NFSv4 D. Noveck
Internet-Draft HPE
Updates: 5661, 7862 (if approved) December 3, 2016
Intended status: Standards Track
Expires: June 6, 2017

Rules for NFSv4 Extensions and Minor Versions
draft-ietf-nfsv4-versioning-08

Abstract

This document describes the rules relating to the extension of 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 and other RFCs defining minor versions.

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 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 extension rules contained in this document allow extensions and other changes to be implemented in a way that maintains compatibility with existing clients and servers.

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.

Creation of a new minor version is no longer the only way in which protocol changes may be made. Optional features may be added as extensions and protocol corrections can be proposed, specified and implemented within the context of a single minor version. Creation of new minor versions remains available when needed.

The goal of allowing extensions within the context of a minor version is to provide more implementation flexibility while preserving interoperability on protocol upgrade. As described in Section 4.4, two implementations can each choose to implement a subset of available extensions, enabling interoperation to proceed just as if both implementations supported only the parts of the protocol they both support.

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

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.

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 protocol extension, 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. 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 [RFC7530] 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 within a given minor version. In using these terms, RFCs which specify the status of features inform:

- client implementations whether they need to deal with the absence of support for these features.

- server implementations whether they need to provide support for these features.

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 after that original definition.

- 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 client and server which implements a specific minor version will implement some particular variant of that minor version. Each variant 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, in the case of a minor version that has not had a protocol correction, one may refer to the set of REQUIRED features for that minor version since it is the same for all variants within the minor version. See Section 9 for a discussion of correcting an existing minor version.

3. Consolidation of Extension Rules

In the past, the only existing extension rules were the minor versioning rules that were 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 extension 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 extension rules, as was done in the minor versioning rules in [RFC5661], this document defines multiple sets of rules that deal with the various forms of protocol change provided for in the NFSv4 extension framework.

- The kinds of XDR changes that may be made to extend NFSv4 are addressed in the rules in Section 4.2.

- Minor version construction, including rules applicable to changes which cannot be made in extensions to existing minor versions are addressed in Section 7.1

- Minor version interaction rules are discussed in Sections 8.1 and 8.2.

This document supersedes minor versioning rules appearing in the minor version specification RFC's, including those in [RFC5661] and also the modification to those rules mentioned in [RFC7862]. As a result, potential conflicts among 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.

- 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 for a discussion of these issues, including how potential conflicts between rules are to be resolved.

- Otherwise, any conflict between the extension 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 6.

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

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 and server may implement different variants of the same minor version, in that they each might add different sets of extensions to the base minor version.

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 method described in Section 4.4.2.

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 XDR extension approach 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 extension not provided for.

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. In this case, as long as the rules in Section 6 are followed, no change in minor version number is needed and the extension may be effected in the context of a single minor version.

4.2. Rules for XDR Extension within NFSv4

In the context of NFSv4, given the central role of COMPOUND and CB_COMPOUND, addition of new RPC procedures is not allowed and the enumeration of operations and callback operations have a special role.

The following XDR extensions, by their nature, affect both messages sent by requesters (i.e. requests, callbacks), and responders (i.e. replies, callback replies).

- Addition of previously unspecified operation codes, within the framework established by COMPOUND and CB_COMPOUND. These extend the appropriate enumeration and the corresponding switches devoted

to requests and responses for the associated direction of operation.

- Addition of previously unspecified attributes. These add additional numeric constants that define each attribute's bit position within the attribute bit map, together with XDR typedefs that specify the attributes' format within the nominally opaque arrays specifying sets of attributes.

Other sorts of changes will generally affect one of requests, replies, callback, or callback replies. Although all are valid XDR extensions, the messages that are affected may determine whether the extension requires a new minor version (see Section 7) or can be made as an extension within an existing minor version (see Section 6).

- 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.

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 procedures, for either the nfsv4 program or the callback program.
- Deletion of existing RPC procedures, operation codes, 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.

4.3. Handling of Protocol Elements by Responders

Implementations handle protocol elements in received in requests and callbacks 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.

- 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 protocol element in the minor version specified in the COMPOUND or CB_COMPOUND. The possibilities which can exist when dealing with minor versions that have not been subject to corrections are listed below. See Sections 9.1 and 9.3 for a discussion of the effects of protocol correction.

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

- The protocol element is part of a feature 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 to 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 a responder's lack of knowledge of or support for problematic requests before they are actually used to effect user requests.

4.4. Inter-version Interoperability

Because of NFSv4's use of XDR extension, any communicating client and server versions have XDR definitions such that each is a valid extension 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 to 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 the XDR extension paradigm combined with the way in which extensions are incorporated in existing minor versions (for details of which see Section 6).

- 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 versions, 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, 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. 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 6 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 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 8.2, 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 8.2, 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 use the approach described in Section 4.4.3 to determine which of the extensions it knows about are also known by the server. 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, just as it does for OPTIONAL features introduced as part of a minor version.

It is best if client implementations make the determination as to the support provided by the server before acting on user requests. This includes the determination of the common protocol variant and the level of support for OPTIONAL protocol elements.

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. Note that in the case of attribute or flag bits, use of a request that refers to 2 or more bits of undetermined status (known versus unknown) may return results which are not particularly helpful. In such cases, when the response is NFS4ERR_INVAL, the requester can only conclude that at least one of the bits is unknown.

- 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 generally be concluded that the responder has no knowledge of the flag bit in question, as long as the requester is careful to avoid other error situations in which the operation in question is defined as returning NFS4ERR_INVAL. 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 8.2.

4.5. XDR Overlay

XDR additions may also be made by defining XDR structures that overlay nominally opaque fields that are defined to allow such incremental extensions.

For example, each pNFS mapping type provides its own XDR definition for various pNFS-related fields defined in [RFC5661] as opaque arrays.

Because such additions provide new interpretations of existing fields, they may be made outside of the extension framework as long as they obey the rules previously established when the nominally opaque protocol elements were added to the protocol.

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.

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.

All such changes that have been made in the past have been made as part of new minor version. Future change of these sorts may not be done in an extension but can only be made in a new minor version.

5.1. 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 8.1.

  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. Such cases are discussed in Section 4.5.

5.2. 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. Even though there was no change to the XDR for existing operations, many existing operations and COMPOUNDs consisting only of them became invalid.

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

6. Extending Existing Minor Versions

Extensions to the most recently published NFSv4 minor version may be made by publishing the extension as a Proposed Standard, unless the minor version in question has been defined as non-extensible. A document need not update the document defining the minor version, which remains a valid description of the base variant of the minor version in question.

In addition to following the rules for XDR extensions in Section 4.2, such extensions must also obey the following rules in order to allow interoperability to be established, as described in Section 4.4:

- Additions to the set of callback requests and extensions to the XDR for existing callback operations can only be made if the server can determine, based on the client's actions, that client is aware of the changes, before sending those new or extended callbacks.

- XDR extensions that affect the structures of responses to existing operations can only be made if the server can determine, based on the client's actions, that it is aware of the existence of XDR changes, before sending responses containing those extensions. This determination can be based on the request being responded to, but that is not required. Use of any protocol element defined in the extension can be the basis of the determination, provided that the requirements for determining client awareness are clearly stated.

Corrections to protocol errors (see Section 9) may be accomplished by publishing an extension, including a compatible XDR change which follows the rules above. Such documents will update the defining documents for the minor version to be corrected.

7. Minor Versions

7.1. Creation of New Minor Versions

It is important to note that this section, in describing situations that would require new minor versions 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 is an option that the working group retains. The listing of situations below that would prompt such actions is not meant to be exhaustive.

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 and 5.2.
- Features whose XDR changes do not follow the rules in Section 6.
- Addition of REQUIRED new features.
- Changes to the status of existing features including converting features to be mandatory to not implement.

8. 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.

8.1. 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 OPTIONAL 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.

8.2. Minor Version Compatibility

The goal of the NFSv4 extension model is to enable compatibility including compatibility between clients and servers implementing different minor versions.

Within a set of minor versions that define the same set of features as REQUIRED and mandatory to not implement, it is relatively easy for clients and servers to provide the needed compatibility by adhering to the following practices.

- Servers supporting a given minor version should support earlier minor versions within that set and return appropriate errors for use of protocol elements that were not a valid part of that earlier minor version. For details see below.

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

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

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 indicating non-support of a flag bit) should be returned.

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 or minor version. In making such corrections, the working group 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 cases, 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.

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 a minor version for which optional features may no longer be added, if the working group decides that it is an appropriate way to compatibly effect a correction.
- 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. See Section 9.2 for a detailed discussion of interoperability issues.
- When a correction is made to a REQUIRED feature, the situation becomes one in which the old version of the feature remains REQUIRED while the corrected version, while OPTIONAL, in intended to be adopted to provide correct operation. Although use of the

corrected version is ultimately better, and may be recommended, it should not be described as "RECOMMENDED", since the choice of versions to support will depend on the needs of clients, which may be slow to adopt the updated version. The nature of such corrections is such that it may result in situations in which different variants of the same minor version may not support both support the corrected version. See Section 9.3 for details.

- 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, change the status of the uncorrected version. See Section 9.4 for more detail.

9.2. XDR Corrections to OPTIONAL features

By defining the corrected and uncorrected version as independent OPTIONAL features, 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.

Based on the type of client:

- A client that uses 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.

Based on the type of server:

- 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.

9.3. XDR Corrections to REQUIRED features

Interoperability issues are similar to those for the OPTIONAL case described above (in Section 9.2). However, because the use of the uncorrected version is REQUIRED, servers have to support this until there is a minor version change. Nevertheless, there is the opportunity for clients and servers to implement the corrected version, while maintaining necessary interoperability with earlier implementations.

The following types of servers can exist:

- Servers only aware of and supporting the uncorrected version, such as servers developed before the issue requiring correction was known.

- Servers aware of both versions while only supporting the uncorrected version.

- Servers aware of and supporting both versions.

With the exception of clients which do not use the feature in question, the following sorts of clients may exist:

- Clients only aware of and prepared to use the uncorrected version, such as those developed before the issue requiring correction was known.

  Clients developed before the correction was defined would be of this type. They would be capable of interoperating with all of the types of servers listed above, but could not use the corrected version.

- Clients aware of both versions while only prepared to use the uncorrected version.

  Some clients developed or modified after the correction was defined would be of this type, until they were modified to support the corrected version. They would also be capable of interoperating with all of the types of servers listed above, but could not use the corrected version.

- Clients aware of and prepared to use either version.

  Such clients would be capable of interoperating with all of the types of servers listed above, and could use the corrected version with servers that supported it.

- Clients aware of both versions while only prepared to use the newer, corrected, version.

  Such clients would only be capable of interoperating with servers that supported the correct version. With other types of server, they could determine the absence of appropriate support at an early stage and treat the minor version in question as unsupported by the server. Such clients are only likely to be deployed when the majority of servers support the corrected version.

9.4. Addressing XDR Corrections in Later Minor Versions

As described in Sections 9.2 and 9.3, a corrected XDR can be incorporated in an existing minor version and be used, while an existing uncorrected version is still supported. Nevertheless, the uncorrected version will remain part of the protocol until its status is changed in a later minor version.

One possible change that could be made in a later minor version is to define the uncorrected version as mandatory to not implement. Because of the difficulty of determining that no clients depend on support for the uncorrected version, it is unlikely that this step would be appropriate for a considerable time.

In the case of a correction to a REQUIRED feature, there are a number of less disruptive changes that could be made earlier:

- Changing the uncorrected version from REQUIRED to OPTIONAL while REQUIRING that servers support at least one of the two versions.

  This would allow new server implementations to avoid support for the uncorrected version.

- Changing the corrected version from OPTIONAL to REQUIRED, making both versions REQUIRED.

  This would allow new clients to depend on support for the corrected version being present.

- Changing the uncorrected version from REQUIRED to OPTIONAL while changing the corrected version from OPTIONAL to REQUIRED.

  This would complete the shift to the corrected version once clients are prepared to use the corrected version.

In making such changes, interoperability issues would need to be carefully considered.

10. Security Considerations

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

11. IANA Considerations

The current document does not require any actions by IANA.

12. References

12.1. 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>.

[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>.

[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>.

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

12.2. Informative References

[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>.

Appendix A. Acknowledgements

The author wishes to thank Tom Haynes of Primary Data for his role in getting the effort to revise NFSV4 version magement started and for his work in co-authoring the first version of this document.

The author also wishes to thank Chuck Lever and Mike Kupfer of Oracle and Bruce Fields of Red Hat for their helpful reviews of this and other versioning-related documents.

Author's Address

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

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