Network File System Version 4
Internet-Draft
Obsoletes: 5667 (if approved)
Intended status: Standards Track
Expires: April 1, 2017

C. Lever, Ed.
Oracle

September 28, 2016

# Network File System (NFS) Upper Layer Binding To RPC-Over-RDMA draft-ietf-nfsv4-rfc5667bis-03

## Abstract

This document specifies Upper Layer Bindings of Network File System (NFS) protocol versions to RPC-over-RDMA transports. These bindings are required to enable RPC-based protocols such as NFS to use direct data placement on RPC-over-RDMA transports. This document obsoletes RFC 5667.

## 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 [RFC2119].

## 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 April 1, 2017.

## Copyright Notice

Table of Contents



## 1. Introduction

An RPC-over-RDMA transport, such as defined in [I-D.ietf-nfsv4-rfc5666bis], may employ direct data placement to convey data payloads associated with RPC transactions. Each RPC-over-RDMA transport header conveys lists of memory locations corresponding to XDR data items defined in an Upper Layer Protocol (such as NFS).

To facilitate interoperation, RPC client and server implementations must agree in advance on what XDR data items in which RPC procedures are eligible for direct data placement (DDP). This document contains material required of Upper Layer Bindings, as specified in [I-D.ietf-nfsv4-rfc5666bis], for the following NFS protocol versions:

- NFS Version 2 [RFC1094]
- NFS Version 3 [RFC1813]
- NFS Version 4.0 [RFC7530]
- NFS Version 4.1 [RFC5661]
- NFS Version 4.2 [I-D.ietf-nfsv4-minorversion2]

## 2. Conveying NFS Operations On RPC-Over-RDMA Transports

Definitions of terminology and a general discussion of how RPC-over-RDMA is used to convey RPC transactions can be found in [I-D.ietf-nfsv4-rfc5666bis]. In this section, these general principals are applied to the specifics of the NFS protocol.

### 2.1. Use Of The Read List

The Read list in each RPC-over-RDMA transport header represents a set of memory regions containing DDP-eligible NFS argument data. Large data items, such as the data payload of an NFS version 3 WRITE procedure, are referenced by the Read list. The NFS server pulls such payloads from the client and places them directly into its own memory.

XDR unmarshaling code on the NFS server identifies the correspondence between Read chunks and particular NFS arguments via the chunk Position value encoded in each Read segment.

### 2.2. Use Of The Write List

The Write list in each RPC-over-RDMA transport header represents a set of memory regions that can receive DDP-eligible NFS result data. Large data items, such as the payload of an NFS version 3 READ procedure, are referenced by the Write list. The NFS server pushes such payloads to the client, placing them directly into the client's memory.

Each Write chunk corresponds to a specific XDR data item in an NFS reply. This document specifies how NFS client and server implementations identify the correspondence between Write chunks and XDR results.

#### 2.2.1. Empty Write Chunks

Section 4.4.6.2 of [I-D.ietf-nfsv4-rfc5666bis] defines the concept of unused Write chunks. An unused Write chunk is a Write chunk with either zero segments or where all segments in the Write chunk have zero length. In this document these are referred to as "empty" Write chunks. A "non-empty" Write chunk has at least one segment of non-zero length.

An NFS client might wish an NFS server to return a DDP-eligible result inline. If there is only one DDP-eligible result item in the reply, the NFS client simply specifies an empty Write list to force the NFS server to return that result inline. If there are multiple

DDP-eligible results, the NFS client specifies empty Write chunks for each DDP-eligible data item that it wishes to be returned inline.

An NFS server might encounter an XDR union result where there are arms that have a DDP-eligible result, and arms that do not. If the NFS client has provided a non-empty Write chunk that matches with a DDP-eligible result, but the response does not contain that result, the NFS server MUST return an empty Write chunk in that position in the Write list.

## 2.3. Use Of Long Calls And Replies

Small RPC messages are conveyed using RDMA Send operations which are of limited size. If an NFS request is too large to be conveyed within the NFS server's responder inline threshold, and there are no DDP-eligible data items that can be removed, an NFS client must send the request using a Long Call. The entire NFS request is sent in a special Read chunk called a Position-Zero Read chunk.

If an NFS client predicts that the maximum size of an NFS reply could be too large to be conveyed within it's own responder inline threshold, it provides a Reply chunk in the RPC-over-RDMA transport header conveying the NFS request. The server places the entire NFS reply in the Reply chunk.

These special chunks are described in more detail in [I-D.ietf-nfsv4-rfc5666bis].

## 2.4. Scatter-Gather Considerations

A chunk comprises exactly one XDR data item. Each Read chunk is represented as a list of segments at the same XDR Position. Each Write chunk is represented as an array of segments. An NFS client thus has the flexibility to advertise a set of discontiguous memory regions in which to send or receive a single DDP-eligible XDR data item.

# 3. NFS Versions 2 And 3 Upper Layer Binding

An NFS version 2 or version 3 client MAY send a single Read chunk to supply the opaque file data for an NFS WRITE procedure, or the pathname for an NFS SYMLINK procedure. For these procedures, NFS version 2 or 3 servers MUST ignore Read chunks beyond the first in the Read list. For all other NFS procedures, NFS version 2 or 3 servers MUST ignore Read chunks that have a non-zero value in their Position fields.

Similarly, an NFS version 2 or version 3 client MAY provide a single Write chunk to receive either the opaque file data from an NFS READ procedure, or the pathname from an NFS READLINK procedure. For these procedures, NFS version 2 or 3 servers MUST ignore Write chunks beyond the first in the Write list. For all other NFS procedures, NFS version 2 or 3 servers MUST ignore the Write list.

There are no NFS version 2 or 3 procedures that have DDP-eligible data items in both their Call and Reply. However, when an NFS version 2 or version 3 client sends a Long Call or Reply, it MAY provide a combination of a Read list, a Write list, and/or a Reply chunk in the same RPC-over-RDMA header.

If an NFS version 2 or version 3 client has not provided enough bytes in a Read list to match the size of a DDP-eligible NFS argument data item, or if an NFS version 2 or version 3 client has not provided enough Write list resources to handle an NFS READ or READLINK reply, or if the client has not provided a large enough Reply chunk to convey an NFS reply, the server MUST return one of:

- An RPC-over-RDMA message of type RDMA_ERROR, with the rdma_xid field set to the XID of the matching NFS Call, and the rdma_error field set to ERR_CHUNK; or

- An RPC message (via an RDMA_MSG message) with the xid field set to the XID of the matching NFS Call, the mtype field set to REPLY, the stat field set to MSG_ACCEPTED, and the accept_stat field set to GARBAGE_ARGS.

These replies do not give any indication to NFS version 2 or version 3 clients of whether an NFS version 2 or 3 server has processed the arguments of the RPC Call, or whether the NFS version 2 or 3 server has accessed NFS client memory associated with that RPC.

NFS version 2 or version 3 clients already successfully estimate the maximum reply size of each operation in order to provide an adequate set of buffers to receive each NFS reply. An NFS version 2 or version 3 client provides a Reply chunk when the maximum possible reply size is larger than the client's responder inline threshold.

## 3.1. Auxiliary Protocols

NFS versions 2 and 3 are typically deployed with several other protocols, referred to as "auxiliary" protocols. These are separate RPC protcols which handle operations that are not part of the main NFS protocol. These include the MOUNT and NLM protocols, introduced in an appendix of [RFC1813]; the NSM protocol, described in Chapter 11 of [NSM]; and the NFSACL protocol, which does not have a public

definition. However NFSACL is treated as a de facto standard and there are several interoperating implementations.

RPC-over-RDMA considers these as individual Upper Layer Protocols [I-D.ietf-nfsv4-rfc5666bis]. Therefore to operate on an RPC-over-RDMA transport, an Upper Layer Binding must be provided for each of these.

Typically MOUNT, NLM, and NSM are conveyed via TCP rather than RPC-over-RDMA. Note that only metadata is conveyed in these protocols, thus direct data placement is never necessary, and the size of RPC messages is uniformly small. The maximum size of replies is easily determined by examining the XDR definitions of these protocols.

Implementations that support the NFSACL protocol typically send NFSACL procedures on the same connection as the main NFS protocol. Thus NFSACL does require an Upper Layer Binding.

No data item in this protocol is DDP-eligible. There is no protocol size limit for NFS version 3 ACL objects. The client can have some difficulty ascertaining the size of ACLs to be read from servers. Practically speaking, ACLs are not large (less than 4KB in most cases), but a large Reply chunk may be provided when the client is in doubt. The usual rules apply to the use of Long Messages when the size of an NFSACL RPC exceeds a connection's inline thresholds.

## 4. NFS Version 4 Upper Layer Binding

This specification applies to NFS Version 4.0 [RFC7530], NFS Version 4.1 [RFC5661], and NFS Version 4.2 [I-D.ietf-nfsv4-minorversion2]. It also applies to the callback protocols associated with each of these minor versions defined in the same documents.

### 4.1. DDP-Eligibility

For each WRITE operation in an NFS version 4 COMPOUND procedure, an NFS version 4 client MAY provide a single Read chunk to supply the opaque file data argument. For each CREATE(NF4LNK) operation in an NFS version 4 COMPOUND procedure, An NFS version 4 client MAY provide a single Read chunk to supply the pathname argument.

Similarly, for each READ operation in an NFS version 4 COMPOUND procedure, an NFS version 4 client MAY provide a single Write chunk to receive the opaque file data argument. For each READ_PLUS operation in an NFS version 4 COMPOUND procedure, an NFS version 4 client MAY provide a single Write chunk to receive NFS4_CONTENT_DATA. For each READLINK operation in an NFS version 4 COMPOUND procedure,

an NFS version 4 client MAY provide a single Write chunk to receive the pathname argument.

An NFS version 4 client MUST NOT provide a Read or Write chunk that corresponds with any other XDR data item in any other NFS version 4 operation in an NFS version 4 COMPOUND procedure, or in an NFS version 4 NULL procedure.

It is possible for NFS version 4 COMPOUND procedures to use both the Read list and Write list simultaneously. An NFS version 4 client MAY provide a Read list and a Write list in the same transaction if it is sending a Long Call or Reply.

If an NFS version 4 client has not provided enough bytes in a Read list to match the size of a DDP-eligible NFS argument data item, or if an NFS version 4 client has not provided enough Write list resources to handle a WRITE or READLINK operation, or if the client has not provided a large enough Reply chunk to convey an NFS reply, the server MUST return one of:

- An RPC-over-RDMA message of type RDMA_ERROR, with the rdma_xid field set to the XID of the matching NFS Call, and the rdma_error field set to ERR_CHUNK; or

- An RPC message (via an RDMA_MSG message) with the xid field set to the XID of the matching NFS Call, the stat field set to MSG_ACCEPTED, and the accept_stat field set to GARBAGE_ARGS.

Such error replies are permanent errors, and constitute both completion of the RPC transaction, and a valid server response. It is not necessary for an NFS version 4 server to drop the transport connection in this case.

### 4.1.1. Session-Related Considerations

In most cases, the presence of an NFS session [RFC5661] has no effect on the operation of RPC-over-RDMA. None of the operations introduced to support NFS sessions contain DDP-eligible data items. There is no need to match the number of session slots with the number of available RPC-over-RDMA credits.

However, there are some rare error conditions which require special handling when an NFS session is operating on an RPC-over-RDMA transport. For example, a requester might receive, in response to an RPC request, an RDMA_ERROR message with an rdma_err value of ERR_CHUNK, or an RDMA_MSG containing an RPC_GARBAGEARGS reply. Within RPC-over-RDMA Version One, this class of error can be generated for two different reasons:

- There was an XDR error detected parsing the RPC-over-RDMA headers.

- There was an error sending the response, because, for example, a necessary reply chunk was not provided or the one provided is of insufficient length.

These two situations, which arise only due to incorrect implementations, have different implications with regard to Exactly-Once Semantics. An XDR error in decoding the request precludes the execution of the request on the responder, but failure to send a reply indicates that some or all of the operations were executed.

In both instances, the client SHOULD NOT retry the operation. A retry is liable to result in the same sort of error seen previously. Instead, it is best to consider the operation as completed unsuccessfully and report an error to the consumer who requested the RPC.

In addition, within the error response, the requester does not have the result of the execution of the SEQUENCE operation, which identifies the session, slot, and sequence id for the request which has failed. The xid associated with the request, obtained from the rdma_xid field of the RDMA_ERROR or RDMA_MSG message, must be used to determine the session and slot for the request which failed, and the slot must be properly retired. If this is not done, the slot could be rendered permanently unavailable.

## 4.2. Reply Size Estimation

An NFS version 4 client provides a Reply chunk when the maximum possible reply size is larger than the client's responder inline threshold. NFS version 4 clients already successfully estimate the maximum reply size of most operations in order to provide an adequate set of buffers to receive each NFS reply.

There are certain NFS version 4 data items whose size cannot be estimated by clients reliably, however, because there is no protocol-specified size limit on these structures. These include but are not limited to opaque types, such as:

- The attrlist4 field

- Fields containing ACLs such as fattr4_acl, fattr4_dacl, fattr4_sacl

- Fields in the fs_locations4 and fs_locations_info4 data structures

- Opaque fields which pertain to pNFS layout metadata, such as loc_body, loh_body, da_addr_body, lou_body, lrf_body, fattr_layout_types and fs_layout_types,

In NFS version 4.1 and later minor versions, the csa_fore_chan_attrs argument of the CREATE_SESSION operation contains a ca_maxresponsesize field. The value in this field can be taken as the absolute maximum size of replies generated by a replying NFS version 4 server. This value can be used in cases where it is not possible to estimate a reply size upper bound precisely. In practice, objects such as ACLs, named attributes, layout bodies, and security labels are much smaller than this maximum.

With regard to NFS version 4.0, things are more troublesome. Typically NFS version 4.0 client implementations rely on their own architectural limits to keep reply buffer sizes reasonable. For instance, although the NFS version 4 protocol is capable of conveying a megabyte-sized ACL, nearly all known physical filesystems store ACLs in on-disk containers which are small in size.

### 4.2.1. Managing READ_PLUS Replies

The NFS version 4.2 READ_PLUS operation returns a complex data type [I-D.ietf-nfsv4-minorversion2]. The rpr_contents field in the result of this operation is an array of read_plus_content unions, one arm of which contains an opaque byte stream (d_data).

The size of d_data is limited to the value of the rpa_count field, but the protocol does not bound the number of elements which can be returned in the rpr_contents array. In order to make the size of READ_PLUS replies predictable by NFS version 4.2 clients, the following restrictions are placed on the use of the READ_PLUS operation on RPC-over-RDMA transports:

- An NFS version 4.2 client MUST NOT provide more than one Write chunk for any READ_PLUS operation. When providing a Write chunk for a READ_PLUS operation, an NFS version 4.2 client MUST provide a Write chunk that is either empty (which forces all result data items for this operation to be returned inline) or large enough to receive rpa_count bytes in a single element of the rpr_contents array.

- If the Write chunk provided for a READ_PLUS operation by an NFS version 4.2 client is not empty, an NFS version 4.2 server MUST use that chunk for the first element of the rpr_contents array that has an rpc_data arm.

- An NFS version 4.2 server MUST NOT return more than two elements in the rpr_contents array of any READ_PLUS operation. It returns as much of the requested byte range as it can fit within these two elements. If the NFS version 4.2 server has not asserted rpr_eof in the reply, the NFS version 4.2 client SHOULD send additional READ_PLUS requests for any remaining bytes.

## 4.3. NFS Version 4 COMPOUND Requests

A single NFS version 4 COMPOUND procedure supplies arguments for a sequence of operations, and returns results from that sequence, all in a single round-trip [RFC7530]. An NFS version 4 client MAY construct an NFS version 4 COMPOUND procedure that provides more than one chunk in the Read list or Write list as long as it observes the restrictions in Section 4.1.

An NFS version 4 client provides XDR Position values in each Read chunk to disambiguate which chunk is associated with which argument data item. However NFS version 4 server and client implementations must agree in advance on how to pair Write chunks with returned result data items.

The mechanism specified in Section 5.3.2 of [I-D.ietf-nfsv4-rfc5666bis]) is applied here, with some additional restrictions. In the following list, an "NFS Read" operation refers to any NFS Version 4 operation which has a DDP-eligible result data item (i.e., either a READ, READ_PLUS, or READLINK operation).

- If an NFS version 4 client wishes all DDP-eligible items in an NFS reply to be conveyed inline, it leaves the Write list empty.

- The first chunk in the Write list MUST be used by the first NFS Read operation in an NFS version 4 COMPOUND procedure. The next Write chunk is used by the next NFS Read operation, and so on.

- If an NFS version 4 client has provided a matching non-empty Write chunk, then the corresponding NFS Read operation MUST return its DDP-eligible data item using that chunk.

- If an NFS version 4 client has provided an empty matching Write chunk, then the corresponding NFS Read operation MUST return all of its result data items inline.

- If an NFS Read operation returns a union arm which does not contain a DDP-eligible result, and the NFS version 4 client has provided a matching non-empty Write chunk, an NFS version 4 server MUST return an empty Write chunk in that Write list position.

- If there are more NFS Read operations than Write chunks, then remaining NFS Read operations in an NFS version 4 COMPOUND that have no matching Write chunk MUST return their results inline.

### 4.3.1. NFS Version 4 COMPOUND Example

The following example shows a Write list with three Write chunks, A, B, and C. The NFS version 4 server consumes the provided Write chunks by writing the results of the designated operations in the compound request (READ and READLINK) back to each chunk.

Write list:

```
A --> B --> C
```

NFS version 4 COMPOUND request:

```
PUTFH LOOKUP READ PUTFH LOOKUP READLINK PUTFH LOOKUP READ
              |                 |                  |
              v                 v                  v
              A                 B                  C
```

If the NFS version 4 client does not want to have the READLINK result returned via RDMA, it provides an empty Write chunk for buffer B to indicate that the READLINK result must be returned inline.

## 4.4. NFS Version 4 Callback

The NFS version 4 protocols support server-initiated callbacks to notify clients of events such as recalled delegations.

### 4.4.1. NFS Version 4.0 Callback

NFS version 4.0 implementations typically employ a separate TCP connection to handle callback operations, even when the forward channel uses a RPC-over-RDMA transport. Therefore no Upper Layer Binding for the NFS version 4.0 callback program is provided in this document.

### 4.4.2. NFS Version 4.1 Callback

In NFS version 4.1 and later minor versions, callback operations may appear on the same connection as is used for NFS version 4 forward channel client requests. NFS version 4 clients and servers MUST use the mechanism described in [I-D.ietf-nfsv4-rpcrdma-bidirection] when backchannel operations are conveyed on RPC-over-RDMA transports.

The csa_back_chan_attrs argument of the CREATE_SESSION operation contains a ca_maxresponsesize field. The value in this field can be taken as the absolute maximum size of backchannel replies generated by a replying NFS version 4 client.

There are no DDP-eligible data items in callback protocols associated with NFS version 4.1 or NFS version 4.2. However, some callback requests, such as messages that convey device ID information, may be large, in which case a Long Call or Reply may be appropriate. When the NFS version 4 client reports a backchannel ca_maxresponsesize that is larger than the connection's inline thresholds, the NFS version 4 client can support Long messages (i.e., Read chunks and Reply chunks). Otherwise an NFS version 4 server MUST use Short messages to convey backchannel operations.

See Section 4.1 for a discussion of how an NFS version 4 server handles situations where an NFS version 4 client has provided inadequate RDMA resources to convey a backchannel reply.

## 4.5. Connection Keep-Alive

NFS version 4 client implementations often rely on a transport-layer keep-alive mechanism to detect when an NFS version 4 server has become unresponsive. When an NFS server is no longer responsive, client-side keep-alive terminates the connection, which in turn triggers reconnection and RPC retransmission.

RDMA transports have no keep-alive mechanism. Without a disconnect or new RPC traffic, RDMA transport connections can remain alive long after an NFS server has become unresponsive. Once an NFS client has consumed all available RPC-over-RDMA credits on that transport connection, it will forever await a reply before sending another RPC request.

NFS version 4 clients SHOULD reserve one RPC-over-RDMA credit to use for periodic server or connection health assessment. This credit can be used to drive an RPC request on an otherwise idle connection, triggering either a quick affirmative server response or immediate connection termination.

To prevent lease expiry, NFS version 4 clients should use a lease-extending operation such as RENEW or SEQUENCE, rather than a NULL request, when performing a periodic health assessment.

## 5. Extending NFS Upper Layer Bindings

RPC programs such as NFS are required to have an Upper Layer Binding specification to interoperate on RPC-over-RDMA transports [I-D.ietf-nfsv4-rfc5666bis]. Via standards action, the Upper Layer Binding specified in this document can be extended to cover versions of the NFS version 4 protocol specified after NFS version 4 minor version 2. This includes NFS version 4 extensions that are documented separately from a new minor version.

## 6. IANA Considerations

NFS use of direct data placement introduces a need for an additional NFS port number assignment for networks that share traditional UDP and TCP port spaces with RDMA services. The iWARP [RFC5041] [RFC5040] protocol is such an example (InfiniBand is not).

NFS servers for versions 2 and 3 [RFC1094] [RFC1813] traditionally listen for clients on UDP and TCP port 2049, and additionally, they register these with the portmapper and/or rpcbind [RFC1833] service. However, [RFC7530] requires NFS version 4 servers to listen on TCP port 2049, and they are not required to register.

An NFS version 2 or version 3 server supporting RPC-over-RDMA on such a network and registering itself with the RPC portmapper MAY choose an arbitrary port, or MAY use the alternative well-known port number for its RPC-over-RDMA service. The chosen port MAY be registered with the RPC portmapper under the netid assigned by the requirement in [I-D.ietf-nfsv4-rfc5666bis].

An NFS version 4 server supporting RPC-over-RDMA on such a network MUST use the alternative well-known port number for its RPC-over-RDMA service. Clients SHOULD connect to this well-known port without consulting the RPC portmapper (as for NFS version 4 on TCP transports).

The port number assigned to an NFS service over an RPC-over-RDMA transport is available from the IANA port registry [RFC3232].

## 7. Security Considerations

RPC-over-RDMA supports all RPC security models, including RPCSEC_GSS security and transport-level security [RFC2203]. The choice of RDMA Read and RDMA Write to convey RPC argument and results does not affect this, since it changes only the method of data transfer. Specifically, the requirements of [I-D.ietf-nfsv4-rfc5666bis] ensure that this choice does not introduce new vulnerabilities.

Because this document defines only the binding of the NFS protocols atop [I-D.ietf-nfsv4-rfc5666bis], all relevant security considerations are therefore to be described at that layer.

# 8. References

## 8.1. Normative References

[I-D.ietf-nfsv4-minorversion2]
Haynes, T., "NFS Version 4 Minor Version 2", draft-ietf-nfsv4-minorversion2-41 (work in progress), January 2016.

[I-D.ietf-nfsv4-rfc5666bis]
Lever, C., Simpson, W., and T. Talpey, "Remote Direct Memory Access Transport for Remote Procedure Call, Version One", draft-ietf-nfsv4-rfc5666bis-07 (work in progress), May 2016.

[I-D.ietf-nfsv4-rpcrdma-bidirection]
Lever, C., "Bi-directional Remote Procedure Call On RPC-over-RDMA Transports", draft-ietf-nfsv4-rpcrdma-bidirection-05 (work in progress), June 2016.

[RFC1833] Srinivasan, R., "Binding Protocols for ONC RPC Version 2", RFC 1833, DOI 10.17487/RFC1833, August 1995, <http://www.rfc-editor.org/info/rfc1833>.

[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>.

[RFC2203] Eisler, M., Chiu, A., and L. Ling, "RPCSEC_GSS Protocol Specification", RFC 2203, DOI 10.17487/RFC2203, September 1997, <http://www.rfc-editor.org/info/rfc2203>.

[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>.

## 8.2. Informative References

[NSM] The Open Group, "Protocols for Interworking: XNFS, Version 3W", February 1998.

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

[RFC1813] Callaghan, B., Pawlowski, B., and P. Staubach, "NFS Version 3 Protocol Specification", RFC 1813, DOI 10.17487/RFC1813, June 1995, <http://www.rfc-editor.org/info/rfc1813>.

[RFC3232] Reynolds, J., Ed., "Assigned Numbers: RFC 1700 is Replaced by an On-line Database", RFC 3232, DOI 10.17487/RFC3232, January 2002, <http://www.rfc-editor.org/info/rfc3232>.

[RFC5040] Recio, R., Metzler, B., Culley, P., Hilland, J., and D. Garcia, "A Remote Direct Memory Access Protocol Specification", RFC 5040, DOI 10.17487/RFC5040, October 2007, <http://www.rfc-editor.org/info/rfc5040>.

[RFC5041] Shah, H., Pinkerton, J., Recio, R., and P. Culley, "Direct Data Placement over Reliable Transports", RFC 5041, DOI 10.17487/RFC5041, October 2007, <http://www.rfc-editor.org/info/rfc5041>.

[RFC5667] Talpey, T. and B. Callaghan, "Network File System (NFS) Direct Data Placement", RFC 5667, DOI 10.17487/RFC5667, January 2010, <http://www.rfc-editor.org/info/rfc5667>.

## Appendix A. Changes Since RFC 5667

Corrections and updates made necessary by new language in [I-D.ietf-nfsv4-rfc5666bis] have been introduced. For example, references to deprecated features of RPC-over-RDMA Version One, such as RDMA_MSGP, and the use of the Read list for handling RPC replies, have been removed. The term "mapping" has been replaced with the term "binding" or "Upper Layer Binding" throughout the document. Some material that duplicates what is in [I-D.ietf-nfsv4-rfc5666bis] has been deleted.

Material required by [I-D.ietf-nfsv4-rfc5666bis] for Upper Layer Bindings that was not present in [RFC5667] has been added, including discussion of how each NFS version properly estimates the maximum size of RPC replies.

Technical corrections have been made. For example, the mention of 12KB and 36KB inline thresholds have been removed. The reference to a non-existant NFS version 4 SYMLINK operation has been replaced with NFS version 4 CREATE(NF4LNK).

The discussion of NFS version 4 COMPOUND handling has been completed. Some changes were made to the algorithm for matching DDP-eligible results to Write chunks.

The following additional improvements have been made, relative to [RFC5667]:

- An explicit discussion of NFS version 4.0 and NFS version 4.1 backchannel operation has replaced the previous treatment of callback operations.
- A binding for NFS version 4.2 has been added that includes discussion of new data-bearing operations like READ_PLUS.
- A section suggesting a mechanism for periodically assessing connection health has been introduced.
- Language inconsistent with or contradictory to [I-D.ietf-nfsv4-rfc5666bis] has been removed from Sections 2 and 3, and both Sections have been combined into Section 2 in the present document.
- Ambiguous or erroneous uses of RFC2119 terms have been corrected.
- References to obsolete RFCs have been updated.
- An IANA Considerations Section has replaced the "Port Usage Considerations" Section.
- Code excerpts have been removed, and figures have been modernized.

## Appendix B. Acknowledgments

The author gratefully acknowledges the work of Brent Callaghan and Tom Talpey on the original NFS Direct Data Placement specification [RFC5667]. The author also wishes to thank Bill Baker and Greg Marsden for their support of this work.

Dave Noveck provided excellent review, constructive suggestions, and consistent navigational guidance throughout the process of drafting this document. Dave also contributed the text of Section 4.1.1.

Thanks to Karen Deitke for her sharp observations about idempotency, and the clarity of the discussion of NFS COMPOUNDs.

Special thanks go to Transport Area Director Spencer Dawkins, nfsv4 Working Group Chair Spencer Shepler, and nfsv4 Working Group Secretary Thomas Haynes for their support.

## Author's Address

Charles Lever (editor)
Oracle Corporation
1015 Granger Avenue
Ann Arbor, MI 48104
USA

Phone: +1 734 274 2396
Email: chuck.lever@oracle.com