Network File System Version 4 C. Lever
Internet-Draft Oracle
Intended status: Informational September 9, 2016
Expires: March 13, 2017

# Using Remote Invalidation With RPC-Over-RDMA Transport Protocols
# draft-cel-nfsv4-reminv-design-03

Abstract

Remote Invalidation relieves requesters/initiators of some of the burden of preparing memory to be accessed remotely, thus reducing the latency of transactions that require the use of explicit RDMA operations. This document considers how to introduce Remote Invalidation to RPC-over-RDMA transport protocols.

Table of Contents



## 1. Introduction

Like other RDMA-enabled storage protocols, RPC-over-RDMA Version Two [I-D.cel-nfsv4-rpcrdma-version-two] employs a Read-Write transfer model when using explicit RDMA operations to transfer data. This means an RPC-over-RDMA requester exposes regions of its memory to an RPC-over-RDMA responder, which then uses RDMA Read and Write operations to transfer bulk data payloads.

In preparation for a bulk data transfer, a requester asks its RNIC to assign a steering tag, or STag, to a region of memory containing the data to be moved. At this time, access rights are granted that allow the RNIC to access or update that memory on behalf of a remote peer. This act is referred to as "memory registration." The RNIC uses this STag to steer data to and from the registered memory region.

When data movement is complete, each STag is dissociated from its memory region. This act is referred to as "memory invalidation." It prevents further responder access to that memory region by revoking its remote access rights. Invalidation should be done before RPC applications on the requester are allowed access to memory that was involved in an explicit RDMA operation.

Remote Invalidation is a technique by which an RDMA peer can request that a remote RNIC invalidate an STag associated with memory on that remote peer [RFC5042]. An RDMA consumer requests Remote Invalidation by posting an RDMA Send With Invalidate Work Request in place of an RDMA Send Work Request. RDMA Send With Invalidate is similar to RDMA Send, but takes one additional argument: a single STag to be invalidated by the RNIC that receives the sent message. An RDMA Send message is transmitted with additional header information that conveys the STag that is to be invalidated [RFC5040].

The benefit of Remote Invalidation is that an extra Work Request, context switch, and interrupt to perform memory invalidation are not

required by the requester as part of handling the completion of an RPC transaction. STag invalidation begins before the Receive completes, thus invalidation is started (and completes) sooner. The upshot is faster completion of RPC transactions that involve registered memory.

This mechanism has the most impact when explicit RDMA operations are needed to move moderate amounts of data. Invalidation latency is quite small compared to the time it takes to convey a large payload with an explicit RDMA operation. Small RPCs are already conveyed entirely via RDMA Send, thus Remote Invalidation is unnecessary for them. When the time it takes to invalidate a memory region is on the same order as the time it takes to move the contents of that region, Remote Invalidation has its greatest impact.

Remote Invalidation confers benefits similar to the benefits of increasing the size of Send and Receive buffers. However, Remote Invalidation does not incur the cost of maintaining a pool of large Receive buffers on either the requester or responder. Moderate-sized RPC payloads can be transferred without the usual costs of memory registration. Requesters can rely on RDMA Write to structure their Receive buffers without introducing additional latency.

There are some downsides, however. Remote Invalidation is not available on all RNIC devices. And, Remote Invalidation does not address the extra latency of using RDMA Read. This extra latency can be eliminated using a large inline threshold for transmitting RPC Calls.

The purpose of this document is to explore generally how Remote Invalidation can be introduced into the RPC-over-RDMA transport protocol. The primary design considerations for the transport protocol are to provide a mechanism to indicate when Remote Invalidation can be used by the transport, and to provide selection criteria for choosing which STag to invalidate remotely. Elements of the XDR definition of the RPC-over-RDMA protocol must be altered to some degree, depending on desired flexibility of operation, invasiveness of XDR changes, and broadness of hardware support.

## 1.1. 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].

## 2. General Requirements

### 2.1. Memory Management Extensions

Remote Invalidation was not available in the original RDMA Verbs API. New verbs API objects were specified that include operations that enable Remote Invalidation, now described in [IB]. The Verbs API provides a capabilities flag, MEM_MGT_EXTENSIONS, that indicates that an RNIC can provide the new APIs and objects.

Only an STag that was registered using the FRWR mechanism, also only available with MEM_MGT_EXTENSIONS, may be invalidated remotely [RFC5040].

RDMA Send With Invalidate is available only with MEM_MGT_EXTENSIONS.

### 2.2. Registration Types

For the purposes of this discussion, there are two classes of STags. Dynamically-registered STags are used in a single RPC, then invalidated. Persistently-registered STags live longer than one RPC. They are typically registered for the life of an RPC-over-RDMA connection, and sometimes even longer.

In RPC-over-RDMA Version One, a requester may provide more than one STag in the chunk lists of an RPC. It may provide any combination of the following registration types in one RPC, any combination of these in a series of RPCs on the same connection, or it may use some other registration model.

Examples of persistently-registered STags include:

- The device's reserved DMA rkey
- An STag registered for a connection that doesn't change from RPC to RPC (for a utility buffer, say)
- An STag registered for a fixed memory region that is updated after each time it is advertised
- An STag covering a large single region that is utilized in small segments by many RPCs

Examples of dynamically-registered STags include:

- An STag registered for a single RPC transaction using a non-FRWR mechanism, then invalidated when the RPC is retired

- An STag registered for a single RPC transaction using the FRWR mechanism, then invalidated when the RPC is retired

Among these examples, only dynamically-registered STags using the FRWR mechanism may be invalidated remotely.

## 2.3. Selecting STags To Invalidate Remotely

Remote Invalidation protocol mechanisms come in different styles:

Fixed Protocol
: The choice of which STag to invalidate remotely is fixed in the protocol specification.

Responder's Choice
: The responder chooses an STag to invalidate remotely from among all the STags in incoming requests.

Requester's Choice
: The requester chooses one or more STags that may be invalidated remotely, indicating its choices in each request. The responder chooses an STag to invalidate remotely from among the requester's picks.

Outside of being told explicitly by the requester, there is no mechanism by which a responder can determine how a requester-provided STag was registered. Thus a requester that mixes persistently- and dynamically-registered STags in one RPC, or mixes them across RPCs on the same connection, cannot tolerate Responder's Choice.

## 2.4. Future Enhancements

There are two related enhancements that further reduce the effort needed to invalidate STags associated with complex RPCs:

- The ability for one registered STag to represent a list of memory regions that are not contiguous

- The ability to specify more than one remote STag in a single Work Request to be remotely invalidated

At this time, the first mechanism has been implemented in at least one RNIC on the market. The second is speculative.

Given support for registering non-contiguous memory regions with one STag, when an RPC-over-RDMA requester constructs an RPC that has both a Read list and a Write list, the requester has a choice:

- The requester can register a separate STag for each access mode (one STag for memory regions needing read access, and one STag for those needing write access) to provide good data security

- The requester can register a single STag with read and write access enabled for the whole set of memory regions, to allow RDMA Send With Invalidate to work optimally

Having the ability to remotely invalidate multiple STags at once enables the combination of optimal performance and optimal security.

## 3. Remote Invalidation In Operation

When requester memory is registered for remote access, an RPC-over-RDMA implementation could use Remote Invalidation by following these steps:

1. The requester DMA-maps a memory region that will participate in an RPC transaction, then registers an STag for that region.

2. The requester transmits the RPC Call, which also conveys the STag, to the responder.

3. The responder processes the RPC transaction. The peer RNICs use the STag to move RPC arguments and/or results.

4. The responder transmits the RPC Reply using an RDMA Send With Invalidate Work Request, setting the Work Request's inv_handle field to the value of the STag.

5. A Receive Work Request completes on the requester, carrying this RPC reply, and reporting the invalidated STag.

6. The requester skips invalidation of the STag, then DMA-unmaps the memory region associated with the STag.

The requester no longer needs to invalidate the STag involved with this RPC. However, there are additional details that must be resolved before the use of Remote Invalidation can commence.

### 3.1. Determining Remote Invalidation Support Status

An RDMA consumer (an Upper Layer Protocol implementation) that does not support Remote Invalidation might not tolerate the use of RDMA Send With Invalidate by the transport layer. Such a requester performs Local Invalidation on STags that already happen to be invalid, and in some cases this can result in protection errors or other issues.

Thus, to avoid spurious connection termination, a responder must not post an RDMA Send With Invalidate Work Request unless it is sure the following three conditions are met:

- The requester's RNIC is prepared to receive the additional header information associated with Remote Invalidation
- The requester has used FRWR to register STags it wants invalidated remotely
- The requester is prepared to recognize remotely invalidated STags and thus avoid invalidating them a second time

When all three of these conditions are true, a requester can report positive Remote Invalidation support status to responders using an Upper Layer Protocol mechanism. When a responder does not know the requester's Remote Invalidation support status, it cannot use Remote Invalidation without endangering the connection.

## 3.2. Selection Of Which STag To Invalidate Remotely

The RDMA Send With Invalidate Work Request invalidates only one STag. RPC-over-RDMA requesters may register more than one STag to handle the movement of payloads for a single RPC. Either the client will have to specify which STag may be remotely invalidated, the protocol will have to specify a fixed way to select which STag to invalidate, or the responder will have to choose arbitrarily which STag to remotely invalidate.

In some circumstances, requesters may wish to utilize STags during transactions that are registered using a mechanism that does not tolerate Remote Invalidation. For example, an STag that is the requester's local DMA rkey should never be invalidated remotely. If a responder attempts to invalidate a such an STag, the result is undefined, but the connection can be terminated or other failures can occur.

Even with Remote Invalidation enabled, requesters remain responsible for ensuring all STags are invalid before RPC transactions complete. To avoid leaving STags registered, a requester must be prepared for the responder or the requester's own RNIC to have not invalidated any of an RPC's STags. When there are multiple STags associated with a single RPC, a requester must be prepared for any of the STags to have been remotely invalidated, or none of them.

### 3.3. Backward-Direction Operation

As of this writing, no current RPC-over-RDMA implementation supports direct data placement in the backward-direction. However, existing protocol specifications do not forbid it [I-D.ietf-nfsv4-rfc5666bis] [I-D.ietf-nfsv4-rpcrdma-bidirection] [I-D.cel-nfsv4-rpcrdma-version-two].

When chunks are present in a backward-direction RPC request, Remote Invalidation allows the responder to trigger invalidation of a requester's STags as part of sending a reply, the same as in the forward direction.

However, in the backward direction, the server acts as the requester, and the client is the responder. The server's RNIC, therefore, must support receiving an IETH, and the server must have registered the STags with FRWR. Thus the server must indicate its Remote Invalidation support status to the client (the opposite of forward direction Remote Invalidation).

## 4. Protocol Elements

In this section, a number of abstract protocol variations are considered. These vary in functionality and invasiveness. Some may be appropriate to use in combination.

### 4.1. Per Protocol Version Remote Invalidation

#### 4.1.1. Description

When a higher protocol version number is negotiated, Remote Invalidation is always enabled. This new protocol version would then be usable only with RNICs that support Remote Invalidation. Both peers assume that Remote Invalidation may be used in either direction.

#### 4.1.2. Similar Existing Implementations

SMB Direct [MS-SMBD]

#### 4.1.3. Advantages

No XDR changes or protocol extensions are required.

Backward-direction use of Remote Invalidation is automatically supported.

#### 4.1.4. Disadvantages

The requester is not in control of which STags in an RPC may be invalidated. Thus, a requester must not advertise STags which must never be invalidated.

Other features and benefits of the new protocol version would not be available when an implementation employs an RNIC that does not support Remote Invalidation. In particular, RNICs that do not support MEM_MGT_EXTENTIONS (i.e., FRWR) could not use the new protocol version.

An extension or addition protocol version bump is required to indicate support for transport-level mechanisms that can invalidate multiple STags at once.

### 4.2. Per Connection Remote Invalidation

#### 4.2.1. Description

At connection initiation time, messages are exchanged that indicate each peer's Remote Invalidation support status. Without these messages, peers assume Remote Invalidation is not supported.

#### 4.2.2. Similar Existing Implementations

iSER [RFC7145]. Information is exchanged in RDMA-CM connection requests to report an implementation's Remote Invalidation support status.

#### 4.2.3. Advantages

No changes to the base protocol XDR are required.

#### 4.2.4. Disadvantages

Out-of-band messages are required to establish support status.

The requester is not in control of which STags in an RPC may be invalidated. Thus, a requester must not advertise STags which must never be invalidated.

To support backward-direction operation, the server must separately indicate that it supports Remote Invalidation.

To enable support for multiple STag invalidation, this negotiation protocol would have to be extended again to indicate when mechanisms

other than RDMA Send With Invalidate are supported by the requester's RNIC.

## 4.3. Fixed Protocol Remote Invalidation

### 4.3.1. Description

No new field is introduced to the transport header. Protocol specification determines how the responder chooses which STag is to be invalidated remotely. Some other means is used to determine whether Remote Invalidation can be used or not.

### 4.3.2. Similar Existing Implementations

iSER [RFC7145]. Two STags fields appear in each request: one advertises Read data and one advertises Write data. When only one STag is used in the request, it may be invalidated remotely. One both STags are used, only the Read STag may be invalidated remotely.

### 4.3.3. Advantages

No changes to the base protocol XDR are required.

### 4.3.4. Disadvantages

Out-of-band messages are required to establish support status.

The requester is not in control of which STags in an RPC may be invalidated. Thus, a requester must not advertise STags which must never be invalidated.

This mechanism may not work well for transport protocols that allow multiple read and write STags.

## 4.4. Per RPC Remote Invalidation (Single STag)

### 4.4.1. Description

A field is added to the transport header that contains an STag which may be invalidated by the responder. A special value can be chosen to mean "no STag may be invalidated" for use by requesters that have no support for Remote Invalidation.

### 4.4.2. Similar Existing Implementations

None.

#### 4.4.3. Advantages

A requester may advertise STags that cannot be invalidated remotely, as long as they are never marked as "may invalidate."

No out-of-band support status negotiation is needed.

Backward-direction RPCs can each indicate whether a backward-direction requester desires or does not support Remote Invalidation.

The responder needs no special logic or assumptions to choose the STag to invalidate remotely.

#### 4.4.4. Disadvantages

Either the base RPC-over-RDMA header XDR definition is altered, or a protocol extension is required.

Requesters transmit a little extra data per RPC, making RPC-over-RDMA messages slightly more costly to send and parse.

This mechanism cannot support the remote invalidation of multiple STags at once.

### 4.5. Per RPC Remote Invalidation (Multiple STags)

#### 4.5.1. Description

A new data structure is added to the transport header that indicate which STags which may be invalidated by the responder.

This information might appear as a new field in the RDMA segment data structure, as each segment has its own STag field. The field indicates whether or not that STag may be invalidated by the responder. Perhaps that field is a boolean, though in XDR, a boolean is a full 32 bits.

Or, this information could appear in the header as an array of STags, to reduce the amount of extra data contained in the RPC-over-RDMA header. Zero array elements means the requester does not support Remote Invalidation.

#### 4.5.2. Similar Existing Implementations

NVMe/Fabrics [NVME]. Each STag in a request has an associated bit flag that indicates whether the responder is allowed to invalidate it remotely.

#### 4.5.3. Advantages

A requester may advertise STags that cannot be invalidated remotely, as long as they are never marked as "may invalidate."

The mechanism allows a requester to request either invalidation of multiple STags at once, or to choose one STag to invalidate remotely.

No out-of-band support status negotiation is needed.

Each backward-direction RPC can indicate whether a backward-direction requester desires or does not support Remote Invalidation.

The responder needs no special logic or assumptions to choose the STag to invalidate remotely.

#### 4.5.4. Disadvantages

The RPC-over-RDMA header XDR definition is possibly extensively altered.

Requesters transmit extra data per RPC. However, it is limited to only one or two 32-bit words in most cases.

### 4.6. Inter-RPC Remote Invalidation

#### 4.6.1. Description

As a subfeature of support for Remote Invalidation, it is possible that a responder can remotely invalidate an STag (using RDMA Send With Invalidate) that refers to registered memory being used in the Read chunk of a different RPC. Such Remote Invalidation would be requested only after the RDMA Read has already been completed.

This can be useful when a responder is replying to an RPC via an inline message, but notices there are other RPC replies pending that have multiple STags, some of which are Read chunks.

#### 4.6.2. Similar Existing Implementations

None

#### 4.6.3. Advantages

This is one way to enable remote invalidation of multiple STags per RPC, using only RDMA Send With Invalidate.

#### 4.6.4. Disadvantages

Additional requester and responder complexity would be required to keep track of STags.

## 5. Recommendations

### 5.1. General Considerations

When constructing a protocol to support Remote Invalidation, one of these designs, or some combination of them, can be chosen.

In no particular order, the design priorities are:

- Do not prevent the efficient operation of RNICs that do not handle RDMA Send With Invalidate
- Introduce as little impact on header XDR and header length as possible, to keep collateral performance impact low
- Enable support for Remote Invalidation when explicit RDMA is used in backward-direction RPCs.

An important question is whether the base RPC-over-RDMA Version Two protocol should support Remote Invalidation, whether Remote Invalidation support should be carried entirely on the shoulders of protocol extensions, or whether some combination of the two is best.

Upper Layer Protocols will likely always be responsible for some degree of signaling Remote Invalidation capabilities, as long as innovation continues at the transport layer (e.g., new RDMA operations that enable Remote Invalidation). Future hardware capabilities are perpetually hazy, limiting the ability to design long-lived protocol support for them. Lastly, it is difficult to estimate how long the industry must continue to support less capable devices.

### 5.2. Analysis And Discussion

All things being equal, making no changes to the base XDR definition has great appeal. If the mechanism in Section 4.2 can be broadly effective at enabling Remote Invalidation in the current set of RPC-over-RDMA implementations, it would be the proper choice.

Unfortunately, among current RPC-over-RDMA client implementations, there is one client that can immediately use a per-connection style protocol, and one that can use only a per-RPC style protocol such as

Section 4.4. A third known client resides in user space and is thus incapable of using the FRWR registration mechanism.

Because there is a wide latitude of implementation choice already allowed by the RPC-over-RDMA transport protocol, the author's preference is to implement Section 4.4. The target STag can be added to the rpcrdma2_chunk_lists data structure as a single field. No further changes or extensions are needed.

In the longer term, the requester appears to be in the better position to determine which STag may be invalidated remotely. With this mechanism, the requester can choose based on which STags may be invalidated remotely, or may use criteria based on the strengths of its RNIC. For instance, choosing the largest registered memory region might be beneficial in some cases.

Allowing the responder to select from among several choices does not seem to bring additional value, and burdens the responder with additional header parsing costs for each chunk-bearing RPC reply.

Furthermore, the ability to request Remote Invalidation of multiple STags in a single Work Request appears to be somewhat distant. It would require additional Upper Layer Protocol mechanisms to distinguish the new mechanism from using RDMA Send With Invalidate, which we are not in a position to design today. Thus it does not seem worth the extra implementation and protocol complexity of having the requester provide a list of STags for the responder to choose from.

As an alternative to modifying the XDR definition for the RDMA_MSG and RDMA_NOMSG message types, a new RDMA message type could be introduced in RPC-over-RDMA Version Two that provides similar functionality to RDMA_MSG and RDMA_NOMSG but adds one or more new fields. This has the advantage of leaving the Version One-compatible parts of the Version Two XDR definition unchanged. It is an open question whether this introduces more complexity to existing implementations than adding new fields to RDMA_MSG and RDMA_NOMSG. However, this approach is similar to the introduction of READ_PLUS in the specification of NFSv4.2 [I-D.ietf-nfsv4-minorversion2].

Allowing the feature described in Section 4.6 is likely to increase the complexity of responder and especially requester implementations, as they would have to remember invalidated STags independently of RPC completions. Because it does not require any XDR changes, it could easily be enabled in a future protocol extension. The author's preference is to forbid this behavior in the initial specification, but allow for a future extension to introduce it.

### 5.3. Example Remote Invalidation Protocol

As an example of how to proceed, the simplest approach would replace struct rpcrdma2_chunk_lists (as defined in [I-D.cel-nfsv4-rpcrdma-version-two]) with the following:

<CODE BEGINS>

```
 struct rpcrdma2_chunk_lists {
     enum msg_type               rdma_direction;
     u32                         rdma_inv_handle;
     struct rpcrdma2_read_list   *rdma_reads;
     struct rpcrdma2_write_list  *rdma_writes;
     struct rpcrdma2_write_chunk *rdma_reply;
 };
```

<CODE ENDS>

The following language describes how to utilize the new field:

> The requester sets the value of the rdma_inv_handle field to the value of any one of the rdma_handle fields in the RPC-over-RDMA header of the RPC call that may be invalidated remotely. If the RPC-over-RDMA header of the RPC call contains no rdma_handles that may be invalidated remotely, the requester MUST set the value of the rdma_inv_handle field to zero. The requester MUST NOT set the value of the rdma_inv_handle field to the value of an rdma_handle that cannot be invalidated remotely.
>
> As part of forming the RPC-over-RDMA header for the reply, the responder copies the value of the rdma_inv_handle field from the RPC-over-RDMA header of the matching RPC call. If the rdma_inv_handle field in the RPC-over-RDMA header of an RPC call contains zero, the responder MUST NOT use RDMA Send With Invalidate to transmit the matching RPC reply. Otherwise, the responder SHOULD use RDMA Send With Invalidate to transmit the reply to this RPC, specifying the value in the RPC-over-RDMA header's rdma_inv_handle field as the Work Request's inv_rkey. The responder MUST NOT specify any other value in the Work Request's inv_rkey field.

## 6. IANA Considerations

There are no IANA considerations for this document.

## 7. Security Considerations

Remote Invalidation metadata is conveyed in the clear in RPC-over-RDMA headers. This does not expose any new information to attackers.

A man-in-the-middle can alter Remote Invalidation metadata while it is in transit. Requesters are prepared to handle the case where responders have not invalidated any STags associated with an RPC. An attacker can cause other STags in flight to be invalidated before the responder is finished with the associated memory. Or an attacker can replace the "to-be invalidated" STag with an STag in the same RPC that should not be invalidated remotely. Any of these might cause loss of connection, or other failures.

A connection relationship is required to exist between a requester and a responder. The requester's RNIC has associated a Protection Domain with that connection. The STag on the requester to be invalidated is associated with that Protection Domain. This protects against arbitrary invalidation of STags by network nodes not part of the connection.

Further discussion appears in [RFC5042].

## 8. Acknowledgments

The author wishes to thank Sagi Grimberg, Christoph Hellwig, Dave Noveck, and Tom Talpey. Special thanks go to nfsv4 Working Group Chair Spencer Shepler and nfsv4 Working Group Secretary Thomas Haynes for their support.

## 9. References

### 9.1. Normative References

[I-D.cel-nfsv4-rpcrdma-version-two]
Lever, C. and D. Noveck, "RPC-over-RDMA Version Two Protocol", draft-cel-nfsv4-rpcrdma-version-two-01 (work in progress), June 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.

[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>.

[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>.

[RFC5042] Pinkerton, J. and E. Deleganes, "Direct Data Placement Protocol (DDP) / Remote Direct Memory Access Protocol (RDMAP) Security", RFC 5042, DOI 10.17487/RFC5042, October 2007, <http://www.rfc-editor.org/info/rfc5042>.

[RFC7145] Ko, M. and A. Nezhinsky, "Internet Small Computer System Interface (iSCSI) Extensions for the Remote Direct Memory Access (RDMA) Specification", RFC 7145, DOI 10.17487/RFC7145, April 2014, <http://www.rfc-editor.org/info/rfc7145>.

## 9.2. Informative References

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

[IB] InfiniBand Trade Association, "InfiniBand Architecture Specifications", <http://www.infinibandta.org>.

[MS-SMBD] Microsoft Corporation, "SMB Remote Direct Memory Access (RDMA) Transport Protocol Specification", July 2016.

[NVME] NVM Express, Inc., "NVM Express Revision 1.2.1", July 2016.

Author's Address

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

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