NFSv4   T. Haynes
Internet-Draft   Primary Data
Intended status: Standards Track   D. Noveck
Expires: May 14, 2015   Dell
                                                       November 10, 2014

NFSv4 Version Management
draft-haynes-nfsv4-versioning-02

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 for minor versioning set out in this document supersede the multiple sets of minor versioning rules in RFC3530 and RFC5661.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

To address the requirement of an NFS protocol that can evolve as the need arises, the Network File System (NFS) version 4 (NFSv4) protocol contains the rules and framework to allow for future minor changes or versioning. These versioning rules SHOULD maintain compatibility with existing clients and servers.

A large portion of such changes will be part of new minor versions. The COMPOUND procedure (see Section 14.2 of [RFC3530]) specifies the

minor version being used by the client. The CB_COMPOUND (see Section 15.2 of [RFC3530]) procedure specifies the minor version being used by the server on callbacks.

Each minor version is specified by one or more standards track RFC:

Minor version 0 is specified by [RFC3530]

Minor version 1 is specified principally by [RFC5661]

Minor version 2 is specified principally by [NFSv42].

In addition to the creation of new minor versions, two other sorts of versioning are discussed in this document:

- o Addition of OPTIONAL features to existing minor versions.

- o XDR-based extensions used to correct protocol bugs in existing minor versions or associated features.

2. Terminology

A basic familiarity with the NFSv4 terminology is assumed in this document, the reader is pointed to [RFC3530].

3. Consolidation of the Minor Version Rules

Previously, versioning rules had been being maintained and specified in the Standards Track RFCs, defining the individual minor versions. As a result, versioning rules were modified on an ad hoc basis for each new minor version.

This document defines a set of versioning rules, including rules for minor version construction, that applies to the NFSv4 protocol as a whole. The rules are subject to change but any such change should be part of a standards track RFC obsoleting or updating this document.

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

- o 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 rules, the protocol stays as it is.

o Otherwise, any conflict between the minor versioning rules in this document and those in minor version specification RFC's are to be resolved based on the treatment in this document.

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

4. The Minor Versioning Rules

The following items represent the basic rules for the development of minor versions.

1.   Procedures are not added or deleted.

     To maintain the general Remote Procedure Call (RPC) model, NFSv4 minor versions will not add to or delete procedures from the NFS program.

2.   Minor versions may add operations to the COMPOUND and CB_COMPOUND procedures.

     The addition of operations to the COMPOUND and CB_COMPOUND procedures does not affect the RPC model.

     * Minor versions may append attributes to the bitmap4 that represents sets of attributes and to the fattr4 that represents sets of attribute values.

       This allows for the expansion of the attribute model to allow for future growth or adaptation.

     * Minor version X must append any new attributes after the last documented attribute.

       Since attribute results are specified as an opaque array of per-attribute, XDR-encoded results, the complexity of adding new attributes in the midst of the current definitions would be too burdensome.

3.   Minor versions must not modify the structure of an existing operation's arguments or results.

     Again, the complexity of handling multiple structure definitions for a single operation is too burdensome. New operations should

be added instead of modifying existing structures for a minor version.

This rule does not preclude the following adaptations in a minor version:

* adding bits to flag fields, such as new attributes to GETATTR's bitmap4 data type, and providing corresponding variants of opaque arrays, such as a notify4 used together with such bitmaps

* adding bits to existing attributes like Access Control Lists (ACL) that have flag words

* extending enumerated types (including NFS4ERR_*) with new values

* adding cases to a switched union

4. Note that when adding new cases to a switched union, a minor version must not make new cases be REQUIRED. While the encapsulating operation may be REQUIRED, the new cases (the specific arm of the discriminated union) is not. The error code NFS4ERR_UNION_NOTSUPP is used to notify the client when the server does not support such a case.

5. Minor versions must not modify the structure of existing attributes.

6. Minor versions must not delete operations.

   This prevents the potential reuse of a particular operation "slot" in a future minor version.

7. Minor versions must not delete attributes.

8. Minor versions must not delete flag bits or enumeration values.

9. Minor versions may declare an operation MUST NOT be implemented.

   Specifying that an operation MUST NOT be implemented is equivalent to obsoleting an operation. For the client, it means that the operation MUST NOT be sent to the server. For the server, an NFS error can be returned as opposed to "dropping" the request as an XDR decode error. This approach allows for

the obsolescence of an operation while maintaining its structure so that a future minor version can reintroduce the operation.

1. Minor versions may declare that an attribute MUST NOT be implemented.

2. Minor versions may declare that a flag bit or enumeration value MUST NOT be implemented.

10. Minor versions may declare an operation to be OBSOLESCENT, which indicates an intention to remove the operation (i.e., make it MANDATORY TO NOT implement) in a subsequent minor version. Such labeling is separate from the question of whether the operation is REQUIRED or RECOMMENDED or OPTIONAL in the current minor version. An operation may be both REQUIRED for the given minor version and marked OBSOLESCENT, with the expectation that it will be MANDATORY TO NOT implement in the next (or other subsequent) minor version.

11. Note that the early notification of operation obsolescence is put in place to mitigate the effects of design and implementation mistakes, and to allow protocol development to adapt to unexpected changes in the pace of implementation. Even if an operation is marked OBSOLESCENT in a given minor version, it may end up not being marked MANDATORY TO NOT implement in the next minor version. In unusual circumstances, it might not be marked OBSOLESCENT in a subsequent minor version, and never become MANDATORY TO NOT implement.

12. Minor versions may downgrade features from REQUIRED to RECOMMENDED, from RECOMMENDED to OPTIONAL, or from OPTIONAL to MANDATORY TO NOT implement. Also, if a feature was marked as OBSOLESCENT in the prior minor version, it may be downgraded from REQUIRED to OPTIONAL from RECOMMENDED to MANDATORY TO NOT implement, or from REQUIRED to MANDATORY TO NOT implement.

13. Minor versions may upgrade features from OPTIONAL to RECOMMENDED, or RECOMMENDED to REQUIRED. Also, if a feature was marked as OBSOLESCENT in the prior minor version, it may be upgraded to not be OBSOLESCENT.

14. A client and server that support minor version X SHOULD support minor versions 0 through X-1 as well.

15. Other than where explicitly documented in a minor version's specification document, except for infrastructural changes, a minor version must not introduce REQUIRED new features.

    This rule allows for the introduction of new functionality and forces the use of implementation experience before designating a feature as REQUIRED. On the other hand, some classes of features are infrastructural and have broad effects. Allowing infrastructural features to be RECOMMENDED or OPTIONAL complicates implementation of the minor version.

16. A client MUST NOT attempt to use a stateid, filehandle, or similar returned object from the COMPOUND procedure with minor version X for another COMPOUND procedure with minor version Y, where X != Y.

5. Extensions within Minor Versions

An important goal of this document is to enable extensions to be made to the set of features included in an existing minor version, without the overhead attendant upon the creation of an entirely new minor version.

5.1. Feature Specification Documents

Each such extension will be in the form of a working-group standards-track document which defines a new optional feature. The definition of the new feature may include one or more "feature elements" which add to the existing XDR in ways already discussed in connection with the creation of new minor versions. Other sorts of XDR modification are not allowed. 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 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.

Such an 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:

- o 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.

- o 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.

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 in NFSv4.2, the feature definition documents would each define a single feature.

In addition to a general explanation of the feature in question, the items to be included in such feature definition documents would be:

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

- o Description of any modified operations (corresponding to Section 15 of [NFSv42]).

- o Description of new attributes (corresponding to Section 13 of [NFSv42]).

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

- o 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.

- o 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 (always OPTIONAL), and its relationship to the feature 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 it is restricted to a single feature and expanded in scope to include all feature elements.

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

Addition of features to an existing 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.

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 5.1.1.

- Issues related to messages sent to the client are discussed in Section 5.1.2.

5.1.1. 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 in question 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 the new union arm is not supported by the server.

Given that some existing servers may have XDR parsing implementations that cannot easily accommodate previously unknown operations or switched union arms, clients should carefully determine whether particular features are supported by the server before proceeding to use them and need to be prepared to treat NFS4ERR_BADXDR as indicating non-support of a new operation or switched union arm where server support for a particular feature is being tested.

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 callback 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 he new switched union arm.

5.1.2. 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 callback 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 callbacks, 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.

5.2. 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.

5.2.1. 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:

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

o A table mapping operations and callbacks to the most recent document containing a description of that operation.

o A table mapping attributes to the most recent document containing a description of that attribute.

o 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.

o A table giving for each operation, callback, and attribute and for each feature element in a published extension giving its status OPTIONAL, REQUIRED, RECOMMENDED, 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, and updated to include all published features that are part of the current NFSv4 minor version, at the date of publication.

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.

5.2.2. 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.

5.2.3. 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.

5.2.4. 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 6.1.

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

Extensibility of minor version are governed by the following rules:

o Minor versions zero and one are not extensible. Each has a fixed set of optional features as described in [RFC3530bis] and [RFC5661].

o 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.

o No new feature may be added to a minor version may be made once the specification document 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 the issue. See Section 6 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:

o 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.

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

o 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. I.e., when it is clear that existing client 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.

6. 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 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 structure may not be modified or deleted.

- XDR extensions may be used to correct existing protocol facilities in a manner similar to those used to add additional OPTIONAL features. Such corrections may be done in an otherwise non-extensible minor version, if the working group judges it appropriate.

- When a correction is made to an OPTIONAL feature, the result is similar to a situation in which there are two independent OPTIONAL features. A server may choose to implement either or both.

- When a correction is made to a MANDATORY feature, the situation becomes one in which neither the old nor the new new version of the feature is MANDATORY. Instead, it is MANDATORY that a server support at least one of the two, while each is individually OPTIONAL. Although use of the corrected version is ultimately

better, it is not 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.

- o When a correction is made to a RECOMMENDED feature, the situation is similar to the case of a MANDATORY feature. Neither version is RECOMMENDED, Neither the old nor the new version of the feature is RECOMMENDED. Instead, it is RECOMMENDED that a server support at least one of the two, while each is individually OPTIONAL.

- o 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 6.1.

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.

- o 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.

- o 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.

- o 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.

- o 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.

- o 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 interopt with all client variants.

By using extensions in this manner, the protocol creates 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.

6.1. 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 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 wit 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.

7. Security Considerations

There are no security considerations in this document.

8. IANA Considerations

There are no IANA considerations in this document.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", March 1997.

[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.

[RFC3530bis]
Haynes, T. and D. Noveck, "Network File System (NFS) version 4 Protocol", draft-ietf-nfsv4-rfc3530bis-33 (Work In Progress), April 2014.

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

9.2. Informative References

[NFSv42]   Haynes, T., "NFS Version 4 Minor Version 2", draft-ietf-nfsv4-minorversion2-27 (Work In Progress), September 2014.

Appendix A. Acknowledgments

Appendix B. RFC Editor Notes

[RFC Editor: please remove this section prior to publishing this document as an RFC]

[RFC Editor: prior to publishing this document as an RFC, please replace all occurrences of RFCTBD10 with RFCxxxx where xxxx is the RFC number of this document]

Authors' Addresses

Thomas Haynes
Primary Data, Inc.
4300 El Camino Real Ste 100
Los Altos, CA 94022
USA

Phone: +1 408 215 1519
Email: thomas.haynes@primarydata.com

David Noveck
Dell
300 Innovative Way
Nashua, NH 03062
US

Phone: +1 781 572 8038
Email: dave_noveck@dell.com