NVM Express®

RDMA Transport Specification

Revision 1.1

August 5th, 2024

Please send comments to info@nvmexpress.org

NVM Express® RDMA Transport Specification, Revision 1.1 is available for download at https://nvmexpress.org. The NVM Express® RDMA Transport Specification, Revision 1.1 incorporates NVM Express® RDMA Transport Specification Revision 1.0, ratified on June 2 2021, TP4129, TP8012, TP6036, ECN 001, ECN105, ECN110, ECN116, ECN117 and ECN120 (refer to https://nvmexpress.org/changes-in-nvme-revision-2-1 for details).

SPECIFICATION DISCLAIMER

LEGAL NOTICE:

This NVM Express RDMA Transport Specification, Revision 1.1 is proprietary to the NVM Express, Inc. (also referred to as "Company") and/or its successors and assigns.

NOTICE TO USERS WHO ARE NVM EXPRESS, INC. MEMBERS: Members of NVM Express, Inc. have the right to use and implement this NVM Express RDMA Transport Specification, Revision 1.1 subject, however, to the Member's continued compliance with the Company's Intellectual Property Policy and Bylaws and the Member's Participation Agreement.

NOTICE TO NON-MEMBERS OF NVM EXPRESS, INC.: If you are not a Member of NVM Express, Inc. and you have obtained a copy of this document, you only have a right to review this document or make reference to or cite this document. Any such references or citations to this document must acknowledge NVM Express, Inc. copyright ownership of this document. The proper copyright citation or reference is as follows: **"© 2008 to 2024 NVM Express, Inc. ALL RIGHTS RESERVED."** When making any such citations or references to this document you are not permitted to revise, alter, modify, make any derivatives of, or otherwise amend the referenced portion of this document in any way without the prior express written permission of NVM Express, Inc. Nothing contained in this document shall be deemed as granting you any kind of license to implement or use this document or the specification described therein, or any of its contents, either expressly or impliedly, or to any intellectual property owned or controlled by NVM Express, Inc., including, without limitation, any trademarks of NVM Express, Inc.

LEGAL DISCLAIMER:

THIS DOCUMENT AND THE INFORMATION CONTAINED HEREIN IS PROVIDED ON AN "**AS IS**" BASIS. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, NVM EXPRESS, INC. (ALONG WITH THE CONTRIBUTORS TO THIS DOCUMENT) HEREBY DISCLAIM ALL REPRESENTATIONS, WARRANTIES AND/OR COVENANTS, EITHER EXPRESS OR IMPLIED, STATUTORY OR AT COMMON LAW, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE, VALIDITY, AND/OR NONINFRINGEMENT.

All product names, trademarks, registered trademarks, and/or servicemarks may be claimed as the property of their respective owners.

The NVM Express® design mark is a registered trademark of NVM Express, Inc.
PCIe® is a registered trademark of PCI-SIG.

NVM Express Workgroup
c/o VTM, Inc.
3855 SW 153rd Drive
Beaverton, OR 97003 USA
info@nvmexpress.org

Table of Contents

Table of Figures

1 Introduction

1.1 Overview

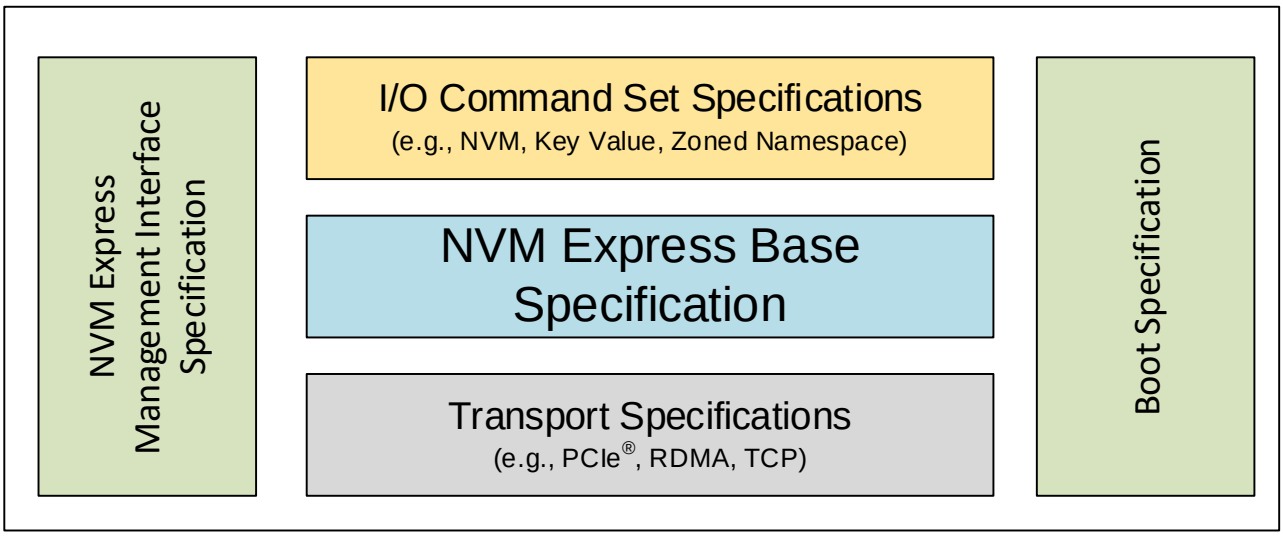
NVM Express® (NVMe®) Base Specification defines an interface for host software to communicate with a non-volatile memory subsystem (NVM subsystem) over a variety of memory-based transports and message-based transports.

This document defines mappings of extensions defined in the NVM Express Base Specification to a specific NVMe Transport: RDMA.

1.2 Scope

Figure 1 shows the relationship of the NVM Express® RDMA Transport Specification to other specifications within the NVMe Family of Specifications.

Figure 1: NVMe Family of Specifications



This specification supplements the NVM Express Base Specification. This specification defines additional data structures, features, log pages, commands, and/or status values. This specification also defines extensions to existing data structures, features, log pages, commands, and/or status values. This specification defines requirements and behaviors that are specific to the RDMA transport. Functionality that is applicable generally to NVMe or that is applicable across multiple NVMe transports is defined in the NVM Express Base specification.

If a conflict arises among requirements defined in different specifications, then a lower-numbered specification in the following list shall take precedence over a higher-numbered specification:

1. Non-NVMe specifications
2. NVM Express Base Specification
3. NVMe transport specifications
4. NVMe I/O command set specifications
5. NVM Express Management Interface Specification
6. NVM Express® Boot Specification

1.3 Conventions

This specification conforms to the Conventions section, Keywords section and the Byte, Word, and Dword Relationships section in the NVM Express Base Specification.

1.4 Definitions

1.4.1 direct data placement

The use of RDMA_READ or RDMA_WRITE to place data exchanged over the RDMA fabric directly into a host or an NVM subsystem memory buffer as specified in the command.

1.4.2 Host Memory Buffer Address

The RDMA Memory Key, byte offset, and byte length identify a host memory buffer within an RDMA Memory Region or Memory Window.

1.4.3 in-order delivery

The use of RDMA_SEND to deliver capsules over the RDMA fabric in the same order that the capsules were submitted to the RDMA transport for transmission for a given Submission or Completion Queue.

1.4.4 Inbound RDMA Read Queue Depth (IRD)

The maximum number of incoming outstanding RDMA_READ Requests that the RDMA-Capable Controller is able to handle on a particular RDMA-Capable Protocol Stream at the Data Source. Refer to InfiniBand™ Architecture Specification Volume 1.

1.4.5 InfiniBand™ R_Key

Remote Memory Region or Window in InfiniBand™ RDMA implementations.

1.4.6 InfiniBand™ RDMA

InfiniBand™ Trade Association definition of RDMA. Refer to www.infinibandta.org.

1.4.7 iWARP RDMA

IETF standard definition of RDMA. Refer to https://tools.ietf.org/html/rfc5040.

1.4.8 iWARP STag

Term used to describe a local or remote Memory Region or Window in iWARP RDMA implementations.

1.4.9 RDMA Memory Key (RKEY)

Component of the Host Memory Buffer Address that associates a host buffer with an RDMA Memory Region or Memory Window. For the RDMA transport, this is either an iWARP STag or InfiniBand™ R_KEY.

1.4.10 RDMA Memory Region

A range of host memory that has been registered with a host-resident RDMA device.

1.4.11 RDMA Memory Window

A range of host memory residing within a registered RDMA Memory Region.

1.4.12 RDMA NIC (RNIC)

RDMA enabled network interface card (i.e., network adapter).

1.4.13 RDMA Private Data

Data that is opaque to the communication management protocol, passed from the sender to the recipient. Refer to the term “private data” in InfiniBand™ Architecture Specification Volume 1.

1.4.14 RDMA Queue Pair (QP)

RDMA communication endpoint that consists of a Send Queue and Receive Queue.

1.4.15 RDMA Verbs

Common functional definition and implementation of the RDMA operational programming model between applications and RDMA providers. Applications are the consumers of RDMA operations and RDMA providers are the various implementations of RDMA operations, such as InfiniBand™, iWARP, RoCE, etc. Refer to https://tools.ietf.org/html/draft-hilland-rddp-verbs-00.

1.4.16 Reliable Connected QP

Two RDMA Queue Pair endpoints connected with reliable in-order communications of RDMA messages and data.

1.4.17 Reliable Datagram QP

Two or more Queue Pair endpoints using a reliable datagram channel to exchange RDMA messages and data.

1.4.18 RoCE and RoCEv2 RDMA

RDMA over Converged Ethernet definition. Refer to Annex A16 (RoCE) and Annex A17 (RoCE-v2) at www.infinibandta.org.

1.4.19 RNR_RETRY_COUNT

The total number of times that the “Request for Communication (REQ)” or the “Reply to Request for Communication (REP)” sender wishes the receiver to retry Receiver Not Ready (RNR) NAK errors before posting a completion error. Refer to the term “RNR retry count” in InfiniBand™ Architecture Specification Volume 1.

1.5 References

NVM Express® Base Specification, Revision 2.1. Available from https://www.nvmexpress.org.

InfiniBand™ Architecture Specification Volume 1. Available from https://www.infinibandta.org/ibta-specification.

RFC 5040, R. Recio, B. Metzler, P. Culley, J. Hilland, D. Garcia, “A Remote Direct Memory Access Protocol Specification”, October 2007. Available from https://www.rfc-editor.org/info/rfc5040.

RFC 5041, Shah, H., Pinkerton, J., Recio, R., and P. Culley, “Direct Data Placement over Reliable Transports”, October 2007. Available from https://www.rfc-editor.org/info/rfc5041.

RFC5044, Culley, P., Elzur, U., Recio, R., Bailey, S., and J. Carrier, “Marker PDU Aligned Framing for TCP Specification”, October 2007. Available from https://www.rfc-editor.org/info/rfc5044.

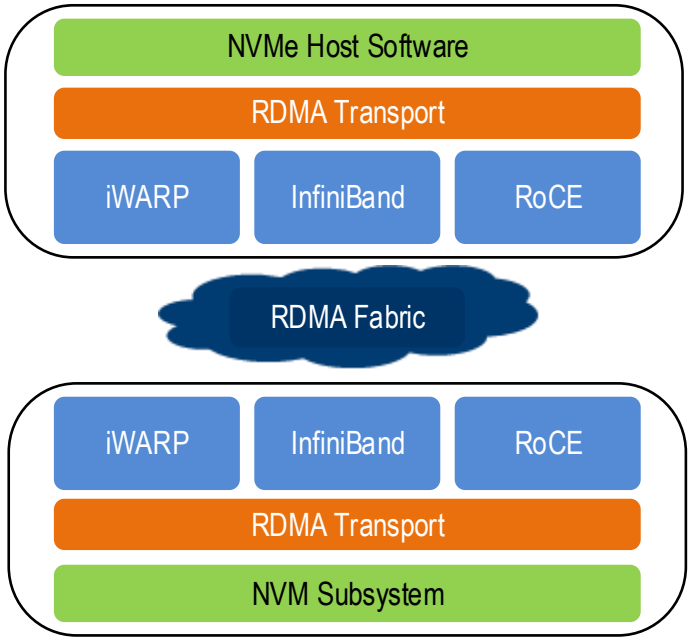
RFC6581, Kanevsky, A., Ed., Bestler, C., Ed., Sharp, R., and S. Wise, "Enhanced Remote Direct Memory Access (RDMA) Connection Establishment", April 2012. Available from https://www.rfc-editor.org/info/rfc6581.

2 Transport Overview

The RDMA transport provides reliable in-order delivery of capsules and direct data placement of Admin and I/O command data through the use of RDMA reliable Queue Pair (QP) modes (i.e., Reliable Connected and Reliable Datagram). The use of RDMA unreliable QP modes is not supported. Refer to the RDMA specifications and RFCs for a description of RDMA Queue Pair modes.

The RDMA transport is RDMA provider agnostic. The diagram in Figure 2 illustrates the layering of the RDMA transport and common RDMA providers (iWARP, InfiniBand™, and RoCE) within the host and NVM subsystem.

Figure 2: RDMA Transport Protocol Layers



The RDMA transport uses a common set of RDMA operations to facilitate the exchange of command capsules, response capsules, and data. These operations are RDMA_SEND, RDMA_SEND_INVALIDATE, RDMA_READ, and RDMA_WRITE. The RDMA transport uses RDMA buffer registration and invalidation operations to facilitate the use of host or NVM subsystem resident buffers for the exchange of data and metadata for Admin and I/O commands.

In some host and NVM subsystem implementations, the interface between the RDMA transport and the RDMA providers is defined by an implementation of RDMA Verbs. When applicable, the host and NVM subsystem RDMA transport implementations should use the common RDMA Verbs software interfaces for the RDMA transport layer to be RDMA provider agnostic.

The iWARP RDMA transport over TCP/IP consists of the RDMAP, DDP and MPA protocols defined by the IETF (refer to IETF RFC 5040, RFC 5041, RFC 5044 and RFC 6581).

2.1 RDMA Command List

2.1.1 RDMA_LOCAL_INVALIDATE

RDMA operation used to invalidate the local system's memory key.

2.1.2 RDMA_READ

RDMA operation used to read from the remote system's memory buffer to the local system's memory buffer.

2.1.3 RDMA_SEND

RDMA operation used to send a message from the local peer's QP Send Queue to the remote peer's QP Receive Queue or Shared Receive Queue.

2.1.4 RDMA_SEND_INVALIDATE

RDMA operation used to perform the RDMA_SEND operation and invalidate the memory key on the remote system.

2.1.5 RDMA_WRITE

RDMA operation used to write memory buffer(s) from the local system's memory to the remote system's memory buffer(s).

3 Transport Binding

3.1 Setup & Initialization

The following section describes the requirements for both RDMA QP creation and the Discovery Log Page Entry definitions that are specific to the RDMA transport.

3.1.1 Transport Specific Address Subtype and Transport Service Identifier

The Discovery Log Page Entry includes a Transport Specific Address Subtype (TSAS) field that is defined in Figure 3 for the RDMA Transport.

Figure 3: Transport Specific Address Subtype Definition for RDMA Transport

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>00</td><td>RDMA QP Service Type: (RDMA_QPTYPE): Specifies the type of RDMA Queue Pair. Valid values are shown in the following table:
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>00</td><td>Reserved</td></tr>
<tr><td>01</td><td>Reliable Connected</td></tr>
<tr><td>02</td><td>Reliable Datagram</td></tr>
<tr><td>03 to 255</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>01</td><td>RDMA Provider Type: (RDMA_PRTYPE): Specifies the type of RDMA provider. Valid values are shown in the following table:
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>00</td><td>Reserved</td></tr>
<tr><td>01</td><td>No provider specified</td></tr>
<tr><td>02</td><td>InfiniBand</td></tr>
<tr><td>03</td><td>RoCE (v1)</td></tr>
<tr><td>04</td><td>RoCEv2</td></tr>
<tr><td>05</td><td>iWARP</td></tr>
<tr><td>06 to 255</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>02</td><td>RDMA Connection Management Service: (RDMA_CMS): Specifies the type of RDMA IP Connection Management Service. Valid values are shown in the following table:
<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>00</td><td>Reserved</td></tr>
<tr><td>01</td><td>RDMA_IP_CM. Sockets based endpoint addressing. For details on the RDMA IP CM Service, refer to the InfiniBand Trade Association specification Annex A11 or the iWARP specification.</td></tr>
<tr><td>02 to 255</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>07:03</td><td>Reserved</td></tr>
<tr><td>09:08</td><td>RDMA Partition Key (RDMA_PKEY): Specifies the Partition Key when AF_IB (InfiniBand) address family type is used. Otherwise this field is reserved.</td></tr>
<tr><td>255:10</td><td>Reserved</td></tr>
</table>

The contents of the Transport Service Identifier (TRSVCID) field in a Discovery Log Page Entry for the RDMA transport depends on the RDMA Connection Management Service (RDMA_CMS) that is used to establish NVMe/RDMA host-controller communication. The RDMA Internet Protocol Connection Management (RDMA IP CM) Service shall be used with the RDMA transport.

The following requirements apply to NVMe/RDMA use of the RDMA IP CM Service:

- The TRSVCID field in a Discovery Log Page Entry for NVMe/RDMA shall contain a TCP port number represented in decimal as an ASCII string;
- If the TRSVCID field in a Discovery Log Page Entry for NVMe/RDMA does not contain a TCP port number represented in decimal as an ASCII string, then the host shall not use the information in that Discovery Log Page Entry to connect to a controller; and

- Hosts and NVM subsystems that support NVMe/RDMA are required to have IP addresses.

These three requirements apply to all RDMA Provider Types.

The RDMA IP CM Service uses the TCP port number indicated by the TRSVCID field as part of establishing NVMe/RDMA host-controller communication. That TCP port number is not used as a component of RDMA protocol endpoint addresses for host-controller communication. RDMA protocol endpoint address establishment and contents are outside the scope of this specification.

UDP port 4420 and TCP port 4420 have been assigned by IANA (refer to https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml) for use by NVMe over Fabrics. NVMe/RoCEv2 controllers use UDP port 4420 by default. NVMe/iWARP controllers use TCP port 4420 by default.

3.1.2 Discovery Log Page Entry Fields

The RDMA Transport's use of the Discovery Log Page Entry fields described in the NVM Express Base Specification is dependent on the value of the RDMA_CMS field. Figure 4 shows the use of these fields. Refer to the Numerical Descriptions section in the NVM Express Base Specification for ASCII string format requirements.

Figure 4: RDMA Transport Discovery Log Page Entry Usage by RDMA_CMS Value

<table>
<tr><th>RDMA_CMS Value</th><th>Discovery Log Page Entry Field</th><th>Discovery Log Page Entry Field Values</th></tr>
<tr><td rowspan="4">RDMA_IP_CM</td><td>TRTYPE</td><td>RDMA Transport</td></tr>
<tr><td>ADRFAM</td><td>AF_INET (IPV4), AF_INET6 (IPV6), AF_IB (refer to the Discovery Log Page Entry in the NVM Express Base Specification)</td></tr>
<tr><td>TRADDR</td><td>The value of this field is dependent on the value of the ADRFAM field. IPv4 and IPv6 address format is described in the Discovery Log Page Entry in the NVM Express Base Specificaton. AF_IB address format is an InfiniBand™ GUID in ASCII string format.</td></tr>
<tr><td>TRSVCID</td><td>The value of this field is dependent on the value of the TRADDR field as shown.
<table>
<tr><th>TRADDR Value</th><th>TRSVCID Value (ASCII string)</th></tr>
<tr><td>IPv4 or IPv6</td><td>TCP or UDP port number: decimal number represented as an ASCII string with no leading zeroes.</td></tr>
<tr><td>AF_IB</td><td>InfiniBand™ Service ID represented as 16 case insensitive ASCII hexadecimal digits (‘0’-‘9’, ‘a’-‘f’, ‘A-F’) in groups of 4 separated by hyphens (‘-‘), (e.g., dead-beef-0123-3210).</td></tr>
</table></td></tr>
</table>

3.1.3 Disabling Submission Queue Flow Control

Hosts and controllers that use the RDMA Transport may disable Submission Queue flow control as part of creating a Submission Queue and Completion Queue pair, refer to the Message-Based Transport Queue Model section and the Connect Command and Response section in the NVM Express Base Specification.

3.1.4 Fabric Dependent Settings

As part of the RDMA QP creation, the host and remote peer exchange the RDMA_READ resources. This exchange is typically facilitated by the RDMA provider.

The host RDMA Transport sets the Inbound RDMA Read Queue Depth (IRD) value and passes that value to the remote peer using these facilities. The remote-peer uses the host's IRD value to limit the number of RDMA_READ operations that the remote-peer issues to the host. Exceeding this limit may result in an RDMA QP error. The remote peer returns an IRD value of 0h to indicate that any host initiated RDMA_READ operations will result in an RDMA QP error.

The host RDMA Transport sets the value of RNR_RETRY_COUNT based on the RDMA provider capability. If the RDMA provider supports RNR_RETRY_COUNT, then the host should set RNR_RETRY_COUNT to infinite (i.e., a value of 7). If the RDMA provider does not support RNR_RETRY_COUNT, then the host

should set RNR_RETRY_COUNT to 0h. The remote peer shall match the host RNR_RETRY_COUNT setting or fail the QP creation.

Figure 5 shows the use of RDMA Private Data for the exchange of NVMe parameters:

Figure 5: RDMA_CM_REQUEST Private Data Format

Bytes	Description
01:00	**Record Format (RECFMT):** Specifies the format of the RDMA Private Data. If a new format is defined, this value is incremented by one. The format of the record specified in this definition shall be 0h.
03:02	**Queue ID (QID):** Refer to the Queue ID definition in the Connect Command and Response section in the NVM Express Base Specification.
05:04	**RDMA QP Host Receive Queue Size (HRQSIZE):** This field indicates the number of RDMA QP receive queue entries allocated by the host's RDMA Transport for capsule reception.
07:06	**RDMA QP Host Send Queue Size (HSQSIZE):** This field indicates the number of RDMA QP send queue entries allocated by the host's RDMA transport for capsule transmission. The value shall be set to the Submission Queue Size (SQSIZE). Refer to the SQSIZE definition in the Connect Command and Response section in the NVM Express Base Specification. This is a 0's based value.
09:08	**Controller ID (CNTLID):** This field is used to indicate the Controller ID if the RDMA QP is for an I/O Queue. If the QID field is cleared to 0h (i.e., the RDMA QP is for an Admin Queue), this field shall be cleared to 0h. If the QID field is non-zero, this field should be set to the value of the Controller ID associated with the creation of this queue.
31:10	Reserved

Figure 6: RDMA_CM_ACCEPT Private Data Format

Bytes	Description
01:00	**Record Format (RECFMT):** Specifies the format of the RDMA Private Data. If a new format is defined, this value is incremented by one. The format of the record specified in this definition shall be 0h.
03:02	**RDMA QP Controller Receive Queue Size (CRQSIZE):** This field indicates the number of RDMA QP receive queue entries allocated by the controller's RDMA Transport for capsule reception as a 1's based value. RDMA Transports that use RNR_RETRY_COUNT flow control may set this field to be less than or equal to the value represented by the 0's based value in the HSQSIZE field specified in Figure 5 (e.g., if HSQSIZE is set to 3, then this field is set to a value less than or equal to 4). RDMA Transports that do not use RNR_RETRY_COUNT shall set this value to be equal to the value represented by the 0's based value in the HSQSIZE field specified in Figure 5 (e.g., if HSQSIZE is set to 4, then this field is set to 5).
31:04	Reserved

Figure 7: RDMA_CM_REJECT Private Data Format

Bytes	Description
01:00	**Record Format (RECFMT):** Specifies the format of the RDMA Private Data. If a new format is defined, this value is incremented by one. The format of the record specified in this definition shall be 0h.
03:02	**Status (STS):** Specifies status for the associated RDMA_CM_REQUEST that is paired with this reject response. The valid status values are specified in Figure 11.

3.2 Queue Model Instantiation

A single I/O Submission Queue and I/O Completion Queue pair shall be mapped to a single RDMA Queue Pair. Multiplexing multiple I/O Submission Queues and I/O Completion Queues onto a single RDMA Connected Queue Pair is not supported. Spanning a single I/O Submission Queue and I/O Completion Queue pair across multiple RDMA Queue Pairs is not supported.

3.3 Data Transfer Model

The RDMA transport is a message-based transport that uses capsules for data transfer as defined in the Capsules and Data Transfer section in the NVM Express Base Specification.

Capsule exchanges are performed using the RDMA_SEND or RDMA_SEND_INVALIDATE operations. The RDMA Transport at the host uses RDMA_SEND to transmit command capsules. The RDMA Transport at the controller uses RDMA_SEND or RDMA_SEND_INVALIDATE to transmit response capsules and optionally invalidate a host memory key. An RDMA_SEND operation contains at most one command or response capsule.

All RDMA_READ and RDMA_WRITE operations are initiated by the controller. Data transfers from a controller to a host are performed using the RDMA_WRITE operation. Data transfers from a host buffer to a controller buffer are performed using the RDMA_READ operation. Data for an I/O command may also be exchanged from the host to the controller using a single RDMA_SEND operation that contains the command capsule and in-capsule data within the RDMA_SEND operation's message payload.

Host Memory Buffer Addresses are communicated in the command's SGL entries. Host Memory Buffer Addresses are represented as the combination of a memory key, offset, and length. The host is responsible for allocating the memory buffers, registering the keys, and constructing the SGL entries with the associated Host Memory Buffer Address.

To ensure proper command data to command completion ordering, all RDMA_WRITE data transfer operations for a command shall be submitted onto the same RDMA QP prior to submitting the associated response capsule onto the same RDMA QP.

The detailed flow of the command sequences using RDMA operations is shown in Figure 8.

Figure 8: Command Sequence Using RDMA Operations

Data Transfer Direction	Command Sequence Using RDMA Operations
No data to transfer	• The host transmits the command capsule to the controller using an RDMA_SEND operation. • Command action completed by the controller. • The controller transmits the response capsule to the host using an RDMA_SEND operation. • The same RDMA QP shall be used for both RDMA_SEND operations.
Controller to host	• The host transmits the command capsule to the controller using an RDMA_SEND operation. The capsule contains or points to SGL(s) required for the data transfer. • The controller uses RDMA_WRITE operation(s) to transfer data from the controller to the host. Each RDMA_WRITE operation is associated with one keyed remote host memory buffer (SGL) and one or more local controller memory buffer(s). • The controller transmits the response capsule to the host using an RDMA_SEND or (optionally) RDMA_SEND_INVALIDATE operation. • The same RDMA QP shall be used for the RDMA_WRITE operation(s) and the RDMA_SEND operation.
Host to controller	• The host transmits the command capsule to the controller using an RDMA_SEND operation. The capsule contains or points to SGL(s) required for the data transfer. The capsule may contain in-capsule data, which is pointed to by an offset address in an SGL within the capsule. • For host-resident command data, the controller uses RDMA_READ operation(s) to transfer the data from the host to the controller. Each RDMA_READ is associated with one keyed remote memory buffer (SGL) and one or more local memory buffer(s). • The controller transmits the response capsule to the host using an RDMA_SEND or (optionally) RDMA_SEND_INVALIDATE operation. • The same RDMA QP shall be used for the RDMA_READ operation(s) and the RDMA_SEND operation.

3.3.1 Capsules

The capsule size for Fabrics commands are fixed in size regardless of whether commands are submitted on an Admin Queue or an I/O Queue. The command capsule size is 64 bytes and the response capsule size is 16 bytes.

The capsule sizes for the Admin Queue are fixed in size. The command capsule size is 64 bytes and the response capsule size is 16 bytes. In-capsule data is not supported for the Admin Queue.

Command capsules for I/O commands sent on I/O Queues may contain up to the I/O command capsule size reported by the I/O Queue Command Capsule Supported Size (IOCCSZ) field multiplied by 16. The response capsule size shall be 16 bytes and shall not contain in-capsule data.

The RDMA Transport facilitates the use of separate locations for SGLs and data (refer to the Data and SGL Locations within a Command Capsule section in the NVM Express Base Specification).

Figure 9: RDMA Capsule Size and SGL Mapping

Capsule Type	Capsule Size (bytes)	SGL Type
Fabrics and Admin Commands	64	Host-resident data buffer only.
Fabrics and Admin Responses	16	n/a
I/O Queue Command	IOCCSZ * 16	Host-resident data buffer or in-capsule data.
I/O Queue Response	16	n/a

Admin command data is transferred using host-resident data buffers specified in Keyed SGL Data Block descriptor entries. I/O command data is transferred using host-resident data buffers specified in Keyed SGL Data Block descriptor entries or within the capsule. The RDMA Transport supports the SGL Data Block with Sub Type Offset, SGL Last Segment with Sub Type Offset, and Keyed SGL Data Block descriptors only. The RDMA Transport does not support SGLs in host memory; all SGLs shall be contained in the command capsule. Fabrics and Admin commands have one Transport SGL Data Block descriptor or Keyed SGL Data Block descriptor (i.e., there are no SGL descriptors following the Submission Queue Entry). I/O commands may have more than one SGL descriptor.

The controller shall set the SGL and Alignment Support (SGLAS) field in the SGLS field to 01b in the Identify Controller data structure (refer to the Identify Controller data structure in the NVM Express Base Specification) (i.e., the controller shall support SGLs and impose no alignment or granularity requirements for data blocks).

There are SGL Descriptor Sub Type values that are specific to RDMA operation as defined in Figure 10.

Figure 10: SGL Descriptor Sub Types Specific to RDMA

SGL Descriptor Types	SGL Descriptor Sub Type	Sub Type Description
All	Ah to Eh	Reserved
0h, 1h, 2h, 3h, 5h	Fh	Reserved
Keyed SGL Data Block (4h)	Fh	**Invalidate Key:** The host uses this SGL Descriptor Sub Type to specify that the controller should remotely invalidate the RKEY. If the controller does not support remote invalidate, then this SGL Descriptor Sub Type is ignored.

3.4 Keep Alive Model

Keep Alive functionality is not supported by all RDMA Provider Types at the RDMA Transport layer. As a result, the RDMA Transport requires the use of the Keep Alive Timer feature (refer to the Keep Alive section in the NVM Express Base Specification). It is recommended that any RDMA provider level functionality be disabled to avoid redundant and conflicting policies.

The RDMA Transport does not impose any limitations on the minimum and maximum Keep Alive Timeout value. The minimum should be set large enough to account for any transient fabric interconnect failures between the host and controller.

3.5 Error Handling Model

3.5.1 RDMA Transport Errors

Errors detected by the RDMA Transport may result in the termination of any command capsule, response capsule, or data transfer operations and may result in the tear down of the RDMA QP(s). The RDMA Transport may detect errors that are not directly associated with a capsule or data transfer operation (e.g., tear down of the RDMA QP due to connection loss, data corruption, or protection error). In the case of a RDMA QP tear down, the RDMA Transport is responsible for terminating the RDMA QP, freeing up any NVMe Transport resources, and then informing the NVMe layer about the termination and the associated cause.

The host and controller may detect connection loss at different times (refer to the Communication Loss Handling section in the NVM Express Base Specification). The host should not assume the controller has detected connection loss if the RDMA Transport has terminated the RDMA QP (e.g., due to host-side failures).

Errors detected by the RDMA Transport during RDMA QP establishment are handled within the RDMA Transport and are not reported to the NVMe layer. These errors are described in Figure 11.

Figure 11: RDMA Transport Errors

Value	Definition
1h	**RDMA Invalid Private Data Length:** The host sent an incorrect RDMA Private Data size.
2h	**RDMA Invalid RECFMT:** The host sent an invalid RECFMT.
3h	**RDMA Invalid QID:** The host sent an invalid QID.
4h	**RDMA Invalid HSQSIZE**: The host sent an invalid HSQSIZE.
5h	**RDMA Invalid HRQSIZE:** The host sent an invalid HRQSIZE.
6h	**RDMA No Resources**: The controller-side RDMA transport is unable to create the RDMA QP due to lack of resources.
7h	**RDMA Invalid IRD:** The host sent an invalid IRD value.
8h	**RDMA Invalid ORD:** The host sent an invalid ORD value.
9h	**RDMA Invalid CNTLID:** The host sent an invalid CNTLID value.
Ah to FFh	Reserved

3.5.2 RDMA Provider Errors

Errors detected by the RDMA provider are communicated to the local NVMe RDMA Transport through implementation specific interfaces (e.g., RDMA Verbs). RDMA providers may have facilities to communicate errors to the RDMA QP remote peer. Details of these facilities or their use is outside the scope of this specification. Details on the types of errors and their associated identification encoding is contained within the RDMA provider specifications.

3.6 Transport Specific Content

3.6.1 Key Management

An RDMA Memory Key (RKEY) is an identifier that associates a data buffer in host memory with a registered “remote access enabled” RDMA memory region or memory window on an RDMA NIC (RNIC) attached to a host. The host NVMe RDMA transport software manages the creation of RKEYs, association of the RKEYs to data buffers, insertion of RKEYs into SGL entries, and invalidation of the RKEYs upon completion of Fabrics, Admin, or I/O commands. Refer to the Scatter Gather List (SGL) section in the NVM Express Base Specification for the definition of the Keyed SGL Data Block descriptor.

Commands that require data transfers between a host memory buffer and the controller shall use SGLs that contain a full RDMA host address tuple consisting of an RKEY, offset, and length. The host NVMe RDMA transport is responsible for allocating this tuple by registering the associated data buffers with the appropriate RNIC to produce the RKEY and then subsequently inserting the RKEY, offset, and length into the SGL entries associated with the command. The same RKEY may be used in multiple SGL entries

associated with the same Fabrics, Admin, or I/O command. The RKEY shall be invalidated only after all RDMA_READ or RDMA_WRITE operations have been completed that use the RKEY.

The host RDMA transport software selects one of two methods to invalidate the RKEY: local invalidate or remote invalidate. To indicate a remote invalidate, the host sets the SGL Descriptor Sub Type field in the Keyed SGL Data Block descriptor to Fh (refer to Figure 10). If the controller RDMA transport does not support remote invalidate, then the host's request for remote invalidation is ignored.

The controller RDMA transport may or may not honor the remote invalidate request. If honored, the controller RDMA transport invalidates the RKEY by using the RDMA_SEND_INVALIDATE operation to return the capsule response. If the command capsule contains multiple SGL entries with the SGL Descriptor Sub Type field in the Keyed SGL Data Block descriptor set to Invalidate Key (i.e., Fh), the controller RDMA Transport shall only invalidate the RKEY in the last Keyed SGL Data Block descriptor.

The host RDMA transport shall check the RDMA Receive Queue to determine if the RDMA_SEND_INVALIDATE was received and check the value of the RKEY that was invalidated. If the controller RDMA Transport did not invalidate the RKEY as requested, the host is responsible for invalidating the RKEY using a local invalidate operation.