- Since minor versioning rules #11 and #13 from [RFC5661] deal with the interactions between multiple minor versions, the situation is more complicated. See Section 8.3 for a discussion of these issues, including how potential conflicts between rules are to be resolved.

- Otherwise, any conflict between the version management rules in this document and those in minor version specification RFC's are to be resolved based on the treatment in this document. In particular, corrections may be made as specified in Section 9 for all previously specified minor versions and the extensibility of previously specified minor versions is to be handled in accord with Section 7.1.

Future minor version specification documents should avoid specifying minor versioning rules and reference this document in connection with rules for NFSv4 version management.

4. XDR Considerations

As an extensible XDR-based protocol, NFSv4 has to ensure interversion compatibility, in situations in which the client and server use different XDR descriptions. For example, the client may implement different variants of the same minor version or different variants that are part of the same minor version group. The XDR extension paradigm, discussed in Section 4.1, assures that these descriptions are compatible, with clients and servers able to determine and use those portions of the protocol that they both share according to the methods described in Sections 4.4.2 and 4.4.4.

4.1. XDR Extension

When an NFSv4 version change requires a modification to the protocol XDR, this is effected within a framework based on the idea of XDR extension. This is opposed to transitions between major NFS versions (including that between NFSv3 and NFSv4.0) in which the XDR for one version was replaced by a different XDR for a newer version.

The use of XDR extension can facilitate compatibility between different versions of the NFSv4 protocol. When XDR extension is used to implement OPTIONAL features, the greatest degree of inter-version compatibility is obtained. For specifics regarding rules for interversion compatibility, see Section 8.3.2. For a discussion of compatibility issues that might arise between different version groups, see Sections 8.1.2 and 8.3.3.

4.1.1. XDR Extension in General

The XDR extension approach provides a way for an XDR description to be extended in a way which retains the structure of all previously valid messages. If a base XDR description is extended to create a second XDR description, the following will be true for the second description to be a valid extension of the first:

- The set of valid messages described by the extended definition is a superset of that described by the first.

- Each message within the set of valid messages described by the base definition is recognized as having exactly the same structure/interpretation using the extended definition.

- Each message within the set of messages described as valid by the extended definition but not the base definition must be recognized, using the base definition, as part of an unsupported extension.

In general, an extension of a given XDR description consists of any set of the following changes:

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

However, none of the following may happen:

- Deletion of existing RPC operations, enum values, flag bit values and switch cases. Note that changes may be made to define use of any of these as causing an error, as long as the XDR is unaffected.
- Similarly, none of these items may be reused for a new purpose.
- Any change to the XDR-defined structure of existing requests or replies other than those listed above.

4.1.2. Particulars of XDR Extension within NFSv4

There are issues, particular to NFSv4, that affect the definition of a valid XDR extension within NFSv4.

- Because NFSv4 has been structured around compound requests and callbacks, addition of previously unspecified RPC operation codes is not allowed.
- Although they fit under the general category of enumerations, operation codes (including those for callbacks) are so central to the structure of NFSv4, that they merit special treatment.
- The fact that attribute value sets are represented within NFSv4 by nominally opaque arrays calls for special handling.

4.1.3. Rules for XDR Extension within NFSv4

In the context of NFSv4, an extension of a given XDR description consists of one or more of the following:

- Addition of previously unspecified operation codes, within the framework established by COMPOUND and CB_COMPOUND.
- Addition of previously unspecified attributes.
- Addition of new, previously unused, values to existing enums.
- Addition of previously unassigned bit values to a flag word.
- Addition of new cases to existing switches, provided that the existing switch did not contain a default case.

However, none of the following is allowed to happen:

- Any change to the structure of existing requests or replies other than those listed above.
- Addition of previously unspecified RPC operation codes, for either the nfsv4 program or the callback program, is not allowed.
- Deletion of existing RPC operations, enum values, flag bit values and switch cases. Note that changes may be made to define use of any of these as causing an error, as long as the XDR is unaffected.
- Similarly, none of these items may be reused for a new purpose.

4.2. Handling of Protocol Elements

Implementations handle protocol elements in one of three ways. Which of the following ways are valid depends on the status of the protocol element in the variant being implemented:

- The protocol element is not a part of definition of the variant in question and so is "unknown". The responder, when it does not report an RPC XDR decode error. reports an error indicative of the element not being defined in the XDR such as NFS4ERR_OP_ILLEGAL, NFS4ERR_BADXDR, or NFS4ERR_INVAL. See Section 4.4.3 for details.
- The protocol element is a known part of the variant but is not supported by the particular implementation. The responder reports an error indicative of the element being recognized as one which

is not supported such as NFS4ERR_NOTSUPP, NFS4ERR_UNION_NOTSUPP, or NFS4ERR_ATTRNOTSUPP. See Section 6.5 for details.

- The protocol element is a known part of the variant which is supported by the particular implementation. The responder reports success or an error other than the special ones discussed above.

Which of these are validly returned by the responder depends on the status of the feature element in the minor version specified in the COMPOUND or CB_COMPOUND. The possibilities which can exist are listed below.

- The protocol element is not known in the current variant of minor version. In this case all implementations of the minor version MUST indicate that the protocol element is not known.

- The protocol element is specified mandatory to not implement in the minor version. In this case as well, all implementations of the minor version MUST indicate that the protocol element is not known.

- The protocol element is defined as part of the current variant of the minor version but is not part of the corresponding base variant. In this case, the requester can encounter situations in which the protocol element is either not known to the responder , is known but not supported by the responder, or is both known to and supported by the responder.

- The protocol element is defined as an OPTIONAL part of the base minor version. In this case, the requester can expect the protocol element to be known but must deal with cases in which it is supported or is not supported.

- The protocol element is defined as a REQUIRED part of the base minor version. In this case, the requester can expect the protocol element to be both known and supported by the responder.

The listing of possibilities above does not mean that a requester always needs to be prepared for all such possibilities. Often, depending on the scope of the feature of which the protocol element is a part, handling of a previous request using the same or related protocol elements, will allow the requester to be sure that certain of these possibilities cannot occur.

Requesters, typically clients, may test for knowledge of or support for protocol elements as part of connection establishment. This may allow the requester to be aware of responder lack of knowledge of or support for problematic requests before they are actually issued.

4.3. Organization of Protocol Elements

To enable compatible operation within a version group, all of the protocol elements within an NFSv4 minor version are organized as follows:

- Each protocol element is defined as a member of exactly one feature. One important reason for this organization (see Section 6 for others) is to regularize and simplify the determination by the client and server as to what protocol elements the other party supports.

- Each feature is defined as a member of a feature package, based on how it was defined. Features established as part of a minor version at the same time belong to the same feature package.

4.4. Inter-version Interoperability

Because of NFSv4's use of XDR extension, any communicating client and server versions have XDR definitions that are each valid extensions of a third version. Once that version is determined, it may be used by both client and server to communicate. Each party can successfully use a subset of protocol elements that are both known and supported by both parties.

4.4.1. Requirements for Knowledge of Protocol Elements

With regard to requirements for knowledge of protocol elements, the following rules apply. These rules are the result of the use of XDR extension paradigm combined with the way in which extensions are incorporated in existing minor versions (for details of which see Section 7.1).

- Any protocol element defined as part of the base variant of particular minor version is required to be known by that minor version. This occurs whether the specification happens in the body of the minor definition document or is in a feature definition document that is made part of the minor version by being normatively referenced by the minor version definition document.

- Any protocol element required to be known in a given minor version is required to be known in subsequent minor version, unless and until a minor version has made that protocol element as mandatory to not implement.

- When a protocol element is defined as part of an extension to an extensible minor version, it is not required to be known in that

minor version but is required to be known by the next minor version. In the earlier minor version, it might not be defined in the XDR definition document for that minor, while in the later version it needs to be defined in the XDR definition document. In either case, if it is defined, it might or might not be supported.

- When knowledge of protocol elements is optional in a given minor version, the responder's knowledge of such optional elements must obey the rule that if one such element is known, then all the protocol elements defined in the same minor version definition document must be known as well.

For many minor versions, all existing protocol elements, are required to be known by both the client and the server, and so requesters do not have to test for the presence or absence of knowledge regarding protocol elements for which knowledge might be optional. This is the case if there has been no extension for the minor version in question. Extensions can be added to extensible minor versions as described in Section 7.1 and can be used to correct protocol flaws as described in Section 9.

Requesters can ascertain the knowledge of the responder in two ways:

- By issuing a request using the protocol element and looking at the response. Note that, even if the protocol element used is not supported by the responder, the requester can still determine if the element is known by the responder.

- By receiving a request from the responder, acting in the role of requester. For example, a client may issue a request enabling the server to infer that it is aware of a corresponding callback.

In making this determination, the requester can rely on two basic facts:

- If the responder is aware of a single protocol element within a feature package, it must be aware of all protocol elements within that feature package

- If a protocol element is one defined by the minor version specified by a request (and not in an extension), or in a previous minor version, the responder must be aware of it.

4.4.2. Establishing Interoperability

When a client and a server interact, they need to able to take advantage of the compatibility provided by NFSv4's use of XDR extension.

In this section, we will deal with situation in which the client and server are of the same version group. Later, in Section 4.4.4, we will discuss possible extensions to the inter-version-group case.

In this context, the client and server would arrive at a common variant which the client would uses to send requests which the server would then accept. The server would use that variant to send callbacks which the client would then accept. This state of affairs could arise in a number of ways:

- Client and server have been built using XDR variants that belong to the same minor version

- The client's minor version is lower than that of the server. In this case the server, in accord with Section 8.3.2, accepts the client's minor version, and acts as if it has no knowledge of extensions made in subsequent minor versions. It has knowledge of protocol elements within the current (i.e. effectively final) variant of the lower minor version.

- The client's minor version is higher than that of the server. In this case the client, in accord with Section 8.3.2, uses a lower minor version that the server will accept. In this case, the server has no knowledge of extensions made in subsequent minor versions.

There are a number of cases to consider based on the characteristics of the minor version chosen.

- The minor version consists of only a single variant (no extension or XDR corrections), so the client and the server are using the same XDR description and have knowledge of the same protocol elements.

- When the minor version consists of multiple variants (i.e. there are one or more XDR extensions or XDR corrections), the client and the server are using compatible XDR descriptions. The client is aware of some set of extensions while the server may be aware of a different set. The client can determine which of the extensions that he is aware of, are also known to the server by using the approach described in Section 4.4.3. Once this is done, the client and server will both be using a common variant. The variants that the client and the server were built with will both either be identical to this variant or a valid extension of it. Similarly, the variants that the client and the server actually use will be a subset of this variant, in that certain OPTIONAL features will not be used.

In either case, the client must determine which of the OPTIONAL protocol elements within the common version are supported by the server as described in Section 6.6.

4.4.3. Determining Knowledge of Protocol Elements

A requester may test the responder's knowledge of particular protocol elements as defined below, based on the type of protocol element.

- When a GETATTR request is made specifying an attribute bit to be tested and that attribute is not a set-only attribute, if the GETATTR returns with the error NFS4ERR_INVAL, then it can be concluded that the responder has no knowledge of the attribute in question. Other responses, including NFS4ERR_ATTRNOTSUPP, indicate that the responder is aware of the attribute in question.

- When a SETATTR request is made specifying the attribute bit to be tested and that attribute is not a get-only attribute, if the SETATTR returns with the error NFS4ERR_INVAL, then it can be concluded that the responder has no knowledge of the attribute in question. Other responses, including NFS4ERR_ATTRNOTSUPP, indicate that the responder is aware of the attribute in question.

- When a request is made including an operation with a new flag bit, if the operation returns with the error NFS4ERR_INVAL, then it can be concluded that the responder has no knowledge of the flag bit in question. Other responses indicate that the responder is aware of the flag bit in question.

- When a request is made including the operation to be tested, if the responder returns an RPC XDR decode error, or a response indicating that the operation in question resulted in NFS4ERR_OP_ILLEGAL or NFS4ERR_BADXDR, then it can be concluded that the responder has no knowledge of the operation in question. Other responses, including NFS4ERR_NOTSUPP, indicate that the responder is aware of the operation in question.

- When a request is made including the switch arm to be tested, if the responder returns an RPC XDR decode error, or a response indicating that the operation in question resulted in NFS4ERR_BADXDR, then it can be concluded that the responder has no knowledge of the operation in question. Other responses, including NFS4ERR_UNION_NOTSUPP, indicate that the responder is aware of the protocol element in question.

A determination of the knowledge or lack of knowledge of a particular protocol element is expected to remain valid as long as the clientid associated with the request remains valid.

The above assumes, as should be the case, that the server will accept the minor version used by the client. For more detail regarding this issue, see Section 8.3.2.

4.4.4. Interoperability Between Version Groups

Within a minor version group, we have complete compatibility in the sense that:

- Servers are REQUIRED to implement a core set of features which cannot change within the minor version group, allowing clients to depend on the continued existence of and support for these features as long as one remains within the minor version group.

- The set of OPTIONAL features supported or known by servers may change but clients, in using such OPTIONAL features need to be prepared for the fact that they might not be implemented on all servers implementing a minor version within the same version group.

The same level of compatibility is not provided between different minor version groups. Nevertheless, the same guarantees of inter-XDR comprehensibility apply across minor version groups. For a discussion of how this comprehensibility can be used between minor version groups, see Section 8.3.3.

5. Other NFSv4 Protocol Changes

There are a number of types of protocol changes that are outside the XDR extension framework discussed in Section 4. These changes are also managed within the NFSv4 versioning framework and may be of a number of types, which are discussed in the sections below

Each such change will be organized, documented and effected as part of a given feature, just as changes discussed in Section 4 are. The way such features will be incorporated in the NFSv4 protocol depends on a number of factors, including the types of changes included in the feature. This subject is discussed in Sections 6.7 and 7.

5.1. Non-XDR Protocol Changes

Despite the previous emphasis on XDR changes, additions and changes to the NFSv4 protocols have not been limited to those that involve changes (in the form of extensions) to the protocol XDR. Examples of other sorts of changes have been taken from NFSv4.1.

5.1.1. Field Interpretation and Use

The XDR description of a protocol does not constitute a complete description of the protocol. Therefore, versioning needs to consider the role of changes in the use of fields, even when there is no change to the underlying XDR.

Although any XDR element is potentially subject to a change in its interpretation and use, the likelihood of such change will vary with the XDR-specified type of the element, as discussed below:

- When XDR elements are defined as strings, rules regarding the appropriate string values are specified in protocol specification text with changes in such rules documented in minor version definition documents. Some types of strings within NFS4 are used in server names (in location-related attributes), user and group names, and in the names of file objects within directories. Rules regarding what strings are acceptable appear in [RFC7530] and [RFC5661] with the role of the XDR limited to hints regarding UTF-8 and capitalization issues via XDR typedefs.

- Fields that are XDR-defined as opaque elements and which are truly opaque, do not raise versioning issues, except as regards inter-version use, which is effectively foreclosed by the rules in Section 8.3.1.

  Note that sometimes a field will seem to be opaque but not actually be fully opaque when considered carefully. For example, the "other" field of stateids is defined as an opaque array, while the specification text specially defines appropriate treatment when the "other" field within it is either all zeros or all ones. Given this context, creation or deletion of reserved values for "special" stateids will be a protocol change which versioning rules need to deal with.

- Some nominally opaque elements have external XDR definitions that overlay the nominally opaque arrays. This technique is useful when the same element may be used in several ways when a switched union is not appropriate.

  For example, each pNFS mapping type provides its own XDR definition for various pNFS-related fields defined in [RFC5661] as opaque arrays. For more information about the handling of pNFS within the NFSv4 versioning framework, see Section 5.2.1.

Another form of protocol change that changes how fields are presented, without affecting the XDR occurs when there is a change in the data elements which may be presented as RDMA chunks.

5.1.2. Behavioral Changes

Changes in the behavior of NFSv4 operations are possible, even if there is no change in the underlying XDR or change to field interpretation and use.

One class of behavioral change involves changes in the set of errors to be returned in the event of various errors. When the set of valid requests remain the same, and the behavior for each of them remains the same, such changes can be implemented with only limited disruption to existing clients.

Many more substantial behavioral changes have occurred in connection with the addition of the session concept in NFSv4.1.

- Because exactly-once is semantics provided by sessions, the use of owner-based sequence values in such operations as OPEN, LOCK, LOCKU are now longer needed and the server is to ignore them.

- Because of the requirement to begin almost all COMPOUNDs with a SEQUENCE operation, the semantics of previously defined operations was changed and all formerly valid COMPOUNDs were defined as resulting in errors.

- Because the clientid is inferable from a previous SEQUENCE operation, the clientid is not needed in operations such as OPEN and LOCK, and the client is required to pass a value of zero.

Also, changes were made regarding the required server behavior as to the interaction of the MODE and ACL attributes.

5.1.3. Rules for non-XDR changes

In the past (e.g. in [RFC5661]) there was often uncertainty about whether any particular difference from NFSv4.0 was:

- A purely editorial change, which may be relevant to other minor versions.

- The correction of a protocol mistake, best handled as described in Section 9.

- A protocol improvement relevant to a new minor version or feature, to be documented as described in Section 10.3.

In order to avoid such situations, all such changes will be documented as part of a feature, specifying the specific changes relative to protocol versions that do not incorporate that new

feature. Also, to provide greater clarity about such changes, the following sets of rules apply.

The following rules apply to "substantive behavior changes", i.e. all changes in which there is a substantive change to non-error behavior. In other words, the change is not one which only changes the set of valid error codes or prescribes that different error codes are to be returned in particular situations.

- Any substantive behavior change must be part of a feature in which there is also an XDR extension present, to enable testing for presence of the feature.

- No feature including a substantive behavior change can be made REQUIRED at initial introduction.

The following rules apply to all behavioral changes.

- No feature including such a change can be introduced as an extension. While the feature may be documented in a separate feature definition document in such cases, that document should be referenced normatively by the minor version specification.

- While it is allowed to include multiple such changes in the same feature this should only be done if there is a good reason for all of these to be included or not included together. Such changes should not be included in the same feature simply because all such changes were introduced in the same minor version.

5.2. Specification of Associated Protocols

The definition of ancillary protocols is a form of protocol extension that is provided as part of pNFS and might be made available for other uses in the future.

As in the case of pNFS, the NFSv4 protocol proper would provide the basic framework for performing some protocol-related task, while allowing multiple independent means of performing that task to be defined. The version management considerations appropriate to creating such additional forms of protocol extension are discussed in Section 5.2.2

5.2.1. Associated Protocols via pNFS Mapping Types

pNFS is structured around the ability to define alternative mapping types in addition to the one defined in [RFC5661], (e.g. [RFC5663], [RFC5664]). Each mapping type specifies the data-transfer protocol

to be used to access data represented by layouts as well as mapping-type-specific XDR definitions of layout-related data structures.

Specifying a new mapping type is an additional form of protocol change within the NFSv4 version management framework. A feature consisting of the new mapping type is not tied to a specific minor version. As explained in Section 7, if a feature consists only of that single change, it is available in multiple minor versions upon publication.

Such a feature has a file system scope and the attribute fs_layout_type can used to determine whether support is present.

5.2.2. Additional Forms of Associated Protocols

The same sort of approach used for pNFS might be used in other circumstances where there is a clear need to standardize a set of protocol-related requirements and where it is desirable, for various reasons, to leave open the choice of mechanism by which those requirements might be met.

Such cases might arise where the function to be performed is likely to be too enmeshed with the structure of the file system implementation to allow a single protocol mechanism to be specified. In such cases, multiple approaches might themselves be standardized, each fitting into a template established previously using any or all of the elements used by pNFS:

- The establishment of a registry of identifiers for the standardized mechanisms to satisfy the established requirements.

- Definition of data structures related to the function to be performed to include both a mechanism identifier, and a nominally opaque portion, the real format of which is to have a mechanism-specific definition.

- The ability to specify multiple protocols to perform the same function, which may include a minor version of NFSv4, a particular use of an established protocol, or a new protocol designed for the purpose.

New instances of such a two-level approach might be established in the future, subject to the following restrictions:

- That there is a template feature establishing the requirements that the associated protocols are to meet.

- That the template feature is defined as an integral part of a particular minor version and not as an extension. This does not exclude this feature being defined in a separate document to which the minor version specification has a normative reference.

- The template feature defines the scope that the individual feature instances will have.

- The template feature defines a means by which support for particular feature instances might be determined by a client.

- That there be at least one instance of a specific protocol mechanism meeting the established requirements. To limit confusion, the requirements and the initial mechanism (an instance of the template feature) should be defined in separate documents.

The above are a minimal set of restrictions for establishing such an additional extension mechanism. The working group may, as part of defining the core feature establishing the extension mechanism specify further restrictions governing as to when minor versions are allowed to incorporate particular instances of that extension mechanism. In the absence of such restrictions, particular extensions will be incorporated, as is the case with pNFS mapping types, in all minor versions upon publication of the instance as a Proposed Standard.

6. NFSv4 Protocol Features

Individual changes, whether they are XDR extensions or other sorts of changes, are organized in term of protocol features. This is in order to

- allow the protocol documentation to more clearly specify what XDR extensions and other changes must be supported together.

- help the client determine which particular changes are present and implemented by the server.

- support the independent development and specification of changes to the protocol, without artificially tying features together in a paradigm solely based on minor versions.

- provide support for a feature-based documentation structure, as described in Section 10.3.

In contrast with some previous uses of the feature concept, every protocol element is defined as a member of exactly one protocol feature.

Because support for particular protocol features may depend on facilities provided by the underlying file systems, or may vary based on characteristics of the session within which communication is occurring, each protocol feature will be defined as having a particular scope, which may be any of the following:

- Client scope in which case support for a given feature is assumed to be uniform between given client and server as long as neither reboots.

- Session scope in which case different sessions associated with the same client may have differences as to feature support but otherwise support is uniform.

- file system scope in which case different file systems may have differences as to feature support but otherwise support is uniform.

6.1. Previous Uses of the Feature Concept

The word "feature" has been used inconsistently in previous documents bearing on issues related NFSv4 versioning, making it necessary to offer some clarification here.

- In some cases, the term "feature" is used colloquially

- In some cases, the word "feature" is used to refer to protocol extensions which are incorporated in the protocol that we refer to as "protocol elements." The term "feature elements" is similar but it differs in that it includes changes in field interpretation and use (Section 5.1.1) and protocol behavior (See Section 5.1.2).

- In some cases the word is used to refer to groups of feature elements, as defined by tables in [RFC5661] and [NFSv42]. This is similar to, but not exactly the same as the way we use the word "feature" is used in this document.

Often, as in previous minor versioning rules, it is not always clear which sense of the word "feature" is meant.

6.2. Rules for Protocol Feature Construction

A protocol feature consists of one or more valid NFSv4 changes, which work together as a functional whole. The change elements may be of any of the types described in Section 5 although the specific types of changes will affect how the feature can be integrated in the NFSv4 protocol.

A critical distinction in this regard is the one between features which can added to the protocol without a new minor version and those which require a new minor version. In this document:

- Features which do not require a new minor version are discussed in Section 7, while the process of incorporation depends on the type of features and is discussed in Sections 7.1, 9, 5.2.1, and 5.2.2,

- For handling of the remaining features which do require a new minor version, see Section 8.

6.3. Statuses of Features

Each feature has one of three statuses with regard to each minor version of which it might be a part.

- The feature is a REQUIRED part of the minor version.

- The feature is not a REQUIRED part of the minor version, but may be implemented as part of that version, i.e. it is OPTIONAL

- The feature is not a valid part of the minor version.

  For features which have been previously defined as valid, this is represented as being "mandatory to not implement" as opposed to simply not being undefined.

These statuses define whether a client implementing the minor version has to be prepared for the protocol feature's non-support by a server implementation, even if the feature in question is known by the server.

The working group is still free to make recommendations regarding the desirability of server and client support for particular features in particular minor versions in the minor version definition document, or in other, presumably informational, documents.

Particular protocol elements have similar statuses, which are derived from a combination of the status of feature of which the protocol element, the status of that protocol element within its feature, and, in some cases, within other supported features. See Section 6.4 for details.

In addition to feature status, there may be other constraints that define when an implementation must or may support a feature. In particular, support for one feature may require support for another, or the presence of one feature may require that another feature not be supported.

6.4. Statuses of Protocol Elements Within Features

The status of a protocol element within its containing feature reflects two pieces of information that are used in determining support for feature and associated protocol elements.

- A status value that allows support for the feature to be inferred based on support for the protocol element. This is referred to as the protocol element's E-to-F status.

- A status value that allows support for the feature element to be inferred based on support for the feature. This is referred to as the protocol element's F-to-E status with regard to the feature.

The purpose of defining these status values is to allow the support or non-support for one protocol elements to be determined based on responses for others, avoiding the complexity that a client would have to deal with if each such support decision were independent. A simpler model would have been to simply assign protocol elements to feature-based support equivalence classes and require all protocol elements in a feature to be supported or not supported together. This approach was not adopted because it is not compatible with many current and expected feature patterns:

- Many existing protocol features contain protocol elements that are optional in the context of the feature.

- Some existing protocol elements are used by more than one feature.

- Boolean attributes that indicate the presence of support for a given feature are tied to that feature, even though the attribute can be supported when the feature is not, in which case the attribute is supported and has the value FALSE.

The following are possible E-to-F statuses.

- Support or non-support for the feature is always the same as that for the protocol element. This is represented as an "IFF" value.

- Support for the feature can be inferred from support for the protocol element but not necessarily the reverse. This is represented as an "SINF" value.

- Lack of support for the feature can be inferred from lack of support for the protocol element but not necessarily the reverse. This is represented as an "NSINF" value.

- Lack of support for the feature can be inferred from lack of support for the protocol element but the reverse can be determined by using the protocol element to determine whether support for the feature is present. An example would be a Boolean attribute indicating whether support for the feature is present. This is represented as an "SVAL" value.

Generally, it will be clear how a client may determine whether any particular OPTIONAL feature is supported. Typically there will be one or more protocol elements belonging to the feature whose E-F status is "IFF" or "SVAL". In these cases, support for the protocol elements in question can be determined as described in Section 6.4

In more complicated cases, the feature specification should clearly specify how to determine whether support is present.

The following are possible F-to-E statuses.

- Support for the protocol element is REQUIRED when support for the feature is present.

- Support for the protocol element is OPTIONAL when support for the feature is present.

- Support for the protocol element unaffected by the presence of support for the feature.

The overall status of a feature element within a minor version is generally determined as follows:

- If there are one or more REQUIRED features which give the protocol element an F-to-E status of REQUIRED, then the overall status of the protocol element within the minor version is REQUIRED.

- Otherwise, if there are one or more REQUIRED or OPTIONAL features which give the protocol element an F-to-E status of REQUIRED or OPTIONAL, then the overall status of the protocol element within the minor version is OPTIONAL.

- If neither of the above is true, the protocol element is treated as not a part of the minor version. That is, it is treated as mandatory to not implement.

In some cases the overall status may be different from that specified above. For example, it could be that there were two features, each of which is OPTIONAL, and it is specified that exactly one of these must always me supported. In such a case, if both features assign a

protocol element an F-to-E status of REQUIRED, then the overall status of the protocol element is REQUIRED.

6.5. Determining Protocol Element Support

If it has already been determined that a particular protocol element is known to the server, the client can determine whether it is supported based on its type, as follows:

- If the protocol element is an attribute, the supported_attr attribute can be interrogated to determine if support is present.
- If the protocol element is an operation, the operation can be attempted, with an error of NFS4ERR_NOTSUPP indicating the operation is known but not supported.
- If the protocol element is a switch case, use of that case can be attempted, with an error of NFS4ERR_UNION_NOTSUPP indicating t the operation is known but not supported.
- If the protocol element is an operation flag bit and the operation is REQUIRED, use of that flag bit can be attempted with an error of NFS4ERR_NOTSUPP indicating the protocol element is known but not supported.
- If the protocol element is an operation flag bit and the operation defines an error to return in the case of unsupported flag bits, use if that flag bit can be attempted with the specified error indicating the operation is known but not supported.

Once this is done, all of the protocol elements the client is aware of can be divided into three sets:

- Those that the server is unaware of and thus cannot support.
- Those that the server knows about but does not support.
- Those that the server supports.

Information obtained in the process of determining knowledge of protocol elements (see Section 4.4.3) may be saved and used in connection with the interrogations above. For example, in testing for knowledge of a given operation, the specific error code returned will indicate support or non-support as well as indicating support or non-support, as well as knowledge of the corresponding operation.

Note that in doing so care needs to be taken regarding protocol elements associated with features whose scope is more limited than

that of an entire client, since support may be different for different sessions or different file systems.

6.6. Feature Discovery

In many cases, a client will need to determine whether particular features are supported before using protocol elements that are part of those features. While some clients may choose to defer this determination until the features in question are actually needed, others may make the determination as part of first connecting with a server, using a session or accessing a file system, depending on the scope of the feature in question.

Once such a determination of feature support or non-support are made, the client may assume that it remains valid and will not change so long as the object defining the feature scope remains valid.

- For features of client scope as long as the clientid remains valid.
- For features of session scope as long as the sessionid remains valid.
- For features of file system scope as long as the clientid and fsid both remain valid.

In making this determination, the client is entitled to rely on, and the server is REQUIRED to obey any inter-feature constraints that are specified as applying to the minor version being used.

The presence or absence of particular features may be determined in a number of ways:

- For features which are REQUIRED within a given minor version, the client can treat the fact that the server accepted a request with that minor version (and did not return NFS4ERR_MINOR_VERSION_MISMATCH) as indicating that support is present.
- For features which consist only of the addition of a pNFS layout type, the fs_layout_type attribute for the fs in question can be interrogated and scanned for the layout type.
- For features which consist only of the addition of an instance of a feature template as defined in Section 5.2.2, the template feature definition will describe the means by which the presence of support for particular feature instances is to be determined.

For the remaining features, which are all OPTIONAL and contain an XDR-extending protocol element, the E-to-F statuses of the constituent protocol elements (see Section 6.4) can be used to determine if support is present within the scope defined by the feature in question. In most cases, support for the protocol element is tested as described in Section 6.5.

- If there are one or more protocol elements whose status is "IFF", support for any of these may be tested, with the result determining support for the feature

- If there are one or more protocol elements whose status is "SVAL", support for it can be tested, and if present the value returned can be tested as described by the feature specification, resulting in a determination of support for the feature.

- If there are protocol elements with statuses of "SINF" and "NSINF", testing of these protocol elements can be used, although, it is not always certain that testing all such will always resolve the question.

- If none of these approaches are determinative, the feature specification should define a method of resolving the question.

Once the set of supported features is determined:

- For protocol elements which have an F-to-E status of REQUIRED for at least one supported feature, it can be assumed that support is present.

- For other protocol elements which have an F-to-E status of OPTIONAL for at least one supported feature, support needs to be tested for as described in Section 6.5.

- For the remaining protocol elements, it can be assumed that support is not present.

6.7. Feature Incorporation

All protocol changes will be organized, documented and effected as part of a given feature. This includes XDR extension and the various sorts of non-XDR-based changes allowed.

Such features may be made part of the protocol in a number of ways:

- In new minor versions, as discussed in Section 8.

- In separately documented new features. When new features are OPTIONAL and do not include any non-XDR-based changes, they may be incorporated in an extensible minor version under construction. See Section 7.1 for details.

- When appropriate compatibility arrangement are in effect, they may be used to correct protocol problems in already approved minor versions and features. See Section 9 for details.

7. Extensions within Minor Versions

The NFSv4 version management framework allows, with certain restrictions, features to be added to existing minor versions

- In the case of features which consist only of a pNFS mapping type, the protocol may be extended by publishing the new mapping type definition as a Proposed Standard. This effects an extension to all minor versions in which pNFS is a valid feature.

  Similar extension facilities could be made available if additional pNFS-like extension frameworks were created (See Section 5.2.2).

- Minor versions designated as extensible (see Section 7.1) may be extended by the publication of a standards-track document defining the additional feature. Details are set out below. The features to be added are considered OPTIONAL in the extensible minor version and must consist only of valid XDR-based extensions

7.1. Adding Features to Extensible Minor Versions

Addition of features to an extensible minor version will take advantage of the existing NFSv4 infrastructure that allows optional features to be added to new minor versions, but without in this case requiring any change in the minor version number. Adding features in this way will enable compatibility with existing clients and servers, who may be unaware of the new feature.

7.2. Use of Feature Specification Documents

Each such extension will be in the form of a working-group standards-track document which defines one or more new OPTIONAL features. The definition of each of the new feature may include one or more "protocol elements" which extend the existing XDR as already discussed (in Section 4.1). Other sorts of XDR modification are not allowed. Protocol elements include new operations, callbacks, attributes, and enumeration values. The functionality of some existing operations may be extended by the addition of new flags bits in existing flag words and new cases in existing switched unions.

New error codes may be added but the set of valid error codes to be returned by an operation is fixed, except that existing operations may return new errors to respond to situations that only arise when previously unused flag bits are set or when extensions to a switched union are used.

Also, certain additional documents may be produced at this time to simplify the process of using new versions that contain the extension, and to help co-ordinate the process of making further extensions. See Section 10.5 for details.

Each such additional feature will become, for all intents and purposes, part of the current NFSv4 minor version upon publication of the description as a Proposed Standard, enabling such extensions to be used by new client and server implementations without, as previously required, a change in the value of the minor version field within the COMPOUND operation.

The working group has two occasions to make sure that such features are appropriate ones:

- At the time the feature definition document becomes a working group document, the working group needs to determine, in addition to the feature's general compatibility with NFSv4, that the XDR assignments (i.e. additional values for operation callback and attribute numbers, and for new flags and switch values to be added to existing operations) associated with the new feature are complete and do not conflict with those in the existing protocol or those currently under development.

- At the time the working group document is complete, the working group, in addition to normal document review, can and should look at what prototype implementations of the feature have been done and use that information to determine the work-ability and maturity of the feature.

7.3. Compatibility Issues

Because the receiver of a message may be unaware of the existence of a specific extension, certain compatibility rules need to be observed. In some cases (e.g., addition of new operations or callbacks or addition of new arms to an existing switched union) older clients or servers may be unable to do XDR parsing on an extension of whose existence they are unaware. In other cases (e.g., error returns) there are no XDR parsing issues but existing clients and servers may have expectations as to what may validly be returned. Detailed discussion of these compatibility issues appears below:

- Issues related to messages sent to the server are discussed in Section 7.3.1.

- Issues related to messages sent to the client are discussed in Section 7.3.2.

7.3.1. Compatibility Issues for Messages Sent to Servers

This section deals with compatibility issues that relate to messages sent to the server, i.e., requests and replies to callbacks. In the case of requests, it is the responsibility of the client to determine whether the server supports the extension in question before sending a request containing it for any purpose other than determining whether the server is aware of the extension. In the case of callback replies, the server demonstrates its awareness of proper parsing for callback replies by sending the associated callback.

Regarding the handling of requests:

- Existing server implementations will return NFS4ERR_NOTSUPP or NFS4ERR_OP_ILLEGAL in response to any use of a new operation, allowing the client to determine that the requested operation (and potentially the feature in question) is not known or known but not supported by the server.

- Clients can determine whether particular new attributes are supported by a given server by examining the value returned when the supported_attr attribute is interrogated. Clients need to do this before attempting to use attributes defined in an extension since they cannot depend on the server returning NFS4ERR_ATTRNOTSUPP for requests which include a mask bit corresponding to a previously unspecified attribute number (as opposed to one which is defined but unsupported).

- Existing server implementations that do not recognize new flag bits will return NFS4ERR_INVAL, enabling the client to determine that the new flag value is not supported by the server.

- Existing server implementations that do not recognize the new arm of a switched union in a request will return NFS4ERR_INVAL or NFS4ERR_UNION_NOTSUPP, enabling the client to determine that the new union arm is not supported by the server.

Regarding the handling of responses to callbacks:

- Error values returned to the server for all callbacks that do not use new features will only be those previously allowed. Only when

the server uses a new extension feature can a previously invalid error value be returned.

- Callback replies may only include a new arm of an existing switched union when the server, typically in the callback being responded to, has used a feature element associated with the feature that defined the new switched union arm.

7.3.2. Compatibility Issues for Messages Sent to Clients

This sections deals with compatibility issues that relate to messages sent to clients, i.e., request replies and callbacks. In both cases, extensions are only sent to clients that have demonstrated awareness of the extensions in question by using an extension associated with the same feature.

Regarding the handling of request replies:

- Error values returned to the client for all requests that do not use new features will only be those previously allowed. Only when the server uses a new extension feature can a previously invalid error value be returned.

- Replies may only include a new arm of an existing switched union when the server, typically in the request being responded to, has used a feature element associated with the feature that defined the new switched union arm.

Regarding the handling of callback requests, the server needs to be sure that it only sends callbacks to those clients prepared to receive and parse them.

- In most cases, the new callback will be part of a feature that contains new (forward) operations as well. When this is the case, the feature specification will specify the operations whose receipt by a server is sufficient to indicate that the client issuing them is prepared to accept and parse the associated callbacks.

- For callbacks associated with features that have no new operations defined, the feature specification should define some way for a client to indicate that it is prepared to accept and parse callbacks that are part of the extension. For example, a flag bit in the EXCHANGE_ID request may serve this purpose.

- In both of the above cases, the ability to accept and parse the specified callback is considered separate from support for the callback. The feature specification will indicate whether support

for the callback is required whenever the feature is used by the client. In cases in which support is not required, the client is free to return NFS4ERR_NOTSUPP upon receiving the callback.

7.4. Relationship Between Minor Versioning and Extensions within a Minor Version

Extensibility of minor versions are governed by the following rules:

- Minor versions zero and one are not extensible. Each has a fixed set of OPTIONAL features as described in [RFC7530] and [RFC5661].

- Minor versions beyond one are presumed extensible as discussed herein. However, any statement within the minor version specification disallowing extension will cause that minor version to be considered non-extensible.

- No new feature may be added to a minor version once the specification document for a subsequent minor version becomes a working group standards-track document.

Even when a minor version is non-extensible, or when a previous minor version is closed to further extension, the features that it contains are still subject to updates to effect protocol corrections. In many cases, making an XDR change, in the form of an extension will be the best way of correcting an issue. See Section 9 for details.

While making minor versions extensible will decrease the frequency of new minor versions, it will not eliminate the need for them. Protocol features that cannot be used as extensions (see Section 8.1.1 require a new minor version.

In addition, change which involve modifications to the set of protocol elements which are REQUIRED or mandatory to not implement require a new minor version which starts a new minor version group. Changes to the organization of protocol features are treated similarly, since they have a similar potential to cause interversion incompatibility. See Section 8.1.2 for details.

8. Minor Versions

8.1. Creation of New Minor Versions

It is important to note that this section, in describing situations that would require new minor versions or minor version groups to be created, does not thereby imply that situations will exist in the future. Judgments regarding desirability of future changes will be

made by the working group or its successors and any guidance that can be offered at this point is necessarily quite limited.

Creation of a new minor version or minor version group is an option that the working group retains. The listing of situations below that would prompt such actions is not meant to be exhaustive.

New minor versions are to be documented as described in Section 10.6.

8.1.1. New Minor Versions within an Existing Group

The following sorts of features are not allowed as extensions and would require creation of a new minor version:

- Features that incorporate any of the non-XDR-based changes discussed in Sections 5.1.1 and 5.1.2.

- Any feature which includes a new mapping type (as described in Section 5.2.1) and includes any other change.

  To prevent new mapping types from evading this restriction by splitting the mapping type and other changes into two separate changes, if new mapping type makes a reference to protocol changes in an extension, it may not be incorporated in minor versions in which that extension is defined but only in later minor versions.

- Any feature that creates a new expansion mechanism as described in in Section 5.2.2.

8.1.2. New Minor Version Groups

The following sorts of changes can only occur in the context of a new minor version group:

- Addition of REQUIRED new features.

- Changes to the status of existing features including converting features to be mandatory to not implement.

- Changes to the status of existing feature elements within features, causing those feature elements to be required or optional when they previously had not been.

- Changes to the scope of existing features.

- Changes to feature organization or to inter-feature constraints. Such changes may have the effect of making support for some change

element required or optional in circumstances in which it previously had not been

Changes to the status or organization of features will, in most case, result in changes to the status of individual protocol elements, changing them between REQUIRED and OPTIONAL, or making them mandatory to not implement.

Conversion of protocol elements to be mandatory to not implement, will not, as had previously been the practice, result in their deletion from the protocol XDR. However, the server will be REQUIRED to treat such protocol elements as not known when responding to requests within minor versions in which they are not to be implemented. See Sections 4.4.3 and 8.3.2 for details.

Such changes give rise to potential compatibility issues. In most cases in which such changes will actually be made, careful consideration of compatibility issues can limit the scope of such issues or ensure that compatibility issues actually experienced are quite limited.

This is opposed to the first new minor version group, that associated with minor version one, which resulted in a situation in which clients for minor version zero could not interoperate with servers for minor version one and vice versa. Issues related to the question of what to do about such situations are discussed in Section 8.1.3

The addition of REQUIRED features may serve to illustrate the issues. Such additions pose no compatibility issue for existing clients. On the other hand, all servers will need to be updated to support the new features. The effort required and any potential for disruption depend on the scope of the feature being added.

A number of features introduced as REQUIRED in NFSv4.1 can serve to illustrate the issues.

- suppattr_exclattr was added as a REQUIRED attribute. This was very simple for servers to implement.

- RECLAIM_COMPLETE was added as a REQUIRED operation.

- TEST_STATEID and FREE_STATEID were added as REQUIRED operations.

Some examples of potential feature status changes may be helpful in illustrating compatibility issues

- Converting a REQUIRED feature to be mandatory to not implement poses the greatest level of difficulty from an interoperability

point of view. Clients need to change to use an alternative means of providing the functionality provided by the feature. Existing servers need to be updated, even if there is a replacement feature available.

Such a transition is only possible without disruption if the feature in question has already fallen into disuse.

- Converting an OPTIONAL feature to be mandatory to not implement poses similar difficulties. If clients have ceased to use the feature, after they have become aware, formally or informally, that it is moribund, the difficulties can be quite limited.

- Converting a REQUIRED feature to be OPTIONAL poses no difficulty for existing server implementations. It may pose difficulties for clients who have not made preparations for server non-support of the feature.

  The degree of such difficulties and the readiness of clients to make such changes should be key considerations in making such a state transition.

- Converting an OPTIONAL feature to be REQUIRED poses no difficulty for existing client implementations. The difficulties for existing server implementations depend on the scope of the feature involved and the set of implementations without support for the feature in question.

  The degree of such difficulties and the readiness of servers to make such changes should be key considerations in making such a state transition. Nevertheless, it should not be the only consideration. If all existing servers support the feature, it does not thereby follow that the transition should be made. The possible effect of making server development more complicated should also be considered.

A number of other changes allowed only in a new minor version group, raise analogous issues.

- In the case of inter-feature constraints or similar reorganizations, the basic issue is whether the client has to deal with the absence of a protocol element when it previously had not had to deal with that or the server has to provide support for a protocol element in situations in which it previously had not had to. When a set of changes cause both sorts of issues, the greatest interoperability difficulties arise, making such a set of changes hard to implement.

- If a feature scope is changed to be more fine-grained, the client has to deal with combinations of support and non-support it previously had not had to deal with, while the reverse forces the server to maintain a unity of support it had previously not had to. The unlikely case of conversion between session and file system scope causes difficulties for both parties.

The tradeoff between interoperability issues and desirable changes to the protocol is one for the working group to make. If the decision is made to create a new minor version group, the working group has decided that absolute compatibility is not required. Nevertheless, it should strive to make necessary changes as non-disruptive as possible.

8.1.3. Limits on Minor Version Groups

The guidance that needs to be offered with regard to appropriate limits on changes that form new version groups does not appear reducible to specific rules.

Instead it is appropriate to return to the basic goal of allowing the NFSv4 protocol to adapt to future circumstances as they develop. Although this was not explicitly stated, it seems to be intended that this would not involve generation of an essentially a new protocol, even if that were, in some sense, a better one.

So the best way we can address the question of limits on new version groups is to state that the purpose of the rules in this document, including the creation of new minor version groups is not the creation of a successor protocol to NFSv4.

If this or a future working group does find itself defining a new file access protocol, it would be helpful if proper care were taken to retain what is valuable in the intellectual heritage of NFSv4. Nevertheless, in doing so, it is important not to assume that adherence to the rules in this document, is, in and of itself, a guarantee that the new protocol is thereby a version of NFSv4.

In dealing with such a future changed situation, the better option would be to face the issue of necessary change forthrightly and acknowledge that such a large change creates a fundamentally new situation. Appropriate responses might include replacing the XDR in whole or in part, using a successor to XDR, or other means.

8.2. Role of Minor Versions

Clearly, the ability to provide protocol extensions without creation of a new minor version, has lessened the role of minor versions in extending the NFSv4 protocol to meet future needs.

We have gone from a situation in which there was a single mechanism, creation of a new minor version, to extend the protocol, to a three-level approach:

- OPTIONAL features which extend but do not change protocol semantics may be added without creating a new minor version.
- Other OPTIONAL features may be added by creating a new minor version within an existing version group, as long as the sets of protocol elements which are REQUIRED and mandatory to not implement.
- Changes which do as the sets of protocol elements which are REQUIRED and mandatory to not implement are only allowed in a new minor version group.

This document does explore the situations that, if they arise, would require the creation of new minor versions or version groups. This does not imply that such situations will exist or that the working will choose to address things in that way. Such choices are left for future decision by the working group and the IESG.

The discussion in Section 8.1.3 raises similar issues. It is possible that situations might arise that would cause NFSv4 development to be done outside the framework established here. Nevertheless, this does not imply that such situations will arise.

8.3. Minor Version Interaction Rules

This section addresses issues related to rules #11 and #13 in the minor versioning rules in [RFC5661]. With regard to the supersession of minor versioning rules, the treatment here overrides that in [RFC5661] when either of the potentially interacting minor versions has not yet been published as a Proposed Standard.

Note that these rules are the only ones directed to minor version implementers, rather than to those specifying new minor versions.

8.3.1. Minor Version Identifier Transfer Issues

Each relationship between a client instance and a server instance, as represented by a clientid, is to be devoted to a single minor version. If a server detects that a COMPOUND with an inappropriate minor version is being used, it MUST reject the request. In doing so, it may return either NFS4ERR_BAD_CLIENTID or NFS4RR_MINOR_VERS_MISMATCH.

As a result of the above, the client has the assurance that the set of REQUIRED and OPTONAL features will not change within the context of a single clientid. Server implementations MUST ensure that the set of supported features and protocol elements does not change within such a context.

8.3.2. Minor Version Intra-Group Compatibility

Within a set of minor versions that belong to the same minor version group, it is relatively easy for clients and servers to provides the needed compatibility by following the following rules.

- Servers supporting a given minor version MUST support any earlier minor version in the same minor version group and return appropriate errors for use of protocol elements that were not a valid part of that earlier minor version. For details see below.

- Servers supporting a given minor version MUST, in returning errors for operation which were a valid part of the minor version, return the errors allowed for the current operation in the minor version actually being used.

- Clients MUST deal with an NFS4ERR_MINOR_VERS_MISMATCH error by a searching for a lower minor version number in the same minor version group that the server will accept.

With regard to protocol elements not known in a given minor version, the appropriate error codes are given below. Essentially, the server, although it has a more extensive XDR reflective of a newer minor version, must act as a server with a more limited XDR would.

- When an operation is used which is not known in the specified minor version, NFS4ERR_OP_ILLEGAL (as opposed to NFS4ERR_NOTSUPP) should be returned.

- When an attribute is used which is not known in the specified minor version, NFS4ERR_INVAL (as opposed to NFS4ERR_ATTRNOTSUPP) should be returned.

- When a switch case is used which is not known in the specified minor version, NFS4ERR_BADXDR (as opposed to NFS4ERR_UNION_NOTSUPP) should be returned. Even though the message may be XDR-decodable by the server's current XDR, it is not so according to the minor version being used.

- When a flag bit is used which is not known in the specified minor version, NFS4ERR_INVAL (as opposed to NFS4ERR_NOTSUPP Or any other error defined as indicated non-support a flag bit) should be returned.

8.3.3. Minor Version Inter-Group Compatibility

It is desirable for client and server implementations to support a wide range of minor versions. The difficulty of doing so can be affected by choices made by the working group in defining those minor versions, and the particulars of the changes made which establish new version groups.

Options for compatibility are affected by the scale and frequency of the changes which require a new minor version group and the working group needs to take needs for inter-group compatibility into account when making such changes. In all cases, the following rules apply:

- Servers supporting a given minor version SHOULD support minor versions in earlier minor version groups. When doing so, it MUST behave appropriately given the definition of the minor version used. For details see below.

- Clients SHOULD deal with an NFS4ERR_MINOR_VERS_MISMATCH error by a searching for a lower minor version number within the appropriate minor version range until it finds one that the server will accept.

In some cases, the server needs to behave as a more restricted one for an earlier minor version might, despite it having extensions for protocol elements added in later minor versions. In these cases, the errors described in Section 8.3.2 should be returned in this case as well.

In the case in which the earlier version contains protocol elements subsequently made mandatory to not implement, the server needs to know of those protocol elements and not return the errors that would appropriate if the most up-to-date minor version were used. In cases in which support for these protocol elements is REQUIRED, support will have to be provided by the server and if it cannot do that, it MUST return NFS4ERR_MINOR_VERS_MISMATCH for any requests using that minor version.

In addition to using an appropriate subset of the protocol XDR definition, the server needs to respect the non-XDR elements of the earlier minor version group as well. In particular, the serve needs to:

- Support REQUIRED features as specified by the earlier minor version group.
- Support (or not) features according to E-to-F statuses specified by the earlier minor version group.
- Respect the inter-feature constraints specified by the earlier minor version group.
- Respect the feature scopes specified by the earlier minor version group.
- Support (or not) protocol elements according to the F-to-E statuses specified in the earlier minor version group.

9. Correction of Existing Minor Versions and Features

The possibility always exists that there will be a need to correct an existing feature in some way, after the acceptance of that feature or a minor version containing it, as a Proposed Standard. While the working group can reduce the probability of such situations arising by waiting for running code before considering a feature as done, it cannot reduce the probability to zero. As features are used more extensively and interact with other features, previously unseen flaws may be discovered and will need to be corrected.

Such corrections are best done in a document obsoleting or updating the RFC defining the relevant feature definition document or minor version specification. In making such corrections, the working will have to carefully consider how to assure interoperability with older clients and servers.

Often, corrections can be done without changing the protocol XDR. In many cases, a change in client and server behavior can be implemented without taking special provision with regard to interoperability with earlier implementations. In those case, and in cases in which a revision merely clarifies an earlier protocol definition document, a new document can be published which simply updates the earlier protocol definition document. Subsequently, the indexing material would be updated to reflect the existence of the newer document.

In other cases, it is best if client or server behavior needs to change in a way which raises interoperability concerns. In such

cases, incompatible changes in server or client behavior should not be mandated in order to avoid XDR changes.

9.1. XDR Changes to Implement Protocol Corrections

When XDR changes are necessary as part of correcting a flaw, these should be done in a manner similar to that used when implementing new minor versions or features within them. In particular,

- Existing XDR structures may not be modified or deleted.
- XDR extensions may be used to correct existing protocol facilities in a manner similar to those used to add additional optional features. Such corrections may be done in an otherwise non-extensible minor version, if the working group judges it appropriate.
- When a correction is made to an OPTIONAL feature, the result is similar to a situation in which there are two independent OPTIONAL features. A server may choose to implement either or both.
- When a correction is made to a required feature, the situation becomes one in which neither the old nor the new version of the feature is required. Instead, it is required that a server support at least one of the two, while each is individually OPTIONAL. Although use of the corrected version is ultimately better, and may be recommended, it should not be described as "RECOMMENDED", since the choice of which version to support if only one is supported will depend on the needs of clients, which may be slow to adopt the updated version.
- In all of the cases above, it is appropriate that the old version of the feature, be considered obsolescent, with the expectation that the working group might, in a later minor version, decide that the older version is to become mandatory to not implement.

Issues related to the effect of XDR corrections on existing documents, including co-ordination with other minor versions, are discussed in Section 10.7.

By doing things this way, the protocol with the XDR modification can accommodate clients and servers that support either the corrected or the uncorrected version of the protocol and also clients and servers aware of and capable of supporting both alternatives.

- A client that supports only the earlier version of the feature (i.e., an older unfixed client) can determine whether the server it is connecting to supports the older version of feature. It is

capable of interoperating with older servers that support only the unfixed protocol as well as ones that support both versions.

- A client that supports only the corrected version of the feature (i.e., a new or updated client) can determine whether the server it is connecting to supports the newer version of the feature. It is capable of interoperating with newer servers that support only the updated feature as well as ones that support both versions.

- A client that supports both the older and newer version of the feature can determine which version of the particular feature is supported by the server it is working with.

- A server that supports only the earlier version of the feature (i.e., an older unfixed server) can only successfully interoperate with older clients. However newer clients can easily determine that the feature cannot be used on that server.

- A server that supports only the newer version of the feature (i.e., a new or updated server) can only successfully interoperate with newer clients. However, older clients can easily determine that the feature cannot be used on that server. In the case of OPTIONAL features, clients can be expected to deal with non-support of that particular feature.

- A server that supports both the older and newer versions of the feature can interoperate with all client variants.

By using extensions in this manner, the protocol creates a clear path which preserves the functioning of existing clients and servers and allows client and server implementers to adopt the new version of the feature at a reasonable pace.

10. Documentation of Features, Extensions, Minor Versions, and Protocol Corrections

As mentioned previously, NFSv4 is evolving towards a finer-grained documentation model. This trend will be continued by:

- The use of extensions within minor versions.

- Features that are added by a minor version being documented in feature definition documents rather than within the minor version specification itself.

10.1. Documentation Approach

Documentation of future changes to the NFSv4 protocol will use feature specification documents as described in Section 10.3. There are a number of ways in which such documents may be used, which reflect the different ways in which features are incorporated in the NFSv4 protocol, as discussed in Section 6.7

This documentation approach is intended to avoid the unnecessary production of large documents in which many unrelated features are tied together because either:

- The entire protocol is described in a single document, as happened with NFSv4.0 (in [RFC7530]) and NFSv4.1 (in [RFC5661]).
- Many unrelated features are described in a single document as occurred with NFSv4.2 (in [NFSv42]).

The production of a larger number of smaller documents will streamline document production and review. A potential problem is that a profusion of smaller documents might cause difficulty for those learning about and implementing the protocol.

The production of indexing material described in Section 10.2 is intended to limit such difficulties. The result will be that, for operations and attributes, we will have essentially a single table of contents, referencing material from multiple minor version definition documents and feature specification documents.

10.2. Indexing material

The items listed below, referred to collectively as "Indexing material" will be useful in many contexts. The reason for frequently publishing such material is to prevent a situation in which large numbers of documents must be scanned to find the most current description of a particular protocol element.

- A table mapping operations and callbacks to the most recent protocol definition document containing a description of that operation.
- A table mapping attributes to the most recent protocol definition document containing a description of that attribute.
- A table giving, for each operation in the protocol, the errors that may validly be returned for that operation. If possible, it would be desirable to give, as does [RFC5661], the operations which may validly return each particular error.

- A table giving for each operation, callback, and attribute and for each feature element in a published extension giving its status (REQUIRED, OPTIONAL, or mandatory-to-not implement), the name of the feature of which it is a part, its associated E-to-F and F-to-E status values and information about other features for which it has a non-empty F-to-E status value. This would be similar to the material in Section 14 of [NFSv42], expanded to include all feature elements.

10.3. Feature Specification Documents

Features will be documented in the form of a working-group standards-track document which define one or more features. Generally, only closely related features should be defined in the same document.

The definition of each of the new features may include one or more "feature elements" which change the protocol in any of the ways discussed in Section 5. Feature elements include new operations, attributes, and enumeration values. Note that in this context, "Operations" include both forward and callback operations. The functionality of some existing operations may be extended by the addition of new flags bits in existing flag words, by new cases in existing switched unions, and by valid semantic changes to existing operations.

Such feature definition documents would contain a number of items, following the pattern of the NFSv4.2 specification. The only difference would be that while the NFSv4.2 specification defines a number of features to be incorporated into NFSv4.2, the feature definition documents would each define a single feature, or a small set of closely related features.

In addition to a general explanation of the feature(s) in question, the items to be included in such feature definition documents would be as listed below. In some cases these items, in addition to descriptive text, would contain fragments of XDR code, to aid in preparation of XDR files that include the additions defined by the feature added to the base protocol that is being extended. For information regarding preparation of such XDR files, see Section 10.4.

- Description of new operations (corresponding to Sections 15 and 16 of [NFSv42]). Such descriptions will contain XDR code defining the structure the arguments and results of the new operation along with preparatory XDR definitions used only by that operation.

- Description of any modified operations (corresponding to Section 15 of [NFSv42]). Such description may contain XDR code

defining the new flag bits, enum values, and cases to be added to existing switched unions. Note that addition of new attributes is not considered an extension of GETATTR, SETATTR, VERIFY, or NVERIFY.

- Description of new attributes (corresponding to Section 13 of [NFSv42]). XDR code defining the types of the attributes would be part of this description.

- Description of any added error codes (corresponding to Section 12.1 of [NFSv42]).

- All operation descriptions, whether for new or modified operations, should indicate when operations or the corresponding results may be presented as RDMA chunks.

- A set of XDR code fragments giving the numeric values of added operation codes, attribute numbers, and error codes.

- Descriptions of all other extensions made to existing flag words, enums and switched unions used by existing operations. Such descriptions will contain XDR code defining the new flag bits, enum values, and cases to be added to existing switched unions.

- Descriptions of all new structures, enums, flag words, and switched unions that are used by more than one new operation, or which are available for future use by multiple operations. Such descriptions will contain XDR code defining the new structures/ union and assigning the new numeric values for enum and flag bits.

- A listing giving the valid errors for each new operation and callback (corresponds to Sections 12.2 and 12.3 of [NFSv42]).

- For each feature, a table giving for each feature element that is part of the feature, its overall status within the minor version and its E-to-F and F-to-E status values. This would be similar to the material in Section 14 of [NFSv42] but restricted to the feature(s) defined in the document and expanded to include all feature elements.

- A table presenting support requirement for each protocol element which is either a part of a feature defined in the document or has an F-to-E status with relation with a feature defined in the document. This could present the F-to-E status value for each relevant combination of feature element and feature. An alternative presentation would give, for each protocol element, boolean expressions in term of supported features, that allows or that guarantees support for the specified element.

- All of the additional Sections required for RFC publication, such as "Security Considerations", "IANA considerations", etc.

Note that the listing above is not intended to define, in detail, the structure of the specification. Rather, the intention is to define the things it needs to contain. If there would be no content for a particular element, there is no need for an empty section corresponding to that list element. If it makes more sense to describe a new structure together with an extended one, then the need for a readily understandable document is primary.

10.4. XDR File Considerations

As mentioned previously, feature specification documents will contain, in addition to description of XDR extensions, XDR code fragments that embody those extensions. There will be various occasions on which people will have occasion to produce XDR files that combine one or more extensions together with the XDR for an existing minor version.

- When a minor version is specified by a number of feature specification documents, there will be a need to produce, in as simple fashion as possible, the corresponding XDR specification document for the new minor version.

- Within an extensible minor version, there will be a need for those developing and testing the feature to have an XDR file that incorporates XDR definitions from early drafts of the feature specification document.

- Also, for an extensible minor version, there will be a need to periodically produce Consolidated XDR documents that reflect all features approved as Proposed Standards and thus incorporated in the current minor version.

- Developers may need to be able to produce XDR files that reflect particular combination of approved features, features under development or experimental features not yet ready for working group consideration.

We are assuming here that the primary task is producing XDR files and that corresponding XDR documents can be produced relatively easily if there is a well understood process to produce the underlying XDR files.

The Feature specification document should contain all of the necessary lines of XDR codes to be added to a base XDR file to effect

the extension. The only remaining issue is where to place each addition to arrive at the correct consolidated file.

- One could rely on those preparing updated XDR file to place the additional XDR code lines in the appropriate place, based on inference from the document text.

- One could rely on the Feature Specification Document to indicate, in the descriptive text, where each XDR extension is to be placed.

- One could formalize a set of conventions whereby the appropriate placements are indicated by specific instructions embedded within comments within the XDR code fragments to be placed.

10.5. Additional Documents to Support Protocol Extension

Additional documents will be required from time to time. These documents will eventually become RFC's (informational or standards track as described below), but the work of the working group and of implementers developing features will be facilitated by a progression of document drafts that incorporate information about new features that are being developed or have been approved as Proposed Standards.

10.5.1. Minor Version Indexing Document

One document will organize existing material for a minor version undergoing extension so that implementers will not have to scan a large set of feature definition documents or minor version specifications to find information being sought. Successive drafts of this document will serve as an index to the current state of the extensible minor version. Some desirable elements of this indexing document would include:

- A list of all feature definition documents that have been approved as working group documents but have not yet been approved as Proposed Standards.

- All of the items of indexing material (see Section 10.2) appropriately adjusted to reflect the contents of all extensions accepted as Proposed Standards.

The frequency of updates for this document will be affected by implementer needs and the ability to easily generate document drafts, preferably by automated means. The most desirable situation is one in which a new draft is available soon after each feature reaches the status of a Proposed Standard.

10.5.2. Consolidated XDR Document

This document will consist of an updated XDR for the protocol as a whole including feature elements from all features and minor versions accepted as Proposed Standards.

A new draft should be prepared whenever a new feature within an extensible minor version is accepted as a Proposed Standard. In most cases, feature developers will be using a suitable XDR which can then be reviewed and published. In cases in which multiple features reach Proposed Standard status at approximately the same time, a merge of the XDR changes made by each feature may be necessary.

10.5.3. XDR Assignment Document

This document will contain consolidated lists of XDR value assignments that are relevant to the protocol extension process. It should contain lists of assignments for:

- operation codes (separate lists for forward operations and for callbacks)
- attribute numbers
- error codes
- bits within flag words that have been extended since they were first introduced.
- enumeration values for enumerations which have been extended since they were first introduced.

For each set of assignments, the individual assignments may be of three types:

1. permanent assignments associated with a minor version or a feature extension that has achieved Proposed Standard status.

   These assignments are permanent in that the assigned value will never be re-used. However, a subsequent minor version may define some or all feature elements associated with a feature to be mandatory to not implement.

2. provisional assignments associated with a feature under development (i.e., one which has been approved as a working group document but has not been approved as a Proposed Standard).

Provisional assignments are not are not permanent and the values assigned can be re-used in certain circumstances. In particular, when a feature with provisional assignments is not progressing toward the goal of eventual Proposed Standard status, the working group can judge the feature effort to have been abandoned, allowing the codes formerly provisionally allocated to be reclaimed and reassigned.

3. definition of individual assignments or ranges reserved for experimental use.

A new draft of this document should be produced, whenever:

- A minor version or feature specification is accepted as a Proposed Standard.
- A new feature is accepted for development and a draft of the corresponding working-group standards-track document is produced
- A feature previously accepted for development is abandoned.
- The working group decides to make some change in assignments for experimental use.

10.5.4. Transition of Documents to RFC's

Each of these documents should be published as an RFC soon after the minor version in question ceases to be considered extensible. Typically this will happen when the working group makes the specification for the subsequent minor version into a working group document. Some specifics about the individual documents are listed below:

- The most current draft of the indexing document for the minor version would be published as an informational RFC.
- The most current draft of the consolidated XDR document should be published as a standards-track RFC. It would update the initial specification of the minor version
- The most recent draft of the XDR assignment document should be published as an informational RFC.

Handling of these documents in the event of a post-approval XDR correction is discussed in Section 10.7

10.6. Documentation of New Minor Versions

Minor versions should be documented by specifying and explaining the changes made relative to the previous minor version.

Features added to the minor version should be documented in their own feature specification documents and normatively referenced.

Changes to the status or organization of existing features should be documented by presenting a summary of the status of all existing protocol elements, their relationship to OPTIONAL features, and any relevant feature dependencies.

In addition, to avoid situation where a large number of minor versions must be scanned to find the most recent valid treatment of a specific protocol element, minor version definition documents will contain the indexing material described in Section 10.2.

10.7. Documentation of XDR Changes for Corrections

In the event of an XDR correction, as discussed above, some document updates will be required. For the purposes of this discussion we call the minor version for which XDR correction is required minor version X and the minor version on which development is occurring minor version Y.

The following discusses the specific updated documents which could be required:

- The specification of the feature in question will have to be updated to explain the issue, how it was fixed, and the compatibility and upgrade strategy. Normally this will require an RFC updating the associated feature specification document. However, in the case of a correction to a feature documented in a minor version definition document, the RFC will update that document instead.

- An updated XDR for minor version X will be produced and will be published as a updated to the minor version specification RFC for minor version X.

  When the correction is to feature documented in a minor version definition, a single RFC will contain both updates to the minor version specification RFC.

- An updated minor version indexing document for minor version X is desirable but not absolutely necessary.

The question of updated minor version indexing documents for minor versions between X and Y should be addressed by the working group on a case-by-case basis.

- An updated XDR assignment document will be required. It should be based on the most recent such document associated with minor version Y and will serve as the basis for later XDR assignment drafts for minor version Y.

The informational RFC's associated with minor version Y (version indexing document and XDR assignment document) will contain the effects of the correction when published. Similarly, the minor version specification RFC will contain the XDR changes associated with the correction.

11. Security Considerations

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

As features and minor versions are 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.

12. 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.

13. References

13.1. 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>.

13.2. Informative References

[NFSv42] Haynes, T., Ed., "NFS Version 4 Minor Version 2", January 2016, <http://www.ietf.org/id/draft-ietf-nfsv4-minorversion2-40.txt>.

Work in progress.

[NFSv42-dot-x]
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-40.txt>.

Work in progress.

[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>.

[RFC5663] Black, D., Fridella, S., and J. Glasgow, "Parallel NFS (pNFS) Block/Volume Layout", RFC 5663, DOI 10.17487/RFC5663, January 2010, <http://www.rfc-editor.org/info/rfc5663>.

[RFC5664] Halevy, B., Welch, B., and J. Zelenka, "Object-Based Parallel NFS (pNFS) Operations", RFC 5664, DOI 10.17487/RFC5664, January 2010, <http://www.rfc-editor.org/info/rfc5664>.

[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>.

Appendix A. Acknowledgements

The author wishes to thank Tom Haynes of Primary Data for his role in getting this effort started and his work in co-authoring the first version of this document.

The author also wishes to thank Chuck Lever and Mike Kepfer of Oracle for their thorough document reviews and many helpful suggestions.

Author's Address

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

Phone: +1 978 474 2011
Email: davenoveck@gmail.com