NFSv4 D. Noveck
Internet-Draft HP
Intended status: Standards Track July 2, 2015
Expires: January 3, 2016

NFSv4 Version Management
draft-ietf-nfsv4-versioning-01

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 January 3, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

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 rules and a framework to allow for future changes via the creation of new protocol versions and certain forms of modification of existing versions. These version management rules allow changes to be implemented in a way that maintains compatibility with existing clients and servers.

1.1. Existing Minor Versions

Previously, all such 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 (see Section 15.2 of [RFC7530]) procedure 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 is specified by [RFC7530] with the XDR description appearing in [RFC7531].
- Minor version 1 is specified by [RFC5661]) with the XDR description appearing in [RFC5662].
- Minor version 2 is specified by [NFSv42] (in terms of changes from [RFC5661]). The XDR description appears in [NFSv42-dot-x]

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. Many changes can be done 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 5 should be used.

This document starts by presenting the conceptual framework on which NFSv4 versioning is built.

- First we discuss (in Section 4) 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.

Using this framework, we then look at the ways the NFSv4 protocol can be changed.

- The addition of new features to existing minor versions is discussed in Sections 7 and 8.

- New Minor versions can be constructed, as described in Section 9.

- Issues relating to the correction of protocol errors in existing features and minor versions are discussed in Section 10.

2. Terminology

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

The word "feature" has been used inconsistently in previous treatments of NFSv4 versioning. Sometimes it is used to indicate a specific XDR extension, while at other times it has been used to indicate a set of multiple such extensions which are either supported or not supported together.

In this document, we use the word "feature" in the second sense, while individual protocol extensions which are incorporated in a feature are referred to as "protocol elements." The term "feature elements" is similar but it differs in that it includes changes in field interpretation and use (Section 4.2.1) and protocol behavior (See Section 4.2.2). See Section 6 for more details.

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 9.3) or implicitly (see Section 8).

- 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 8.3.

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

As noted above, 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 term is often not appropriate. Instead, what document writers need to do is stated without these specialized terms. In any case, one should not conclude that the normative character of this document is compromised by this.

- In speaking of the previously existing possible statuses of feature elements, the lower-case versions of these terms are used, following the practice of [RFC3530]. This is despite the fact that the corresponding uses of these terms in [RFC5661] was switched to upper-case, for no clear reason, and similarly in the case [RFC7530], presumably due to inertia.

- In speaking of the potential statuses of features, the words "required" and "non-required" are used. By using the latter term, we focus on the fact that the feature in question is not required to be supported, while treating any potential recommendation for support as out-of-scope.

- 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, or in a quotation, sometimes indirect, from another document.

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. As a result, 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, 4.2.3, 4.3.1, and 4.3.2.

- Rules relating to the composition of changes into features are addressed in Section 6.1

- Minor version construction, including rules applicable to features which cannot be used as extensions to existing minor versions are addressed in Section 9.1

- Minor version interaction rules are discussed in Sections 9.2.2 and 9.2.1.

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 4.2.3 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 9.2 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 10 for all previously specified minor versions and extensibility of previously specified minor versions is to be handled in accord with Section 8.

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

4. NFSv4 Protocol Changes

Protocol changes that are to be managed within the NFSv4 versioning framework may be of a number of types, which are discussed in the sections below. Such changes include, but are not limited to, changes in the underlying protocol XDR.

Each such change will be organized, documented and effected as part of a given feature. The way this will be done depends on a number of factors, including the types of changes included in the feature. This subject is discussed in Section 6.5.

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 non-required features, the greatest degree of inter-

version compatibility is obtained. For specifics regarding rules for interversion compatibility, see Section 9.2.2.

4.1.1. XDR Extension in General

XDR extension allows 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.

An extension of a given XDR description consists of any of the following:

- 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 chosen to structure itself 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 sets are represented within 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:

- 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 structure of existing requests or replies other than those listed above.

4.2. 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.

4.2.1. 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 type, 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 object 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 9.2.

  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 4.3.1.

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.

4.2.2. 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.

Many such 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.

4.2.3. 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 10.

- A protocol improvement relevant to a new minor version or feature, to be documented as described in Section 6.4.

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 rules apply:

- Any such 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 such a change can be made required at initial introduction.

- 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.

4.3. 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 4.3.2

4.3.1. Associated Protocols via pNFS Mapping Types

pNFS is structured around the ability to define alternate 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, it is available in multiple minor versions upon publication.

4.3.2. 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 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 feature establishing the requirements that the associated protocols are to meet.

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

- 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 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 may specify further restrictions governing when minor versions may incorporate particular instances of that extension mechanism.

5. Documentation Approach

Documentation of future changes to the NFSv4 protocol will use feature specification documents as described in Section 6.4. There are a number of ways in which such documents may be used, as discussed in Section 6.5

5.1. Indexing material

The following items, 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 document containing a description of that operation.

- A table mapping attributes to the most recent 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 or not, or mandatory-to-not implement), and its relationship to the feature which allows its inclusion (i.e., required for every implementation of the feature, or optional in the presence of the feature). This would be similar to the material in Section 14 of [NFSv42], expanded in scope to include all feature elements.

6. NFSv4 Features

Individual changes, whether they are XDR extensions or other sorts of changes, are organized in term of 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.

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

- provide support for a feature-based documentation structure, as described in Section 6.4.

6.1. Rules for Feature Construction

A 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 4 although the specific types of changes will affect how the feature can be integrated in the NFSv4 protocol.

6.2. Feature Statuses

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..

- 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 being undefined.

These statuses define whether a client implementing the minor version has to be prepared for the existence of the feature and/or its non-support 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.

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.

6.3. Feature Discovery

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, a client can determine whether the feature is supported by seeing if the minor version is supported.

- For non-required features that contain an XDR-extending protocol element, a client can try to use techniques described in Section 8.2.1 to determine what features the server supports.

- For non-required features that do not contain an XDR-extending protocol element, appropriate feature discovery facilities can be constructed on an ad hoc basis by defining a non-required per-server or per-fs boolean attribute to serve as an indication of support. To address compatibility issues with earlier servers, an appropriate default value to assume when the attribute is not supported should be specified.

6.4. 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 4. Feature 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, 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 in question, the items to be included in such feature definition documents would be:

- Description of new operations (corresponding to Sections 16 and 17 of [NFSv42]).

- Description of any modified operations (corresponding to Section 15 of [NFSv42]).
- Description of new attributes (corresponding to Section 13 of [NFSv42]).
- 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 summary description of all changes made by this feature to the XDR definition of the protocol, including operation codes, attribute numbers, added flag bits and enumeration values, and request and response structures for new operations together with the other XDR extensions needed to support them.
- A listing giving the valid errors for each new operation and callback (corresponds to Sections 12.2 and 12.3 of [NFSv42]).
- A table giving for each new feature element its status (required or not) and its relationship to the feature(s) being described (i.e., required for every implementation of the feature, or optional in the presence of the feature). This would be similar to the material in Section 14 of [NFSv42] but restricted to the feature(s) defined in the document and expanded in scope to include all feature elements.
- All of the additional Sections required for RFC publication, such as "Security Considerations", "IANA considerations", etc.

6.5. 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 9.
- In separately documented new features. When new features are non-required and do not include any non-XDR-based changes, they may be incorporated in an extensible minor version under construction. See Section 8 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 10 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 4.3.2).

- Minor versions designated as extensible (see Section 8) may be extended by the publication of a standards-track document defining the additional feature. Details are set out in Section 8. The features to be added are considered non-required in the extensible minor version and must consist only of valid XDR-based extensions

8. 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.

8.1. 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 non-required 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.

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.

8.2. 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 8.2.1.

- Issues related to messages sent to the client are discussed in Section 8.2.2.

8.2.1. 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 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 NFS4ERRATTRNOTSUPP 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.

8.2.2. 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.

8.3. Additional Documents to be Produced

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.

8.3.1. 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 5.1) 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.

8.3.2. 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.

8.3.3. 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 support.

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.

8.3.4. 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 10.1

8.4. 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 non-required 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 may be made 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 10 for details.

While making minor versions extensible will decrease the frequency of new minor versions, it will not eliminate the need for them. In particular:

- A new minor version will be required for any change in the status of a feature element (i.e., an operation, callback, attribute, added flag or switch case). For example, changes which make feature elements Recommended, Required or Mandatory to Not Implement will require a minor version.

- Any incompatible semantic change in the required or allowed processing of an existing operation or attribute will require a minor version.

- Any change that extends the set of errors returned that an existing operation, with the exception noted above. New errors may be added when the conditions that give rise to these new errors cannot arise as long as new flag bits or switched union arms are not used. In these cases, it is clear that existing clients cannot receive these errors.

- Any change in the mapping of feature elements to features will require a minor version. For example, if a feature is to be split into two separate features clients would no longer be able to infer support for one operation from support for the other, in the same way that had been done previously, invalidating logic in existing clients

9. Minor Versions

9.1. Minor Version Construction

In addition to the sorts of non-required features that may be made in the context of extensible minor version, a number of other sorts of changes may be made in a new minor version. Because such changes have the potential to disrupt inter-version such changes should only be made after careful consideration of the effects on interversion interoperability.

- Addition of new features that incorporate any of the non-XDR-based changes discussed in Sections 4.2.1 and 4.2.2. Such features must always be introduced as non-required.

- Addition of required new features.

- Changes to the status of existing features or to inter-feature constraints.

- Changes to feature organization. Such changes may have the effect of making support for some change element obligatory in circumstances when it had not been previously.

9.2. Minor Version Interaction

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.

9.2.1. 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 allowed features will not change within the context of a single clientid. Server implementations MUST ensure that the set of supported features does not change within such a context.

9.2.2. Minor Version Compatibility Issues

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.

Within a set of minor versions that have exactly the same set of required features, it is relatively easy for clients and servers to provide appropriate compatibility and they are well-advised to do so.

Servers SHOULD accept any minor version number within the range and return NFS4ERR_OPNOTSUPP or NFS4ERR_OP_ILLEGAL for non-supported operations based on the version number.

Clients SHOULD deal with an NFS4ERR_MINOR_VERS_MISMATCH error by searching the appropriate minor version range for one the server will accept.

Servers and clients MAY deal with changes in the set of required features by supporting at least the union of the set of required features for all minor versions within the range to be supported. This may involve logic to specifically withhold support for features when not allowed for a particular minor version.

9.3. Minor Version Documentation

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 non-required 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 5.1.

10. 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 bis document updating the RFC defining the relevant feature definition document or minor version specification. In making such a correction, 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. However, incompatible changes in server or client behavior should not be mandated in order to avoid XDR changes. 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 a non-required feature, the result is similar to a situation in which there are two independent non-required 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 non-required. 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 10.1.

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 non-required 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 allowing client and server implementers to adopt the new version of the feature at a reasonable pace.

10.1. Documentation of XDR Changes

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, March 1997.

13.2. Informative References

[NFSv42] Haynes, T., Ed., "NFS Version 4 Minor Version 2", April 2015, <http://www.ietf.org/id/draft-ietf-nfsv4-minorversion2-38.txt>.

Work in progress.

[NFSv42-dot-x] Haynes, T., Ed., "NFS Version 4 Minor Version 2 Protocol External Data Representation Standard (XDR) Description", April 2015, <http://www.ietf.org/id/draft-ietf-nfsv4-minorversion2-dot-x-38.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, April 2003.

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

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

[RFC5663] Black, D., Fridella, S., and J. Glasgow, "Parallel NFS (pNFS) Block/Volume Layout", RFC 5663, January 2010.

[RFC5664] Halevy, B., Welch, B., and J. Zelenka, "Object-Based Parallel NFS (pNFS) Operations", RFC 5664, January 2010.

[RFC7530] Haynes, T. and D. Noveck, "Network File System (NFS) Version 4 Protocol", RFC 7530, March 2015.

[RFC7531] Haynes, T. and D. Noveck, "Network File System (NFS) Version 4 External Data Representation Standard (XDR) Description", RFC 7531, March 2015.

Author's Address

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

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